Implementing an Organizational Software Process Improvement Program

Beth Layman

This article describes in overview the key activities involved in implementing a software engineering process. Since this is such a broad topic, the article will focus primarily on describing the "meta-process" that allows an organization to create, use, and improve a set of documented processes that guide software development and maintenance activities. Since this meta-process represents a set of ongoing activities, we will refer to it as a program. The steps we will outline below are complementary to various process improvement paradigms and frameworks such as Deming's PDCA (Plan-Do-Check-Act), the Software Engineering Institute's IDEAL model [1], and ISO/IEEE standards (e.g., 9000 series, 12207, 15504).

1. PLANNING FOR PROCESS IMPROVEMENT

Process improvement programs are commonly initiated because a software developing organization is encountering problems or issues that have made apparent the need for change. These are typically issues such as customer dissatisfaction, inadequate software quality, inability to deliver on time and within budget, and excessive rework. When the "pain" these issues cause is felt and understood at all levels of the organization, the impetus for a process improvement program is strong. When process improvement is initiated and sponsored by the highest levels in the organization, this upper management support and involvement bodes well for the program.

In some cases, the impetus for process improvement may derive from the need to achieve a customer-required certification or proficiency level. In others, it may spring from little more than a vague desire to "do better." In yet other cases, the motivation may be about little more than management bragging rights. The first of these provides compelling motivation for the entire organization, but the others lack focus and portend programs that are not likely to gain enthusiastic buy-in at all levels of the organization. Commitment (or lack thereof) is a crucial risk factor, and it should be understood that process improvement initiatives come with no guarantee of success [2].

Once the motivation for process improvement exists, it is necessary to consider the strategies to be used in establishing the program. Using a navigation analogy, we might think of the challenge of process improvement as one of planning routes to various destinations, where the "destinations" are solved problems. We could conceivably begin by mapping the entire software development terrain, but it is reasonable to wonder if there are any existing roadmaps to aid us in charting our problem-solving course. Indeed there are, in the form of several "models" for software process. One such guide is the well-known Capability Maturity Model for Software® (CMM®), developed by the Software Engineering Institute at Carnegie-Mellon University. The SW-CMM has been used successfully as a guide to process improvement effort for more than 10 years [3]. The model provides a roadmap for process improvement by prioritizing improvement efforts into a framework of "maturity" levels.[1] These levels are structured in such a way that they represent a natural progression of process competence, proficiency, and sophistication. An organization uses the model as a guide by determining where on the scale of maturity levels its process currently stands, and then working toward satisfying the criteria for the next-higher level.

So, which model to use? Here we will consider two: the SW-CMM or the emerging, expanded model, the Capability Maturity Model Integration® (CMMI℠)[2]. The SW-CMM will "sunset" at the end of 2005,[3] so many organizations are choosing or switching to the CMMI℠. Because the CMMI℠ is a more comprehensive model than the SW-CMM, it requires additional decisions to be made concerning coverage (what disciplines within the organization are to be included) and representation (whether to use the model in a "staged" or "continuous" manner, or both[4]). These additional decisions do not alter the overall roadmap for process improvement that we will be considering here.

Beyond the choice of a guiding model, there remain yet further decisions to be made regarding process improvement strat-

[1]It is the author's experience that, in the absence of a guiding framework such as the CMM, process improvement programs usually flounder or stall due to lack of focus, and are then cancelled before significant results can be realized.
[2]See Chapter 9 of this volume for overviews of the CMM and CMMI.
[3]For the rationale behind the sunset of SW-CMM, see the FAQ at http://www.sei.cmu.edu.
[4]Although this article is generally relevant to either of these approaches, the second, more common, approach is emphasized throughout.

egy and approach. Assuming an organization to be at CMM Level 1, it has two options. The first option is an improvement approach that focuses on building process capability at the project level, which is to say that each project documents its own approach to basic project management practices. Using this approach, the emphasis is on documenting what people are currently doing and allowing everyone a role in evolving their parochial process until it meets the model's criteria (repeatability) for Level 2. The second option entails defining a more standardized process that all projects are expected to follow. Such a standardized process is the ultimate goal of most organizations for their process improvement program, and it requires a well-coordinated process development and implementation effort [4].

Objective assessment of an organization's current capability is a necessary first step in implementing a process improvement program. Assessment is usually a watershed event that serves as a baseline or you-are-here indicator. Using the staged representation of the CMM/CMMI, this indicator will be the stage or level to which the organization currently conforms. Once this has been established, the organization is in a position to set realistic improvement goals, the typical "waypoint" being the next-higher level of the model. The assessment highlights both the organization's strengths and weaknesses in comparison to the model's best practices. These findings are then used to develop process improvement plans. There are two formal assessment methods recognized within the industry: the CMM-Based Appraisal for Internal Process Improvement (CBA IPI) and the Standard CMMI^{SM} Appraisal Methods for Process Improvement (SCAMPI^{SM}); however, less formal and costly methods (e.g., baseline or mini assessments) are often used to establish an organization's process maturity baseline.

A person from outside the organization is usually selected to lead an assessment. This person should be knowledgeable in both the model and assessment methods.[5] Assessments involve a focused review by an in-house team consisting of representatives from projects recently completed or currently underway. The assessment includes interviews with project personnel, as well as a review of available process descriptions and project work products/artifacts. This analysis uncovers the organization's strengths and weaknesses, which findings are typically presented formally at the end of the assessment. Although these findings are rarely a surprise to the development group, the event serves as a public acknowledgement of those areas in which improvement is critically needed.

Both strengths and weaknesses are used in planning for process improvement. Some projects in the organization may be performing at a higher level than others, and strengths identified in these projects represent an opportunity to leverage and transfer good practices to other projects or parts of the organization. Weaknesses, of course, pinpoint areas of focus for the process improvement program. It is also common for themes to emerge during the course of an assessment, and these help in planning for improvement. For example, the lack of measures of project performance is a common thematic deficiency of Level 1 organizations. This lack hinders a number of CMM Level 2 practices, and would likely show up in multiple weaknesses identified during an assessment. This theme, then, can often be addressed as a single improvement opportunity.

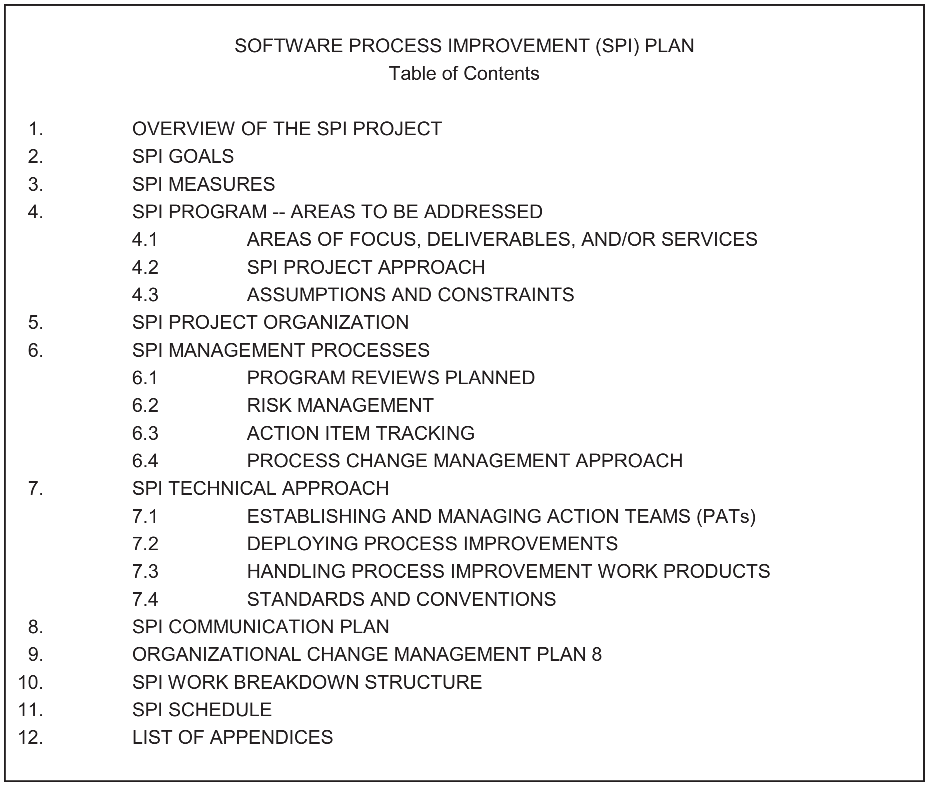
Regardless of the approach used, process improvement programs are doomed to fail without good planning. A tenet used by the author is to treat software process improvement like any other project. I often tell SPI champions they need to "eat their own dog food" and use the same disciplined approach advocated for any significant software project. A process improvement program should have a SPI plan, associated budget and schedule (See Figure 1). The plan should outline purpose, scope, strategy, tactics, SPI infrastructure (see Section 2), and methods of operation (meetings, risk management, measures, etc.).

2. ESTABLISH THE PROCESS IMPROVEMENT INFRASTRUCTURE

Roles and responsibilities for the process improvement program are typically established as part of the start-up and planning activities. Sometimes, there are preexisting forums, working groups, or leadership teams that can assume these roles and responsibilities. The infrastructure required to support process improvement depends on a number of factors, including the organization's culture and size, the diversity of software groups, and the SPI approach chosen. It is best when the infrastructure incorporates all levels of the organization. There are typically three or four infrastructure elements (see Figure 2):

- **Executive Leadership Group (ELG)**—Serves as the sponsoring team for the program; responsible for reviewing and approving SPI plans, budgets, and for periodically reviewing SPI progress and results. The ELG may be subsumed into the MSG in smaller organizations.
- **Management Steering Group (MSG)**—Typically made up of a cross-functional set of influential and well-respected middle managers. This group sets the pace and tone for the program and has primary accountability for its success. Regularly presents plan details, progress, and status reports to the ELG.

[5]SEI sponsors an authorization program for lead assessors. Candidates are required to have extensive software/systems engineering and management experience. They must complete comprehensive courses in the model and CBA IPI or SCAMPI methods, and successfully undergo an observed field trial in the role of leading an assessment.



SOFTWARE PROCESS IMPROVEMENT (SPI) PLAN
Table of Contents

1. OVERVIEW OF THE SPI PROJECT
2. SPI GOALS
3. SPI MEASURES
4. SPI PROGRAM -- AREAS TO BE ADDRESSED
    - 4.1 AREAS OF FOCUS, DELIVERABLES, AND/OR SERVICES
    - 4.2 SPI PROJECT APPROACH
    - 4.3 ASSUMPTIONS AND CONSTRAINTS
5. SPI PROJECT ORGANIZATION
6. SPI MANAGEMENT PROCESSES
    - 6.1 PROGRAM REVIEWS PLANNED
    - 6.2 RISK MANAGEMENT
    - 6.3 ACTION ITEM TRACKING
    - 6.4 PROCESS CHANGE MANAGEMENT APPROACH
7. SPI TECHNICAL APPROACH
    - 7.1 ESTABLISHING AND MANAGING ACTION TEAMS (PATs)
    - 7.2 DEPLOYING PROCESS IMPROVEMENTS
    - 7.3 HANDLING PROCESS IMPROVEMENT WORK PRODUCTS
    - 7.4 STANDARDS AND CONVENTIONS
8. SPI COMMUNICATION PLAN
9. ORGANIZATIONAL CHANGE MANAGEMENT PLAN 8
10. SPI WORK BREAKDOWN STRUCTURE
11. SPI SCHEDULE
12. LIST OF APPENDICES

Figure 1. Sample table of contents for an organizational process improvement plan.

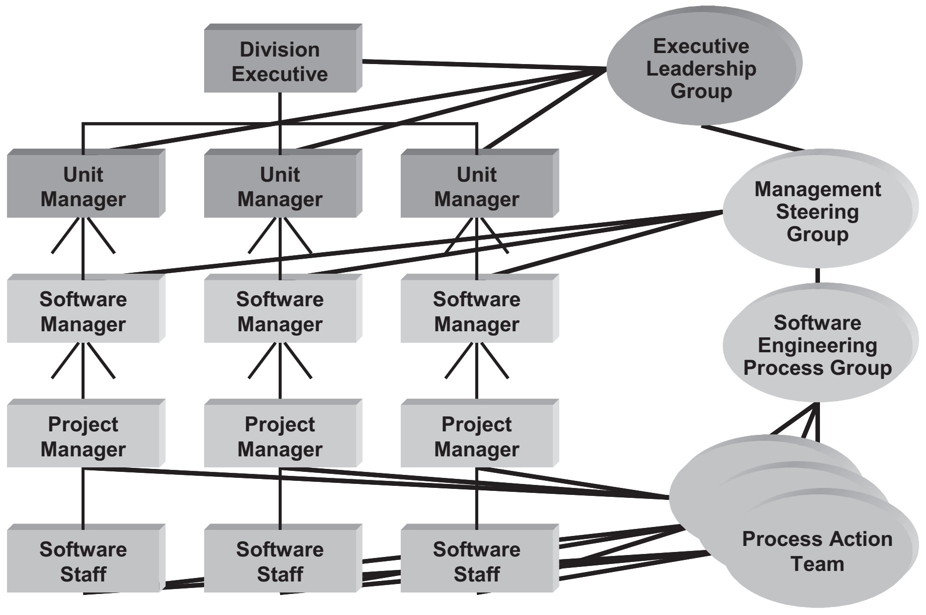


Figure 2. Functional organization and the "virtual" process improvement infrastructure. (Source: TeraQuest *Process Improvement Leadership* Courseware, 2003.)

- **Software Engineering Process Group (SEPG)**—Typically composed of senior software engineering professionals from across disciplines: analysts, architects, developers, testers, configuration management specialists, and so on. Members often serve on a rotating basis and may be part-time only. May be led by a full-time manager who also sits on the MSG.
- **Process Action Teams (PATs)**—Formed as needed to undertake some process development or improvement task (e.g., develop and pilot a standard process for conducting code inspections). Often led by a member of the SEPG and made up of part-time subject matter experts. Teams are dissolved once their task has been completed.

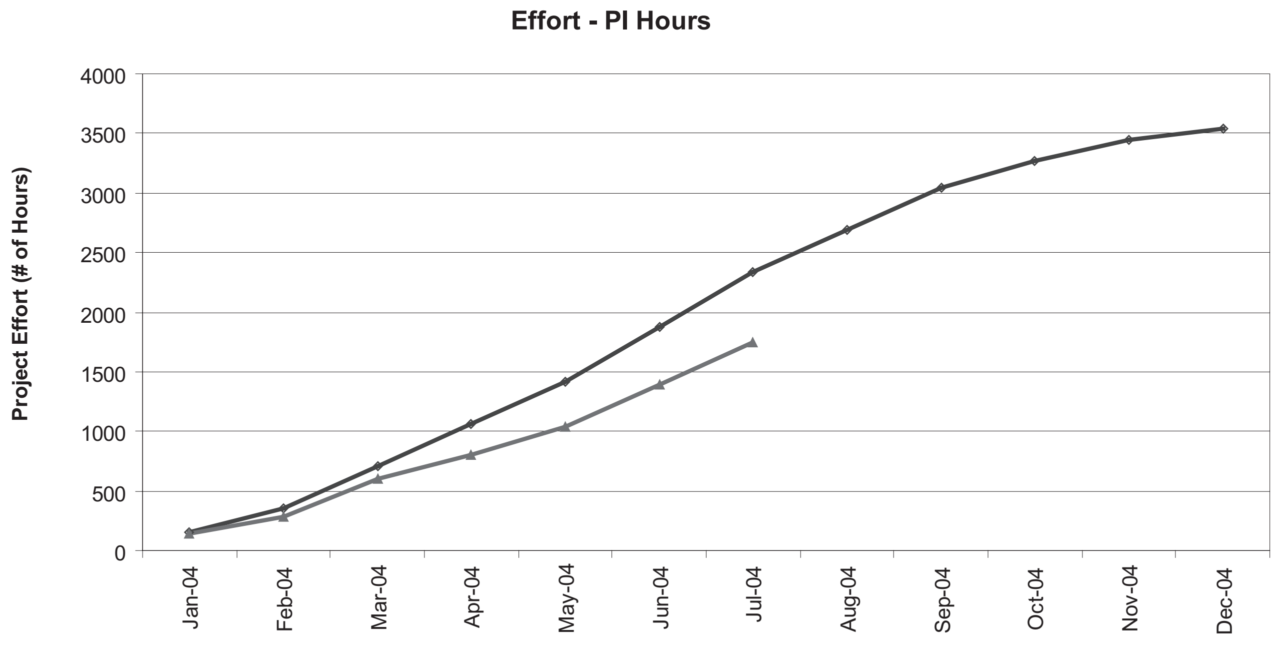
Perhaps the most critical success factor in establishing an infrastructure is the involvement of senior management. Senior management's role is to exhibit a constancy of purpose regarding SPI, and to continuously reinforce the importance of process discipline through word and action. They do this by funding SPI, showing interest in the progress of the program, removing obstacles to SPI, enforcing process discipline throughout the organization, and so on.

The infrastructure, of course, must be underwritten with funding, equipped with tools, and staffed with personnel who have the skills necessary for making progress. A SEPG, for instance, may need to acquire skills in process definition, process measurement, and change management, as well as project management skills for planning and tracking the work of action teams.

One common problem arises from the inherent conflicting priorities of part-time staff on the MSG, SEPG, and PATs. Programs stall when the allocation of an individual is not explicitly planned and managed. For example, a written contract might be drawn stating a resource commitment of 20% to process improvement, allowing 80% to remain devoted to normal software development effort. It is good practice to formally plan resource requirements, and visibly track time spent on SPI (see Figure 3).

3. DEFINING PROCESSES

Once an organization knows where it needs to improve and has dedicated resources to making improvements, the next challenge is to define a set of processes that meets the organization's unique needs and is easy to follow. For lower-maturity organ-



	Jan-04	Feb-04	Mar-04	Apr-04	May-04	Jun-04	Jul-04	Aug-04	Sep-04	Oct-04	Nov-04	Dec-04	Jan-05	Feb-05	Mar-05	
Plan 1 (01/04)	155	355	710	1065	1420	1880	2340	2695	3050	3270	3445	3545				
Actual Hours	142	283	603	803	1040	1395	1750									
Effort Variance (%)	8.39%	29.50%	9.86%	43.66%	33.24%	22.83%	22.83%									

Figure 3. Example of an indicator of planned versus actual time spent on process improvement activities.

izations, this process definition step represents the bulk of the initial work. The same quality attributes we strive to build into software have relevance here. Processes must be relevant, correct and consistent; they also must be usable, flexible, extensible, and adaptable. Many organizations jump hastily into process definition without considering that, just as with software applications, processes must be designed with these quality attributes in mind. Rushing into process definition will almost certainly result in an unwieldy process set that will fail Humphrey's description as "fit for use" [5].

At the highest level, an organization's process set includes *process descriptions, process assets,* and *process-related measures.* Figure 4 shows how these process components are related. As software development activities (represented by Project XYZ) are performed, the process users, that is to say the software development project staff (project managers, analysts, developers, testers, and other team members), refer to process descriptions to guide them in doing their work. Development activities are carried out and deliverables or work products are produced in accordance with the documented process. Examples of such products would be meeting agenda and minutes, plans, estimates, designs, software, tests, documentation, and so forth. Process assets are made available to users to help them as they execute the process. Examples of process assets include templates, forms, checklists, guidebooks, and even sample deliverables from previous projects.

Process-related measurements are made throughout the course of a project. These quantify such things as size, complexity, and effort expended (just to name a few possibilities). Measurement helps in gauging a project's *performance* and can be used in real time to manage the project. However, measures accumulated from many projects help characterize process *capability.* This cumulative measurement refines the organization's understanding of how a process can be expected to perform under typical circumstances, making it possible to better predict and plan future project performance. Analyses of measurement data also make it possible for the organization to recognize positive or negative trends and identify areas on which future process improvement effort should be focused.

Process Description

The process is typically described using three levels of abstraction:

- **Policy**—A statement (often by senior management) of the organizational expectations for the process, when the process is to be followed (under what conditions), and why it is needed.
- **Process**—Unfortunately, the term "process" has been overloaded. Here, we use it to name the "what" level of process description. It outlines what the process accomplishes, identifying key activities, roles, and deliverables, and portraying

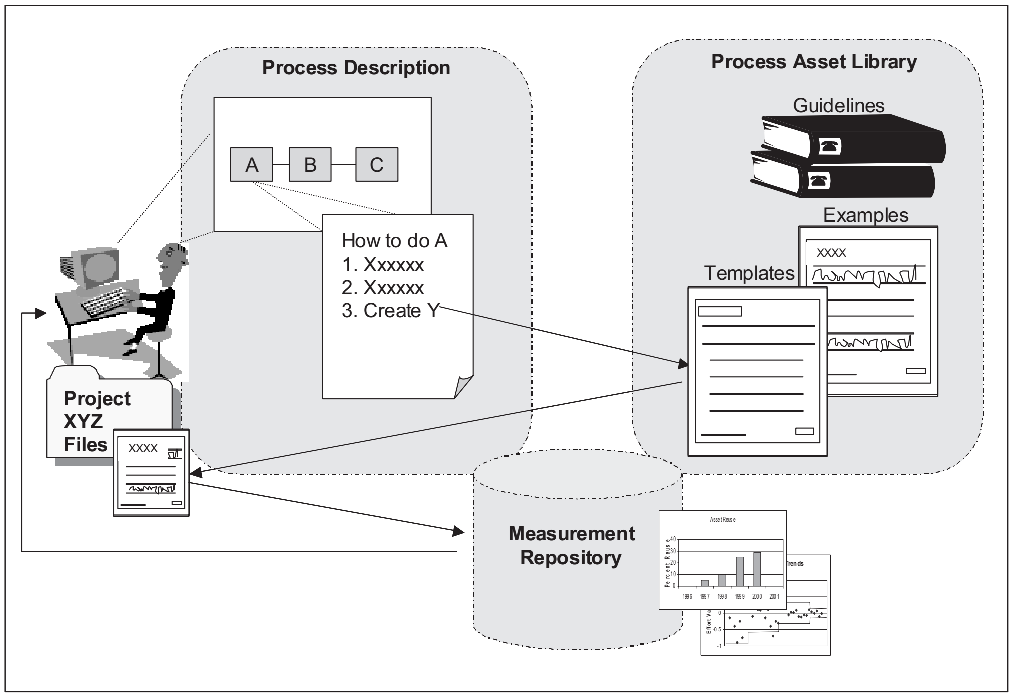


Figure 4. The three major components of a standard process.

relationships and interconnections between process components (e.g., the relationship between project planning and risk management). This description may be depicted graphically, textually, or both.

- **Procedure**—Defines "how" each component part of the process is conducted. Typically, a step-by-step description is made that identifies specific techniques, methods, and tools to be used.

To ensure consistency and ease of use, process descriptions should be written to adhere to established process design standards.

In addition to the levels outlined above, process elements such as the following should be incorporated as required to create a process specification or operational definition:

- **Role**—A defined set of role titles and descriptions should be used consistently throughout process descriptions. It is helpful to identify each role's level of participation in an activity or deliverable (for example, the RACI model [6] is sometimes used to show whether a role is *responsible* for, *accountable* for, *consulted* with, or simply *informed of* the event).
- **Activity**—Work that is performed to carry out a process. This generic term is used to refer to a breakdown into phases, steps, tasks, and so on, and sometimes organized into a hierarchy of work detail in a process description.
- **Work Products**—The tangible items consulted while performing an activity or produced as a result of performing the activity. Synonyms are "deliverable" and "artifact."

Methods for documenting process descriptions are varied, and include flowcharts, activity diagrams, swim lanes, process maps, and techniques such as EXTM (entry, exit, task, measure) for textual materials.

The exact form of process documentation will vary depending upon several factors including the organization's business context, "legacy" process documentation, life cycle models used, software development "methodologies" employed, the size and complexity of the organization, and the range of software development work it undertakes. For some organizations, the middle "process" level may consist of the set of approved life cycle models (so that process descriptions are built around a life cycle model.) In other organizations, the process description may be independent of life cycle, and the merging of the two occurs as a part of process "instantiation"; that is, the process to be used and development strategies such as life cycle to be followed are determined during project planning. The resulting merger of process and strategy is considered the project's defined process.

Process Assets

Process assets include anything helpful in executing a process. Typically, these assets are housed in a library where process users have ready access during process execution. A process asset library (PAL) typically includes:

- **Templates**—"Shells" that can be used for creating work products outlined in the process descriptions. Templates encourage a standard look and feel for work products such as project plans, test plans, requirements, design documents, and meeting agenda and minutes. They also speed up creation and review of work products. Finally, procedural information incorporated in a template can improve process compliance by creating a real-time process review for the template user.
- **Automated Tools**—Software that automates performance of activities or the development of work products. Example: automated requirements management tools can be used to define, store, review, approve, and manage evolving requirements on a software project.
- **Guidelines**—Additional how-to detail useful for executing an activity. Such rules of thumb are meant as guidance, but conformance is not required. Guidelines are especially useful when an activity is performed infrequently, is being performed for the first time, when there are multiple ways of performing a process, or when one or more techniques can be used depending on conditions.
- **Standards**—Rules applicable to a particular type of work product (e.g., coding standards for programs written in a particular programming language).
- **Checklists**—An aid in assessing the completeness and/or correctness of an activity or work product. Checklists typically take the form of a series of questions that serve as a reminder to process users or reviewers.
- **Examples/Lessons Learned**—Good examples from past projects can be helpful process assets. Simple statements of lessons learned on past projects, when they are well organized and easily accessed, are good additions to a process asset library.

- **Training Material**—Slides or other reference materials from formal training on process, tools, or methods can be valuable assets.

Process-Related Measures

The basis for measurement exists as the natural byproduct of process execution (e.g., planned versus actual size/scope, schedule, effort/cost, progress, quality.) Unless automatic, or nearly automatic, capture mechanisms are built into processes; however, measurements will be difficult to obtain. In organizations at lower maturity levels, simple measures of planned versus actual data are captured to help track the project. For instance, the time required to perform task A in the process shown in Figure 4 may have been estimated to take 80 hours but, in actuality, it took 100 hours on this project. The original estimate is captured during the planning process and "actuals" are then captured through a time-reporting or project management system. These data can be used at the task level to take corrective action on individual activities that are underperforming, and can also be used cumulatively at the project level to monitor the progress and overall status of the project.

No one single measure can accurately characterize the health of a project; rather, a set of "key indicators" covering the critical dimensions of scope/size, effort/cost, schedule, and quality are needed [7]. Bringing planned-versus-actual data together at the project level often involves use of measurement spreadsheets, databases, or other tools.

There are numerous and varied uses for measurement when a common set of measures can be captured across many projects in an organization. This requires the development of some type of measurement repository. Projects not only submit data to the repository, but access and use it, particularly as a resource for estimating key planning parameters on similar projects. The repository should capture not only quantitative data on project performance, but project characteristics such as project type, team experience, tools used, and so on. These data provide a context for measurement analysis and reporting. Questions like the following can begin to be answered [8]:

- How consistently are we performing?
- Have we seen performance/quality improvements over time?
- How does our performance map to our goals?
- Which processes are still in need of improvement?
- What is the ROI of our improvement efforts?
- How is our performance compared to others?
- In what training/tools should we invest?

Organizations at higher levels of maturity can use measurement data to establish statistical control limits and manage processes [9].

Process Definition Challenges

Some organizations struggle with the issue of whether to have one or many process set(s). A key determinant is how much variation is necessary in order to satisfy diverse requirements across business units, locations, departments, and/or clients. Somewhat related to this is the diversity of software and system domains, hardware and software platforms, and system vintage. Large organizations in which many of these factors may be present may develop common policies and some common processes but allow specialized local processes and procedures to be applied where appropriate. Alternatively, the organization may publish guidelines under which projects are permitted to tailor standard processes to meet their unique situation.

Two common risks facing software process improvement programs are:

- The tendency to remain too long in process definition mode while not showing any improvement. Good planning and the use of an incremental approach in which processes are piloted and rolled out in stages can help mitigate this problem.
- Developing processes that are not "fit for use." This may be the result of defining processes in so much detail that they are difficult to follow. This can also happen when processes are written based on theory (or model requirements) rather than actual practice and organizational realities, as when the people charged with defining the processes lack practical experience or are not well versed in the organization's business realities. Hiring outside consultants or industrial engineers with no background in software development to define processes is an obvious route to such problems.
- The organization's maturity must also be considered when developing processes. After all, process improvement is

meant to be an ongoing, continuous activity. Processes are not static, but rather dynamic entities that will evolve to incorporate added complexity and increased sophistication as the organization matures.

It should go without saying that following a defined process should make life easier, not more difficult, for the people in a software developing organization. With good processes, people know what they are supposed to do, and their creativity can be directed to the business and technology challenges, not toward inventing processes.

Process Implementation/Roll-out

cess implementation is concerned with process *fidelity,* the faithfulness with which the process is followed. In other words, once processes have been developed, the organization needs to ensure that (the processes 1) the processes are usable, and that users (2) understand the processes, (3) can access the processes and assets, and (4) follow the processes consistently and correctly. One of the most painful situations to witness is when an organization invests in defining processes, but then fails to deploy them successfully. Let's face it—process improvement is a matter of asking the organization to change, and we all know that change is very hard. The "change agents" of a good process improvement team will have learned in advance about managing change and will have planned accordingly.

Good SPI programs publish major changes to a process only following peer reviews (minimally) or pilots (ideally) of the proposed processes. Walkthroughs of the proposed processes and assets are a good way to get early feedback from the target community of process users who are not actively working on the PAT. Piloting is a further means of verifying that the processes will work in actual practice. Can the process descriptions be faithfully executed? Can templates be completed (efficiently)? Pilots allow the organization to make adjustments based on actual results prior to general deployment. A pilot can be confined to a completed process "chunk" (e.g., estimation, risk management, peer review, unit testing, etc.) and can be run by members of the PAT that created the process. They can train, mentor, coach, and monitor adoption on the pilot project. They must be ready to alter the process and assets based on pilot use. Another possibility is to pilot the processes on the PAT member's own project(s). Since pilots can fail due to inadequate resources rather than process deficiency, adequate resources must be allocated to ensure that the pilot can succeed. If at all possible, pilots should demonstrate (preferably with quantitative data) that the process represents an improvement upon over what went before. These results should then be published during the general rollout.

After pilots have been conducted and processes have been revised based on feedback, they can be rolled out to the rest of the organization. A common approach is to allow ongoing projects to proceed without disruption, but to require all new projects to follow the new processes.

Roll-out usually entails some form of training which that officially introduces the new or revised processes. This training includes overviews of the process set, individual process descriptions and assets, and tutorials in any newly introduced techniques or tools. Both internally developed and vendor-provided coursework may be required.

One good practice is to conduct a "project launch workshop" [10] at the outset of each project. During this session, the project team develops estimates and plans. This exercise should naturally entail a review of the process set, and consideration of how the processes will apply to the project. The project launch workshop, consequently, serves to provide just-in-time training.

How can you ensure that people will actually refer to processes as they execute their work? Paper-based process descriptions are rapidly becoming a thing of the past. Most organizations now opt for the ease of reference and navigation a Web-based interface provides. Typically made available via the organization's intranet, online process documentation allows project team members to readily access and switch between process descriptions, process assets, tools, and project workspace, right from their workstations.

How can you know processes are being adopted and used correctly? The Software Quality Assurance KPA (Key Practice Area) in the SW-CMM (PPQA in CMMI) was designed to help answer this question. This KPA requires that organizations review current projects to objectively determine their conformance to documented processes and report results back to project teams and to management. This can serve two important functions: (1) to mentor and coach individual projects throughout their life cycle regarding process, and (2) to provide the SPI program with the quantitative information needed to answer the following questions:

- Are projects adopting the new processes?
- Is the trend in compliance increasing throughout the organization?
- Will we be ready for an assessment in x months?
- Which processes are generating the most noncompliances?

5. CONTINUOUS PROCESS IMPROVEMENT

As discussed in at the beginning of this article, process improvement can and should be viewed as an ongoing program. Once an initial set of processes has been rolled out, the program should be reviewed and adjusted at least annually. Possible sources of input to drive the future direction of the program include:

- **User feedback.** Process users should have an easy mechanism for providing feedback to the SEPG. This feedback becomes the basis for process change. It is the equivalent of change requests in a software project and should be dealt with similarly. At the meta-process level, lessons learned concerning process development and roll-out should be used to direct future PI strategies and plans.
- **SQA** and **model-based assessment findings.** Continuous SQA audit of process adoption and compliance provides valuable feedback to the PI program. Periodic assessments against the selected reference model highlight additional areas needing improvement. Both forms of monitoring identify problems in process descriptions and assets, as well as problems in the process improvement program approach itself (Figure 5).
- **Measurement data.** Analysis of measurement data across projects can help identify areas needing improvement. For example, if projects are still overrunning budgets and schedules, a root cause analysis is in order to feed refinement of the estimating, planning, scheduling, tracking, or requirements management processes at fault.

Demonstrating return on investment of process improvement has proven to be a challenge. Cost is usually not difficult to measure. When the “hands on” approach to PI advocated in this paper is used, cost is mostly incurred in human capital. Benefit, on the other hand, is more difficult to gauge. A common difficulty is the lack of good quantitative data at the start of a process improvement program, so that no historical baseline exists against which later performance can be compared [11]. It is useful to plan into early efforts the compilation of a baseline that directly relates to the goals of the business and the process improvement program.

Lastly, it is important that the process improvement program remain in alignment with the goals of the business. Most programs need a way to demonstrate quantitatively that the PI program is contributing positively to goal achievement. Careful consideration should be given to how goals are expressed, because this bears on the business sense, alignment, and evaluation of the program. There are instances where advancement to a higher CMM maturity level may be a direct business goal (as a basis for competing in a contract environment, for example), and one whose attainment can be easily measured. More commonly, however, applying the CMM reference model is merely a tactic employed to help achieve business goals such as improved time to market, quality, predictability, and customer satisfaction.

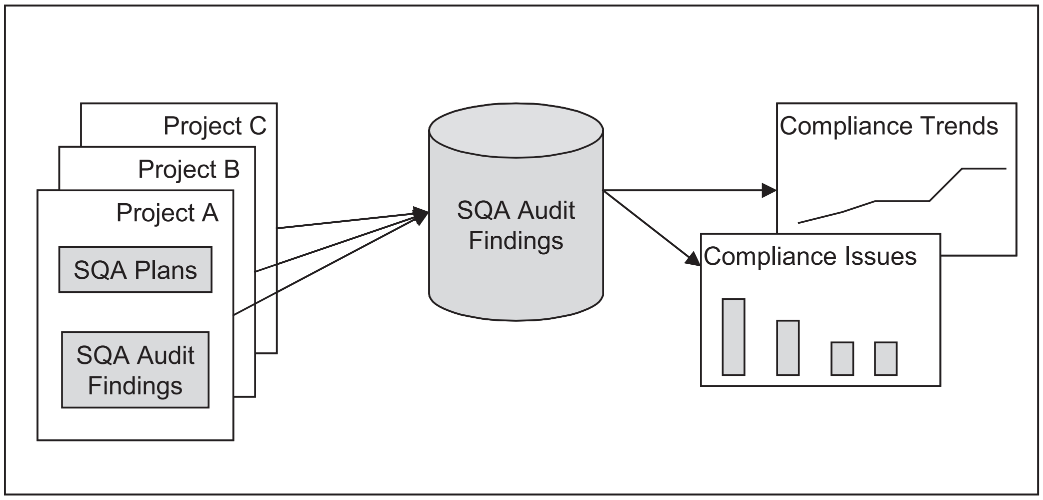


Figure 5. Use of SQA audit findings to report on process compliance.

REFERENCES

1. The IDEALSM Model: A Practical Guide for Improvement: http://www.sei.cmu.edu/ideal/ideal.bridge.html.
2. El Emam, Khaled, Dennis Goldenson, James McCurley, and James Herbsleb (1998), *Success or Failure? Modeling the Likelihood of Software Process Improvement,* International Software Engineering Research Network Technical Report ISERN 98-15, v05, 8/05/98.
3. Paulk, Mark C., Bill Curtis, Mary Beth Chrissis, and Charles V. Weber (1993a). *Capability Maturity Model for Software, Version 1.1.* Software Engineering Institute Technical Report SEI-93-TR-024, February, 1993.
4. Software Engineering Institute, *CMMI for Systems Engineering /Software Engineering, Version 1.1 (Staged Representation),* CMU/SEI-2002-TR-004, Software Engineering Institute, Pittsburgh PA, December 2001.
5. Humphrey, Watts S. *A Discipline for Software Engineering.* Reading, MA: Addison-Wesley, 1995.
6. An overview of RACI responsibility modeling can be found at: http://www.army.mil/aeioo/toolkits/documents/tools/3/RACI_Approach.pdf.
7. McGarry, John, David Card, Cheryl Jones, Beth Layman, Elizabeth Clark, Joseph Dean, and Fred Hall. *Practical Software Measurement, Objective Information for Decision Makers.* Reading, MA: Addison-Wesley, 2002.
8. TeraQuest's Measurement Planning Workshop v2.3. See the training section of www.teraquest.com.
9. Beth Layman and Charles Weber, "Measurement Maturity and the CMM: How Measurement Practices Evolve as Processes Mature," *Software Quality Practitioner, 4,* 3, 2002.
10. TeraQuest's Project Launch Workshop v.1.8. See the training section of www.teraquest.com.
11. Beth Layman and Joyce Statz, "Navigating the Speed Bumps for Measurement and Analysis," Tutorial presentation, SEPG Conference, March 2004, Orlando, FL.