Technical Report

CMU/SEI-90-TR-024
ESD-90-TR-225

Software Engineering Process Group Guide

Priscilla Fowler
Stan Rifkin

September 1990

Technical Report
CMU/SEI-90-TR-024
ESD-90-TR-225
September 1990

Software Engineering Process Group Guide

Priscilla Fowler
Technology Applications Project

Stan Rifkin
Software Process Development Project

with an appendix by
David M. Card
Computer Sciences Corporation

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This report was prepared for the SEI Joint Program Office HQ ESC/AXS

5 Eglin Street

Hanscom AFB, MA 01731-2116

FOR THE COMMANDER

(signature on file)

Thomas R. Miller, Lt Col, USAF, SEI Joint Program Office

This work is sponsored by the U.S. Department of Defense.

Table of Contents

List of Figures

Preface

The Software Engineering Institute (SEI) was established in 1984 at Carnegie Mellon University as a federally funded research and development center. Its charter includes the directive to expedite software engineering technology transfer leading to "rapid improvement of the quality of operational software in the mission-critical computer systems" of the United States Department of Defense (DoD).[1] One aspect of the SEI's work is to tap the technology transfer efforts already existing within the complex of organizations that comprise the DoD and surrounding defense industry. These efforts include education and training, software tools, newsletters, and briefings at all levels. There are also public-access databases, such as the Ada Information Clearinghouse and those present on SIMTEL20; national document distribution centers, such as the National Technical Information Service (NTIS) and Defense Technical Information Clearinghouse (DTIC); and proof-of-concept projects, such as those funded by Defense Advanced Research Projects Agency (DARPA) and the Air Force Electronic Systems Division (ESD).

Synergism is needed to make these efforts more effective. Specifically, there is a need for someone to actively coordinate and combine existing mechanisms to accomplish the challenging task of effecting *rapid* transfer of the technology and processes of an emerging software engineering discipline. In response to this need, the SEI encourages and facilitates the growth of expertise in technology transfer, that is, in the management of technological change within the DoD and the defense industry. One way the SEI does so is to develop resource materials such as guidebooks and courses. Another way, one which has great leverage, is to work through technology receptor groups located within defense organizations. The SEI is helping to establish receptor groups throughout the DoD, and is providing them courses and materials on the management of technological change.

One important form of technology receptor group is the software engineering process group (SEPG), which focuses on software process improvement. Working with managers and engineers from software development organizations, the process group tracks, screens, installs, and evaluates new methods and technology that can improve the software engineering capability of an organization.

This guide was written to support process groups, especially those in corporations and government agencies that have participated in SEI-assisted assessments and/or SEI self-assessment training. In the future, the SEI will prepare other materials (courses, workshops, other guides, and consulting activities) that will help software organizations improve their development process and transfer new technology into practice.

[1]From the "Charter for the Department of Defense Software Engineering Institute," which accompanied the Request for Proposal, dated June 15, 1984 (Section J, Attach. 12; Contract No. F19628-85-C-003).

Software Engineering Process Group Guide

Abstract: Improving the *process* of software systems development and maintenance is the most reliable way to improve product quality. This document offers guidance on how to establish a software engineering process group (SEPG) and related software engineering process improvement functions. The process group works with line organizations to improve process quality by helping to assess current status, plan and implement improvements, and transfer technology to facilitate improvement in practice.

Introduction

This guide was written to help organizations establish and sustain a process group as the focal point of a software engineering process improvement program. The guide emphasizes "what" over "how." It provides criteria by which improvement efforts can be evaluated, such as, "What characterizes a good training program?" and "Who is a good candidate to serve in the process group?" It does not provide a "cookbook" set of instructions, but it does contain supplementary tutorial material and a reference list for those who wish to explore topics in more depth; in some cases, reference material provides procedural "how to" guidance.

The guide is a good starting place for process improvement: it offers a basic introduction to the subject and provides guidance for initiating and sustaining an improvement program in an organization. The guide is as much concerned with the human side of stimulating a higher quality process—that is, the organizational aspects of technological change—as with the technology[2] of improved processes, such as assessment methods and approaches to process definition.

The guide is based on the experience of the authors as well as that of process groups that were interviewed between October 1988 and March 1990, from the following companies:

- AT&T Bell Laboratories
- Boeing Aerospace
- Boeing Computer Services
- Computer Sciences Corporation
- Contel

[2]The term *technology* is used throughout this guide in the broadest sense. For example, this use of the term would include software inspection as a peer review technology for detecting errors in software development work products; computer-aided software engineering (CASE) as a technology providing automated support for performing activities and creating work products in the software development process; and change management as a technology for planning and implementing organizational change.

- Digital Equipment Corporation
- General Dynamics
- GTE
- Hewlett Packard
- IBM
- NASA Goddard Space Flight Center
- Raytheon
- Systems Research and Applications
- TRW
- Westinghouse

Audience. The guide is intended, first and foremost, for members of process groups in software organizations that are at maturity levels 1 and 2.[3] Another important audience for portions of the guide is the community of software developers and their managers who participate in the process of software development and maintenance. Some sections may be useful for higher level management, and others appropriate for staff in functions that may collaborate with the process group, such as training and education, quality assurance, and standards.

Tips for the Reader. The guide is organized into three parts. The first part, Chapters 1-5, includes an overview of software engineering process improvement and the rationale behind process groups. It also discusses activities that are performed in establishing a process group for the first time, including obtaining sponsorship, performing assessments, planning, creating process definitions, and setting up a process database.

The second section, Chapters 6-8, addresses activities that an established process group would perform on an ongoing basis. These include facilitating technical working groups for technology screening and evaluation; coordinating pilot uses of technology; consulting in process improvement; and maintaining a relationship with sponsors to ensure continuing support.

The third section, Chapters 9-10, addresses organizational issues. These include staffing the process group and locating it effectively within the organization structure.

The appendices contain supplementary material. The first appendix (a reprint of [Humphrey88]) describes the SEI software process maturity framework; readers not familiar with this framework should read the appendix before going on to Part I of this guide. Other

[3]Maturity level 1 and 2 organizations are characterized by ad hoc approaches to software development; see Appendix A for more details.

appendices include a discussion of process improvement and the quality movement (by David Card), a brief description of a planning process called a "search conference," tutorial material on technology transfer and technological change in organizations, a discussion of approaches to training and education, action plan templates, and a summary and analysis of data collected from participants at the SEPG Workshop sponsored by the SEI on June 21-22, 1989.

Implicit in this guide is the need for organizational change. Software process improvement occurs through change, and it is the job of the process group to foster that change. The authors thus encourage liberal translation, particularly of vocabulary, so that the principles and guidelines contained here have maximum benefit for the reader's organization.

Process Improvement: What Is It And Who Owns It?

Quality is a key factor today in international business competition. And quality, most people would now agree, is not something added to the product during testing at the end of the development process; it is something everyone owns and is responsible for throughout that process. From Ford's "Quality is Job 1" slogan to the DoD Total Quality Management program [TQM88], industry in the United States seems to be adopting what the Japanese call *kaizen,* meaning [Kenkyusha54] "continuous improvement." Software development organizations are no exception to this: Hewlett Packard measures its progress through a company-wide program in software metrics [Grady87], and both AT&T Bell Laboratories and IBM have devoted entire issues of their technical journals to software quality technology [Myers86, Gershon85].

How do organizations move from their current state to one where there is continuous improvement? First, they must establish an organizational commitment to quality; next, they must create an entity in the organization that is the focal point, some group responsible for facilitating actions supporting that commitment; and finally, they must carefully plan each step to move from the current situation to the desired one. In the software industry, the organizational focal point is a *software engineering process group*, and the model for the step-by-step change is the *process improvement cycle*. In fact, the phrase "software process improvement" is often used as a synonym for "software quality."

The following sections describe the process improvement cycle and the role of the process group and others within a software engineering organization.

The Process Improvement Cycle

Software process improvement is a continuous cycle. The following steps are adapted from the well-known Shewart cycle [Deming86]. See also Figure B-2, page 98.

1. Set expectations.
2. Assess the current practice.
3. Analyze the variance between expectation and practice.

4. Propose changes that will reduce the variance and thereby improve the process.
5. Plan the integration of the improvements into the existing process and update the process definition. If a formal process definition does not exist, it should be documented now.
6. Implement the improvements.
7. Perform the process as it is now defined.
8. Start over.

Each of the steps is discussed below.

1. Set Expectations. Expectations of what process improvement goals are desirable—from both a competitive and technical perspective—may be set in a number of ways. Senior management typically sets its goals in response to business conditions, such as reductions in the DoD budget, offshore competition, and changes in contractual requirements such as software warranties. Managers may receive this information from technology tracking offices or market research functions within their own organization, from colleagues in their own organization or in the larger professional or business community, from business or technical literature, and from consultants. In addition, internal engineering organizations may lobby senior executives for new directions and change; this is often the case in software organizations where senior managers may be unfamiliar with the technology and competition.

Eventually, all this information may lead to significant policy changes. For example, IBM has a policy that every new software release will have fewer defects than the mature release or product it is replacing. Motorola has a policy that to continue to do business with them, current suppliers must apply for the Malcolm Baldridge National Award for Quality by 1992. Actions such as these lead to a shared vision among consumers, producers, and practitioners of what it means to have quality of product and process.

High-level organizational expectations must, of course, be translated to specific quality goals and lower level operational objectives. Appendix A presents general guidelines for objectives that are likely to be appropriate for software organizations at various stages in their evolution.

2. Assess the Current Practice. Assessing the current practice can be accomplished in a number of ways. One example is the SEI software capability evaluation method [TR23], which is now being used as part of source selection and as a government contract management vehicle [Thomas88]. Variants of this method include the SEI-assisted assessment and the SEI self-assessment procedure; organizations can use the latter to assess themselves. Another example is the Software Development Capability/Capacity Review [ASD800-5] of the Air Force Aeronautical Systems Division, also used as part of source selection. (See Chapter 3 for more information about assessing current practice.) Whether conducted by an outside agent or by an organization for their own purposes, assessment procedures such as these give organizations a systematic and thorough way to examine and document the current status of their software development process.

3. Analyze the Variance Between Expectation and Practice. Analyzing variance means determining clearly and specifically what areas fall short of expectations. The assessment results will have identified areas for investigation. For example, given an objective of having all projects under configuration control within a year, a group of software project managers will determine the current situation and report, in detail, exactly where it falls short. If routine cost and schedule estimation is another objective, other managers and engineers might review current obstacles to its use. They might note lack of rewards, tools, or education, for example. Based on the results of an assessment, managers may also refine their expectations and goals.

Organizations with mature processes (maturity levels 4 and 5 in Appendix A) analyze the variance between expectation and practice using the tools of statistical process control [Feigenbaum83, Ishikawa85]. Such practices can simultaneously show the quality goals and the outcome of quality practice. Knowledge of software engineering process and technology, and experience in software development and management lead to the generation of viable alternatives to current practice. Selection can be made by considering these alternatives in the context of analysis of data from actual practice. Thus the process improvement approach leads systematically to reduction of the variance to an acceptable level. Appendix B provides an introduction to the subject and pointers to additional references. Excellent additional sources are [Harrington87] and [Humphrey89], both based on the pioneering work of [Deming86].

4. Propose Improvements. Once areas needing improvement have been identified, an organization must determine what activities are appropriate. Alternatives are explored and decisions made. If a cost-estimation procedure needs to be developed, for example, requirements for it need to be analyzed and carefully specified. The procedure, and possibly tools to support it, can then be obtained or created.

5. Plan the Integration of Improvements. Planning the integration of improvements into existing approaches to work is closely connected with proposing those improvements. In both activities, potential users can provide valuable input, resulting in an effective approach to helping projects use the improvements.

If the organization does not have a formal process definition, this stage of the improvement cycle is the time to create it. If a formal process definition exists, this is the time to update it.

The field of organization development within management science has much to offer those who plan and implement the technological changes required by improvements. A brief tutorial on change management and technology transition is provided in Appendix D.

6. Implement Improvements. The process group helps plan and execute improvements, but the actual changes occur in the line organizations that develop software. These changes then become part of the ongoing practice and should be reflected in an updated process definition or description.

Analyzing variance, proposing improvements, planning the integration of improvements, and implementing improvements are all aspects of the process group’s job of implementing change. One useful model of this process comes from [Freeman87]: each new improvement, along with its planning and installation processes, can be treated as a project. Chapters 6 and 7 discuss this approach in more detail.

7. Perform the Process as Defined. After improvements are in place, they become part of routine operation within software development or maintenance functions.

8. Start Over. Periodically or after several improvements have been completed, the organization should begin the cycle again. Eventually, as the organization becomes more adept at the improvement process, and as it develops a well-measured, well-managed process, continuous improvement itself becomes routine.

Who is a Part of Process Improvement?

Indicating that everyone in an organization "owns" software software process improvement may be appropriate for high-level policy and corporate quality philosophy. However, more specificity is required for making continuous improvement operational. Because the process group works as part of an organization, and must also work with groups and individuals within that organization, it is helpful to look at the specific contributors to improvement efforts.

Although organizations, whether corporations in industry or government labs and agencies, vary widely, they have some essential functional areas and levels in common. Each area or level plays particular roles and performs certain functions with respect to owning quality. Each of several key areas is described below and discussed briefly. Further discussion of the roles of management and engineers in adopting new software engineering technology may be found in [Ebenau83].

Executive management. Executive or senior managers play a major role in setting and directing strategy for continuing process improvement across their organizations. They also provide sponsorship of improvement efforts by taking action in support of the strategy. As efforts to improve the software process begin, they may, for example, establish a high-level steering committee to provide periodic broad review of organization-wide activities. Later, they may review the continuing education policy and budget to ensure sufficient support for process improvement. One senior-level advocate of software process improvement can be a catalyst for moving an entire organization toward significant improvements in software quality.

Middle management. Middle managers often control the disbursement of resources for technical projects. They support process improvement efforts by their willingness to revise existing and planned resource strategies. They may, for example, recast schedules to accommodate the implementation of a new technology; they may allow engineers to work on technical committees to rewrite a process definition for the use of Ada or for code inspections; or they may sit on committees themselves to examine and improve software management practice.

First-line supervisors and engineers. Software engineers contribute to quality by building it into the systems they create and maintain, and by joining together to think about *how* they build systems. Software engineers may work on technical committees that are chartered to track, evaluate, implement, and consult on new technologies; they may serve as technical mentors for new employees; they may build tools; and they may teach courses or hold informal seminars. Most valuably, they may collaborate on new approaches that might lead to improved software processes, and provide crucial feedback on what works, what doesn't, and why.

Staff organizations. Corporate or project/program level quality assurance functions may focus on quality technology as well as quality audits. When this is the case, they may be willing to collaborate in preparing training and education, especially for management, in the continuous improvement philosophy. They may also be able to provide important visibility at a corporate level for process improvement efforts that senior executives, if they are unfamiliar with software, may overlook.

Education and training groups may offer expertise for developing and purchasing courses in support of particular improvement efforts, such as a new approach to technical reviews or a CASE tool set. They may also provide administrative and logistical support by announcing course offerings, registering and tracking students, and supplying and maintaining materials.

Even though everyone owns software quality, each individual or group will implement that ownership in a very specific way. Part of the effort of the process group will be devoted to helping to define and plan these specific activities. The following pages provide information that will help process groups to do their job.

Part I — Starting a Process Group

1. The Process Group

The software engineering process group is the focal point for process improvement. Composed of line practitioners who have varied skills, the group is at the center of the collaborative effort of everyone in the organization who is involved with software engineering process improvement. Group size is usually equal to 1-3% of the development staff. Because the process group is small, it relies upon outside support, in particular, the support of a steering committee and technical working groups.

Most organizations have many different projects or contracts underway simultaneously, and perhaps several different software development or maintenance divisions. Thus, many instances of the process improvement cycle may be operating concurrently. For example, one part of the organization may be following up on assessment findings by examining current cost-estimation practice, and another may be training software engineers in code inspections. The process improvement cycle works both globally across the organization, through policy changes and process assessments, and locally, with the implementation of particular changes. Assessments lead to strategy and policy updates; these updates lead to plans for the incorporation of new approaches to management and new software technology; the results meanwhile must be tracked and reported.

The software engineering process group is a central force for process improvement. The group maintains the overall view of current efforts and facilitates these efforts on a continuing basis. Its members foster collaboration among everyone in the organization who is involved with software process improvement. Following are ongoing activities of the process group:

- Obtains and maintains the support of all levels of management.
- Facilitates software process assessments.
- Works with line managers whose projects are affected by changes in software engineering practice, providing a broad perspective of the improvement effort and helping them set expectations.
- Maintains collaborative working relationships with software engineers, especially to obtain, plan for, and install new practices and technologies.
- Arranges for any training or continuing education related to process improvements.
- Tracks, monitors, and reports on the status of particular improvement efforts.
- Facilitates the creation and maintenance of process definitions, in collaboration with managers and engineering staff.
- Maintains a process database.

- Provides process consultation to development projects and management.

The process group is not part of product development but is staffed by practitioners. As a result, it has expertise in software engineering. It may also have, and at any rate should develop, expertise in process definition, organizational change, and technology related to improving or measuring quality. The process groups that were interviewed in preparation for this guide were predominantly staffed by engineers, highly motivated and generally very experienced people, many of whom had already worked for years as individual champions to improve the software engineering process in their organizations. Staffing the process group with engineers who have customarily worked on building systems reassures the organization at large that the work of the process group will be practical and relevant.

One way to view the process group is as a permanent task force. If software process improvement were a one-time occurrence, then a task force—most likely of engineers and engineering managers—would be created. Because this effort is ongoing, the group that supports it must also be ongoing. The challenge of the process group and its managers and sponsors is to maintain the enthusiasm and vitality of a task force on a continuing basis.

1.1. Costs and Benefits

SEI interviews with existing process groups indicated that few, if any, process groups had to present a persuasive business case to become chartered and operational. Software engineering process improvement was a concept with inherent merit in their organizations: competitive pressures required a steady increase in quality, which in turn required a commitment to improve. Nonetheless, it should be helpful to look briefly at specific costs and benefits of process groups and related efforts.

1.1.1. Costs

The costs of establishing the software engineering process group and related functions—the process improvement infrastructure—are primarily labor costs. These are consumed in staff for the software engineering process group, and in the time spent in planning and implementing improvement actions. Often these are costs that would be expended on similar activities in any case, but in a less organized and therefore less visible way.

In organizations of a hundred or more software professionals, the recommended budget for a process group [Humphrey89] is normally 1% to 3% of the overall software development budget of the organization it supports.[4] Appendix G contains data gathered informally at the SEPG Workshop held by the SEI in 1989. The percentage of overall budget spent by workshop participants on process groups varied widely, and was generally much lower than 1%.

[4]Contel spends about 6%. General Motors spends about 1/2 of 1% on a $3.8 billion internal budget for technology exploration [Trainor89], which is only a portion of the process group’s responsibilities.

There are different approaches to budgeting funds for process groups. Some organizations use research funds, and others include the costs in overhead. Several process groups surveyed charge all their costs (direct, indirect, research, and development) back to users.

1.1.2. Benefits

The primary benefit of an improved—that is, more disciplined—software process is improved visibility of the process. This visibility makes the process more manageable during software development and maintenance, thus reducing risk. Schedules become more predictable, as do costs. Software is of higher quality at delivery and is easier to maintain. The cost of investment in process improvement work is amortized and ultimately more than repaid, as is illustrated in Figure 1-1.

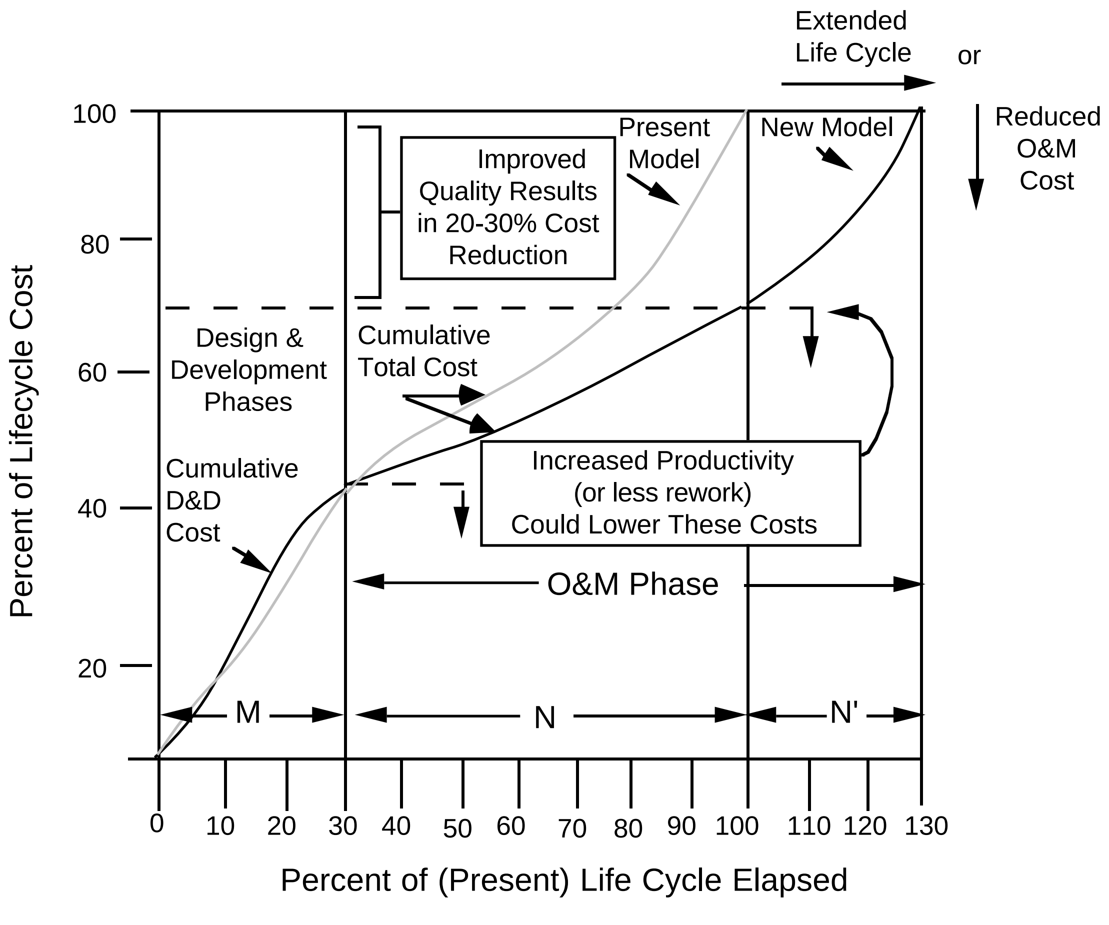


Figure 1-1: Benefits of a Mature Software Engineering Process

Reprinted from [Jacobson89].

Another benefit is the improved prospect of winning and keeping contracts when the government includes software process maturity in its source selection and contract management criteria [Thomas88]. Moreover, if certain contract incentives are inaugurated, developers would be paid an award fee of up to 15% based on software quality as assessed during the early post-deployment phase [Jacobson89]. A mature development process would make it more likely that the award fee would be earned.

An improved process also allows easier acquisition and adoption of new technology because that technology can be acquired in direct support of defined processes. The process definition necessary to a disciplined software process is also prerequisite to reasoned analysis about what software tools and methods best support the goals and the creation of products and systems within the organization.

1.2. Organizing for Process Improvement: The Collaborators

Process improvement is a long-term effort. The working relationships between the process group and the rest of the organization must be ongoing, and management oversight and direction must continue over the long term. This section describes some organizational approaches that help keep efforts to change visible in the software engineering organization. Because organizational terminology varies widely, readers are encouraged to consider the spirit of each of the entities described and to implement their functional equivalents in a suitable way.

1.2.1. The Steering Committee

One means by which line and supervisory management guide process improvement is through membership on a steering committee that meets periodically (perhaps monthly), translates related corporate policy, and sets priorities. The committee reviews the results of assessments, charters technical area working groups to prepare plans, approves those plans in priority order, monitors the progress of the working groups, and helps obtain resources for the process group. In this way, the steering committee and the manager to whom the process group reports provide important and highly visible support for process improvement. Support should also be cultivated at any other management levels that influence or are affected by the process improvement work.

The ongoing work of a standing steering committee includes policy oversight, resource and process management, integration and consensus building among groups having different perspectives, and serving as liaison to higher level steering groups or corporate planning organizations. In two of the companies interviewed, the steering committee that oversees the process group reports to a higher level steering group that has broader concerns. Software process improvement issues are thus connected to product quality issues on a corporate level.

The steering committee provides additional organizational benefits. Kanter [Kanter83] discusses at length the need to foster open communication and "integrative approaches" that bridge interest groups. Open communication is important because it gives managers "crosscutting access" to information, resources, and support outside their immediate domain. Kanter refers to "innovating" organizations, meaning organizations that foster change, both technological and non-technological, that can lead to greater productivity. In such organizations, Kanter claims there is benefit in

> ... drawing members from a diversity of sources, a variety of areas.... It is not the ‘caution of committees’ that is sought—reducing risk by spreading responsibility—but the better idea that comes from a clash and an integration of perspectives. (p. 167)

Often software managers have no opportunity to work with their peers except in steering committees; these committees provide a valuable opportunity to compare notes, share ideas, sharpen thinking in a context of "we’re all in this together." The Software Management Steering Committee at Westinghouse was created from an older and more informal committee of middle-level software managers. This group has now worked together for nearly ten years. Members confirm that it has required some effort to learn how to reach consensus, but that it is worth the investment to be able to rely on each other for honest and constructive criticism offered from many points of view.

1.2.2. Technical Working Groups

Humphrey recommends that organizations "...aim at full-time assignments to the SEPG of about 2 percent of the software professionals" [Humphrey89]. This level of staffing was rarely found either in the organizations interviewed or in those who sent participants to the SEI 1989 SEPG Workshop (see Appendix G). Moreover, process improvement is such a large task that process groups need to augment their permanent membership. Groups usually do this by taking advantage of the experience and expertise of the population to be served.

Standing working groups of engineers can accomplish a great deal on a part-time basis. These groups are chartered to work on a wide range of issues from selecting new design methods to choosing the measures of software productivity and quality to be included in the process database.

The members of working groups, engineers from a number of project types and application domains, benefit for the same reason their managers benefit from having a steering committee, that is, cross-fertilization. And since working groups are typically staffed by senior people, they supply ready consulting expertise. Members are also a natural choice to serve as representatives to professional and technical organizations such as AdaJUG and the IEEE Computer Society.

2. Assessments

Assessment often provides the stimulus for provess improvement. Several methods and questionnaires are available that result in a written set of findings and recommendations. High-level sponsorship, confidentiality, and collaboration are the principles that guide successful assessments.

Software process improvements are often initiated as a result of an assessment of the current situation. The impetus for beginning the improvement cycle may be a general awareness of the need to improve software productivity and quality. It may also result from unsuccessful attempts to incorporate CASE tools, or from a review of increasing numbers of customer complaints. While both management and practitioners may agree that improvement is needed, the organization must have a clear set of priorities *before* planning begins. The results of an assessment provide a basis for planning and, later, for process group work, as well as a benchmark against which process improvement efforts can be measured.

One of the forces prompting software process assessment is the DoD initiative to select software contractors based upon the quality of their development process as revealed by such an assessment (see, for example, [Thomas88]).

Formal assessment procedures based on questionnaires and interviews include the SEI Software Capability Evaluation [TR23], the Air Force Software Development Capability/Capacity Review [ASD800-5], and the approach described in [Pressman88]. All three approaches call for a written report that describes strengths and weaknesses and includes recommendations for addressing weak areas. This information is input to an action plan for overall, long-term process improvement. The assessments typically result in an increased awareness of the quality and characteristics of an organization's current software process, including procedures, technologies, and the capability to use them.

In addition to formal assessments, there are other ways to identify areas needing improvement. These include special meetings such as search conferences (see Appendix C), focus groups, and interviews or surveys of technical staff and management. Prior records of consensus on areas needing work may also be tapped. For example, many organizations routinely reassess themselves as part of strategic planning or project postmortems, and it is useful to review findings from these activities. Some life-cycle definitions include a lessons learned activity [SMAP4.3, Gale90, Mays90]. These techniques usually address specific areas of concern; and because they may be less structured or require less time, they can be used more frequently. Thus, they complement formal assessments.

A thorough, formal assessment is the best way to initiate the development of organization-wide strategies and broad changes that will lead to software process improvement. One reason for this is that an assessment offers a forum for presenting concerns and ideas, particularly in the case of practitioners. The high-level sponsorship typically obtained for an assessment assures participants that follow-up action will be taken. This assurance of results generates enthusiasm for participation not only in the assessment activities, but also in subsequent planning and improvement projects.

An initial assessment can be a prelude to the establishment of the process group or can be the first task of a newly formed process group. A follow-up assessment in 18 to 24 months provides an opportunity to review progress.

2.1. Example of an Assessment Method

This section summarizes the SEI self-assessment approach, which is described in more detail in [TR7]. The approach uses a questionnaire developed at the SEI (see [TR23]) and based on the framework described in Appendix A as well as [Crosby79]. Figures 2-1 and 2-2 present the SEI process in some detail. Figure 2-3 shows sample questions. For comparison, Figure 2-4 lists some questions from [Pressman88], which describes a different approach along with related process improvement actions.

In using SEI methods, it is common to assess about a half dozen projects and then analyze the responses to the assessment questionnaire across the projects, across levels (management and practitioner), and across types of organizations (projects as well as functional areas, such as software quality control, standards, and independent testing). The cross-section views that result are often very fruitful in identifying issues. For example, if half the assessed projects use software configuration management (SCM) and the other half do not, clearly there is no *organization-wide practice* of SCM, even though there may be a policy, standard, training course, and SQA oversight. SCM then becomes an area requiring further investigation by the assessment team during the interview sessions.

Interviews are conducted after the questionnaire is completed and responses analyzed by the assessment team. The interviews consist of discussions among assessment team members and project personnel and functional area specialists to elicit concerns and ideas for improvement. These discussions typically end with an open-ended question such as: "If you could improve just one facet of the process, what do you think would have the greatest leverage, and why?" With most groups, this question elicits numerous creative ideas.

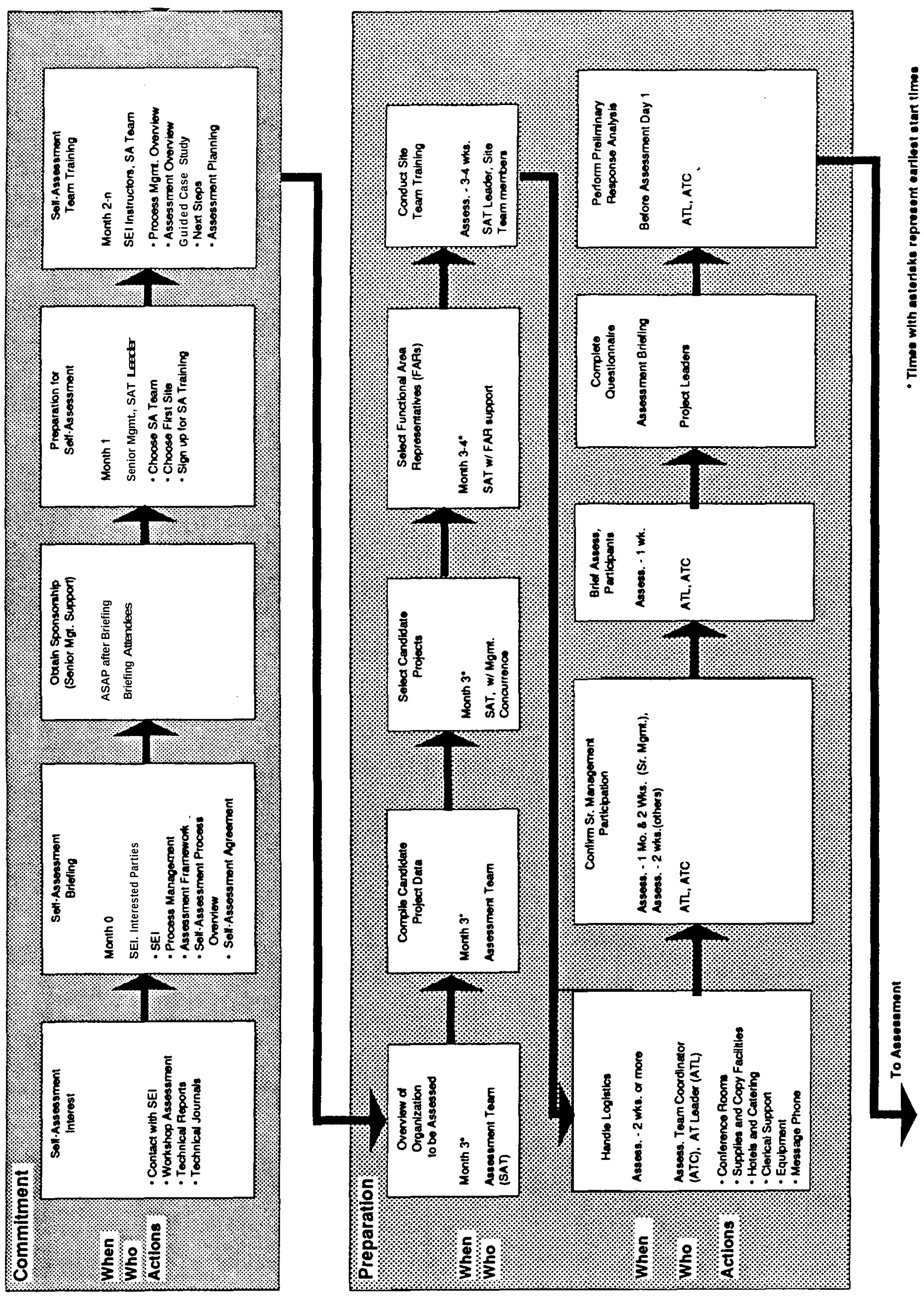


Figure 2-1: Self-Assessment Map - 1

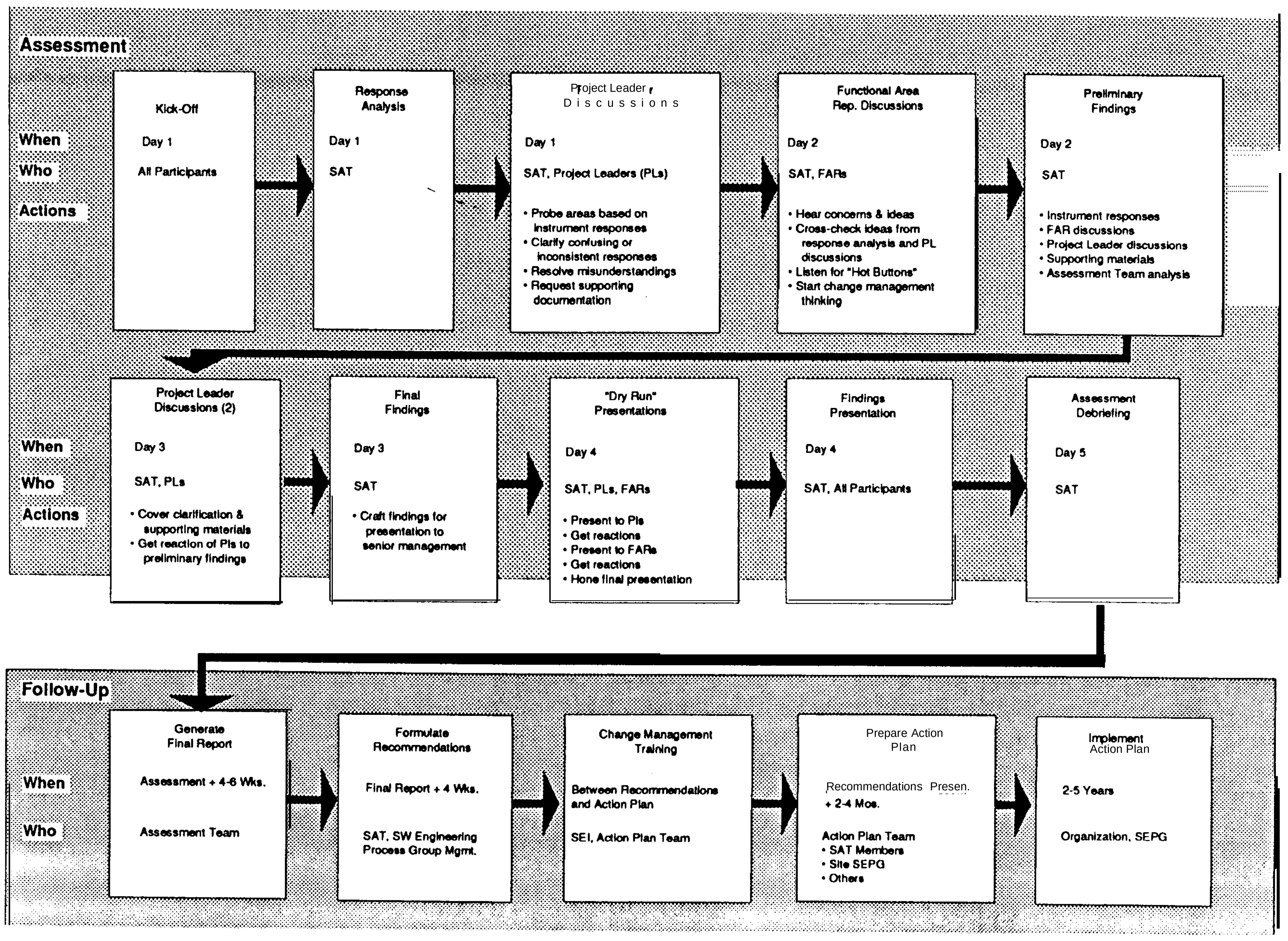

Assessment
When
Who
Actions
Kick-Off
Day 1
All Participants
Response Analysis
Day 1
SAT
Project Leader Discussions
Day 1
SAT, Project Leaders (PLs)
• Probe areas based on instrument responses
• Clarify confusing or inconsistent responses
• Resolve misunderstandings
• Request supporting documentation
Functional Area Rep. Discussions
Day 2
SAT, FARs
• Hear concerns & ideas
• Cross-check ideas from response analysis and PL discussions
• Listen for "Hot Buttons"
• Start change management thinking
Preliminary Findings
Day 2
SAT
• Instrument responses
• FAR discussions
• Project Leader discussions
• Supporting materials
• Assessment Team analysis
When
Who
Actions
Project Leader Discussions (2)
Day 3
SAT, PLs
• Cover clarification & supporting materials
• Get reaction of PIs to preliminary findings
Final Findings
Day 3
SAT
• Craft findings for presentation to senior management
"Dry Run" Presentations
Day 4
SAT, PLs, FARs
• Present to PIs
• Get reactions
• Present to FARs
• Get reactions
• Hone final presentation
Findings Presentation
Day 4
SAT, All Participants
Assessment Debriefing
Day 5
SAT
Follow-Up
When
Who
Generate Final Report
Assessment + 4-6 Wks.
Assessment Team
Formulate Recommendations
Final Report + 4 Wks.
SAT, SW Engineering Process Group Mgmt.
Change Management Training
Between Recommendations and Action Plan
SEI, Action Plan Team
Prepare Action Plan
Recommendations Presen. + 2-4 Mos.
Action Plan Team
• SAT Members
• Site SEPG
• Others
Implement Action Plan
2-5 Years
Organization, SEPG

Figure 2-2: Self-Assessment Map - 2

1.1.1. For each project involving software development, is there a designated software manager?

1.1.6. Is there a software configuration control function for each project that involves software development?

1.3.4. Is a mechanism used for managing and supporting the introduction of new technologies?

2.1.1. Does the software organization use a standardized and documented software development process on each project?

2.1.12. Are internal design review standards applied?

2.1.17. Is a mechanism used for ensuring that the software design teams understand each software requirement?

2.2.3. Are statistics on software design errors gathered?

2.3.7. Is a mechanism used for initiating error prevention actions?

2.4.8. Is a mechanism used for controlling changes to the software requirements?

Figure 2-3: Samples from the SEI Software Capability Evaluation Questionnaire [TR23]
All questions are to be answered Yes or No.

2.2. Principles

Certain basic principles apply to assessments, regardless of the specific approach; they are discussed briefly here.

Sponsorship. Assessments must have a sponsor—the higher the sponsor's position in the organization, the better. The sponsor selects the assessment team, provides authorization for the team to meet with staff and conduct the assessment, and promises in advance to respond to the findings and recommendations. By being personally involved, the sponsor communicates to the organization that the assessment, a first step in process improvement, is critical and will shape the priorities.

Confidentiality. It is vital to solicit the opinions of the practitioners "in the trenches" (including project leaders) because they are in the best position to know the status of the present practices. However, developers familiar with possible shortcomings of the existing process

```
8.1.    List by priority five areas of software project planning,
        development, tracking, and control that need improvement.

9.1.    List by priority five areas of software project specifica-
        tion, development (design, coding) testing, maintenance that
        need improvement.

12.1.   Describe any project management methodology that is actively
        applied in your group. How long would it take to determine
        the exact status of a project (in minutes)?

12.7.   Do you have a formal software configuration management ap-
        proach?

12.8.   Do you have a formal metrics collection program in place?

13.1.   Do you conduct reviews of computer software as it is being
        developed?

Provide data (if such data can be easily collected) for each
        software development application area (e.g. MIS, engineer-
        ing) separately:

18.1.   Delivered source lines per person per year (average) for all
        projects (include all specification, design, test work)?

18.7.   Number of days of training per software manager and prac-
        titioner per year?  Is this number mandated?
```

Figure 2-4: Sample questions from [Pressman88]

would be reluctant to discuss them if there were a chance their comments could affect them in an adverse way. To get an accurate picture of the software engineering process, therefore, the assessment team must ensure confidentiality. The team must be careful in aggregating data (especially with small sample sizes) to present composite findings that protect the identity of the contributors. For the same reason, no one who is involved in reviewing, supporting, or managing any projects that are being assessed should participate on the assessment team.

Collaboration. The assessment process is based upon the belief that the local software practitioners are the best sources of information on the current practice and that they have the greatest interest in its improvement. The assessment itself demonstrates the value of collaboration across levels and functional units. The assessment process, therefore, represents an opportunity to show practitioners that their ideas are important and that their support is prerequisite to team success.

2.3. Phases of an Assessment

Assessments are typically conducted in four phases:

1. The sponsor and, if possible, a representative cross-section of the sponsor's organization make a commitment to the full assessment process. This commitment includes a promise of personal participation and agreement to follow up on recommended actions.
2. Preparation is made for the on-site assessment. An assessment team is selected and trained, and an assessment schedule is approved.
3. The assessment is conducted. Practitioners who will be answering questions and participating in the assessment receive introductory presentations. The assessment questionnaire is answered, and the assessment team analyzes the resulting information. Finally, the team presents preliminary findings to assessment participants and senior management as a "sanity check."
4. The assessment team prepares a written report of findings and recommendations, and the report is formally presented to the organization.

The assessment is just the beginning of software engineering process improvement; it is similar in many ways to the requirements analysis process. After the assessment, the organization needs a plan of action for implementing the recommendations. Action planning is discussed in the next chapter.

3. Action Plan

The action plan is a formal, written response to the assessment, and the "map" for improvement. It is actually a set of plans—high-level strategic plans, tactical plans for each technical area, and operational plans.

The action plan is the heart of software engineering process improvement. The action plan grows out of a broad collaboration of all parties involved in improvement. It documents the following:

- Overall strategy for the improvement effort.
- Major areas that will be addressed, and general guidelines on how to improve in other areas.
- Underlying motivation for the process improvement effort.
- Procedures for planning improvement tasks for specific projects in each major area of improvement.
- Groups responsible for taking action, the resources needed, and the schedule.

This chapter discusses the components of the action plan, the planning process, and the ongoing responsibility for maintaining and updating the plan. The discussion assumes that there has been an assessment; a brief section at the end of the chapter addresses action planning when this is not the case.

3.1. Structure

The action plan is a *set* of plans and guidelines. Together these plans and guidelines address the fourth, fifth, and sixth steps in the process improvement cycle: propose improvements, plan the integration of the improvements and update affected process definitions, and implement the improvements.

Figure 3-1 illustrates the action plan structure. The strategic plan describes the overall motivation, direction, and vision. The tactical plans focus on broad functions (such as the process group) that benefit the overall organization and on key technical areas in which work will support the strategy. The tactical plans include templates and guidelines to help individual projects adopt a particular technology or procedure smoothly. The improvement activity plans,[5] that is, operational plans, instantiate the guidelines for each project. If this structure is adhered to, each improvement activity will be fixed within the context of the overall improvement effort. Appendix F provides annotated outlines of all three plan types.

[5]The terms *improvement activity plans* and *operational plans* are used interchangeably in this chapter and in Appendix F. Technically, all levels of the action plan are improvement plans, but the term *improvement activity plans* is used to convey a specific instance of the overall improvement activity; for example, an effort to create configuration management procedure for all projects.

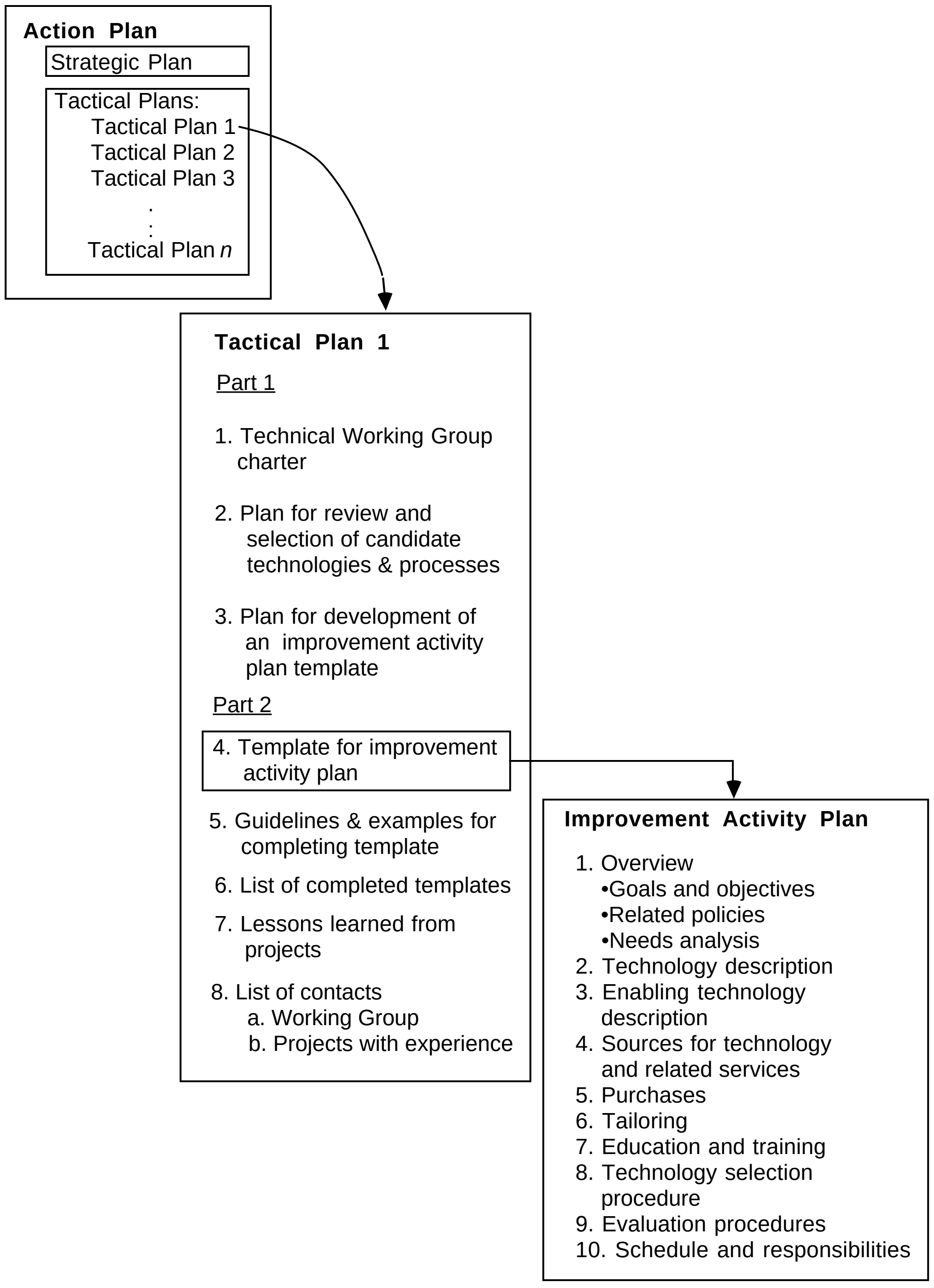

Action Plan
Strategic Plan
Tactical Plans:
Tactical Plan 1
Tactical Plan 2
Tactical Plan 3
.
:
Tactical Plan *n*
Tactical Plan 1
Part 1
1. Technical Working Group charter
2. Plan for review and selection of candidate technologies & processes
3. Plan for development of an improvement activity plan template
Part 2
4. Template for improvement activity plan
5. Guidelines & examples for completing template
6. List of completed templates
7. Lessons learned from projects
8. List of contacts
a. Working Group
b. Projects with experience
Improvement Activity Plan
1. Overview
•Goals and objectives
•Related policies
•Needs analysis
2. Technology description
3. Enabling technology description
4. Sources for technology and related services
5. Purchases
6. Tailoring
7. Education and training
8. Technology selection procedure
9. Evaluation procedures
10. Schedule and responsibilities

Figure 3-1: Action Plan Structure

The strategic plan is based on the findings from the assessment. It describes the motivation and direction for addressing those findings within an improvement program. The plan might begin with a statement such as this:

> We plan to increase our organization's software process capability to Level 3 in the maturity framework by the end of fiscal year 199x. We intend to accomplish this incrementally, with an average of two specific major improvement efforts per project per fiscal year. We will spend approximately 10% of our overhead budget on this effort. We are making these improvements to strengthen our position in a marketplace that increasingly values quality.

The **tactical plans** include charters for the following:

- Software engineering process group.
- Management steering committee.
- A technical working group for each technical area necessary to meet the strategic goal.

The plans also describe the direction of work in each area and would be produced by committee or group representatives. If groups already exist whose domain is similar to the suggested working groups or committees, those groups should be incorporated into the action plan and process improvement framework.[6]

The remainder of the tactical section of the plan contains two major parts, which should be developed and delivered in sequence:

1. The plan for addressing a particular technical area. Information includes how to study the alternative technologies and make selections, and how to consider the impact of that technology (e.g., the need for training and workstations).
2. A template for planning the implementation of those decisions on a per project basis, that is, a template for the operational plans.

These sections serve as guidelines for creating operational plans for specific improvement actions in projects. For example, if the first portion of the tactical plan charters a technical working group on Ada, then the guidelines for producing the operational plan might offer an Ada-specific planning template, developed by the working group, that would address how to incorporate the use of Ada at the beginning of a new contract or project. The working group might also include reference material and a checklist of activities that projects that have already adopted Ada have found useful, such as particular types of training sessions and compiler selection procedures. Finally, the guidelines should include an example of a completed template, prepared by a project with experience.

[6]A good example of an action plan containing working group charters and an excellent discussion of overall process improvement structure can be obtained by contacting Ms. V. Mosley, Westinghouse Electronic Systems Group, M/S 5370, PO Box 1693, Baltimore, MD 21203; phone (301) 765-5560.

Operational plans are developed for a particular project working on improvement in a particular area. For example, a working group on technical reviews develops a general plan describing the type of reviews used in an organization and guidelines on best practice. A particular project then tailors the general plan, adding details such as the name of the reviews coordinator, conference rooms set aside on a standing basis for holding reviews, the name of the secretary responsible for organizing review meetings and providing duplication service, and a set of checklists for reviews of various types common to that project. A member of the working group should assist project members in tailoring the plans, and should consult or "trouble shoot," especially when the new procedures are first used.

3.2. Development

The process of developing an action plan should build on the momentum begun with the assessment and its findings and recommendations, and broaden the scope of involvement by members of the organization at as many levels as possible. One former SEI resident affiliate, Kevin Kendryna from the Naval Undersea Warfare Engineering Station, has noted that the action plan is only as good as the action planning process.

The objectives of the process are to do the following:

- Enlist the support of those who will be affected by the improvement effort.
- Collect the best ideas about improvement and put them in writing.
- Develop a prioritized list of tasks that are feasible and on which all levels of management and practitioners agree.
- Clearly articulate the motivation for making the proposed improvement.

There are a number of approaches to developing an action plan that will meet these objectives. They range from corporate strategic planning efforts to broadly inclusive search conferences (the subject of Appendix C). Whatever the approach, it should build upon assessment findings and aim at developing, as specifically as possible, a vision of the improved organization and the motivation to drive the change. The approach should be selected based on the best prospects for its effectiveness in a particular setting.

Both the search conference and the more conventional strategic planning approaches have advantages, depending on the corporate culture. Corporate strategic planning is often the norm in a strongly centralized organization, while search conferences are more likely to succeed in a highly participative enterprise. Centralized strategic planning draws on the wisdom and breadth of perspective of upper managers but may not result in depth of information or buy-in at lower levels. The multi-level, participative design of a search conference draws on a rich range of organizational experience, but senior managers may not have confidence in the results, and even lower level managers may feel a loss of power and influence.

Another approach is to assemble a working group that surveys participants in the improvement process, drafts the action plan, conducts reviews, and obtains buy-in at as many levels as possible. An advantage of this approach is that it is relatively less work than a search conference yet still taps a broad population. A disadvantage is that there is no direct communication between organization levels or between people holding different viewpoints. Because enthusiasm for the process is largely contained within the working group, as is integration of perspectives, working groups may be more viable in the role of implementors of action plans.

Figure 3-2 illustrates four possible sequences of activities for creating action plans.

It is most important to sustain broad enthusiasm for software process improvement. This is best done by enlisting support during the planning phase, creating a planning document that effectively communicates the process improvement vision and the motivation for improvements. To prevent those involved from getting "bogged down" in generating a stack of detailed plans, the action plan is hierarchical and modular.

A month or so after the assessment results have been presented, the strategic action plan should be released to all those affected by the plan. The challenge is then to encourage a high level of interest among them, while the working groups develop more detailed tactical plans. Maintaining interest is best accomplished through broad participation in the planning process; it is critical to keep *everyone* informed. There should be activities that supplement the planning process and that are broadly accessible; major events should occur about every six months. Examples include:

- A twice yearly review of the improvement plan and its status by a board of outside advisors who report to the steering committee.
- Open status review sessions held by working groups.
- An annual symposium that includes reports on the progress of the process improvement projects.
- Reviews of related technology that was screened or applied successfully, and informal working sessions for giving feedback to technical working groups.

There is a considerable technology of action planning, so elaboration is beyond the scope of this guide. The interested reader is encouraged to start with (listed alphabetically) [Lorange84], [Pinto88], [Quinn77], [Schultz87], and [Tichy83].

3.3. Ownership

The action plan represents a set of commitments to software process improvement, and spells out the responsibilities of each party involved. The process of creating the plan should include as many people as possible. Once the plan has been created, however, each piece must have one owner. This does not eliminate further involvement by the rest of the organization; it does mean clear-cut assignment of responsibilities. These responsibilities include updating and revising each part of the plan and reporting status of both the improvement actions and the plan itself.

Key Off Exisiting Strategic Planning Cycle

1. Conduct an assessment.
2. Integrate the assessment results into the existing strategic planning cycle.
3. Conduct a meeting to transfer the assessment and planning results to an audience of implementors, yielding a tactical action plan.
4. Use the tactical action plan to derive operational action plans on a per-project basis.

Create the Strategic Plan at a Search Conference

1. Conduct an assessment.
2. Convene a search conference in which the assessment results are available as expert evidence of concerns and solutions, yielding a strategic action plan.
3. Hand results to working groups as input for creating a tactical plan.
4. Create operational plans on a per-project basis guided by the tactical plan.

Create the Strategic Plan in the Steering Committee

1. Conduct an assessment.
2. Convene a software process improvement steering committee to recommend a strategic action plan to the sponsor and obtain the sponsor's approval.
3. Establish a process group to use the strategic plan to create a tactical plan.
4. Convene a working conference where managers and practitioners review, revise, and buy in to the strategic and tactical plans.
5. Create operational plans based on guidelines in the tactical plans.

Throw One Away

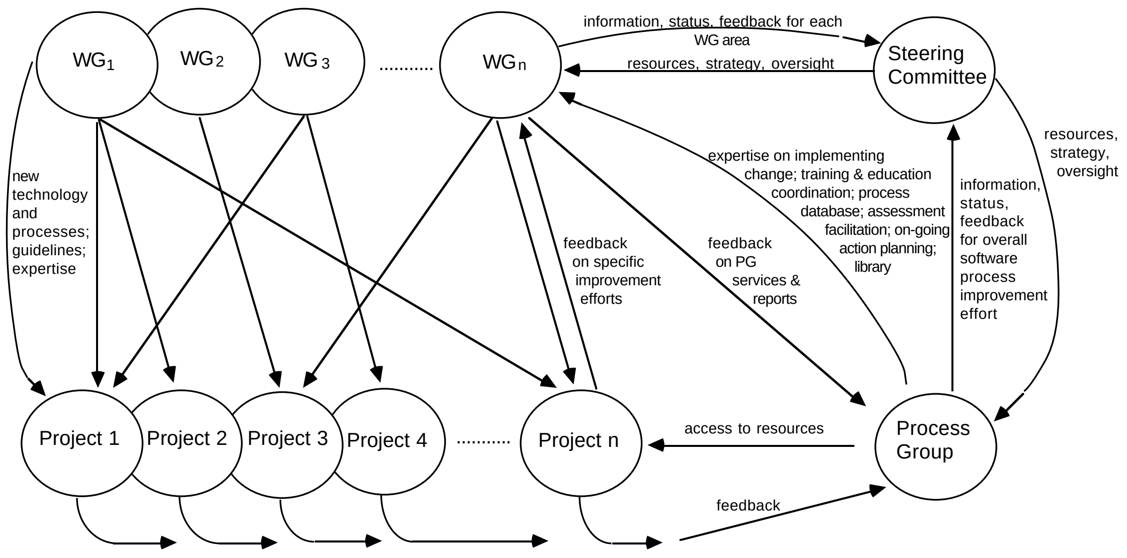
1. Conduct an assessment.
2. Develop strategic, tactical, and operational plans by any means.
3. Begin implementation of a pilot process improvement on a project.
4. Conduct a review of that implementation by the developers of the original plans, the current pilot project staff, and the staff of the next project slated for pilot implementation.
5. Use the findings to revise all parts of the action plans; repeat steps 3 and 4 until there are no further major changes to the action plan.

Figure 3-2: Sequence of Activities for Generating Action Plans

The action plan, thus, is not complete until it describes all the actions to be taken, names the groups or individuals responsible, and includes the schedules for the actions. Figure 3-3 lists the action plan "owners" and their responsibilities. Figure 3-4 diagrams the relationship of these groups to each other. In the actual plan for an organization, the names of individuals as well as groups should be listed and kept up to date.

	Action Plan Parts			
Action Planners	Strategic Plan	Tactical Plans	Operational Plans	**On-going Responsibilities**
Executive Sponsor	X			Policy, funding
Steering Committee	X	X		Strategy, oversight, provide reports to sponsor
Process Group	X	X	X	Maintain action plan, provide reports to all involved, orchestrate process improvement process
Working Groups		X	X	Track, screen, acquire, evaluate new technology & processes; develop guidelines for their use; consult on their use
Projects			X	Pilot new technology or processes; provide feedback to working group, process group

Figure 3-3: Action Plan Owners



WG1
WG2
WG3
....
WGn
information, status, feedback for each WG area
resources, strategy, oversight
Steering Committee
resources, strategy, oversight
new technology and processes; guidelines; expertise
expertise on implementing change; training & education coordination; process database; assessment facilitation; on-going action planning; library
information, status, feedback for overall software process improvement effort
feedback on specific improvement efforts
feedback on PG services & reports
Project 1
Project 2
Project 3
Project 4
....
Project n
access to resources
Process Group
feedback

Figure 3-4: Organizational Architecture for Process Improvement

3.4. Action Plans With Multiple Assessments

It is common for an organization to conduct many assessments. Each assessment can yield its own action plan. These plans can conceivably be disjointed and even contradictory unless there is a unifying influence applied.

It is the job of the corporate (i.e., highest level) process group to track and harmonize the action plans that result from multiple assessments, and to resolve contradictions. This effort may, in fact, provide opportunities for economies of scale and cross-fertilization of experience. The process group must also keep sponsoring management informed of the total cost of improvement because the combined cost of all the action plans together may exceed management's expectations.

3.5. Action Plans Without Assessment

If an assessment is not possible, perhaps because it lacks high-level sponsorship, the approach to creating an action plan will need to be modified. The advantage of beginning with an assessment is that priorities are derived from an organization-wide look at issues. The risk of undertaking improvement in the absence of this perspective is that decisions are based on limited information. Planning will, in any case, have its benefits even if it must, of necessity, involve a smaller piece of the organization.

There are two approaches to creating action plans without an assessment. The first is to perform, instead of a full-blown formal assessment as discussed in Chapter 2, a "quick and dirty" evaluation of what works and what doesn't in the organization, using brainstorming or some other group technique. The resulting action plan focuses on a few of the most critical issues, and the time frame for action should be smaller—perhaps a few months instead of two or three years. Technical working groups would be formed to address each critical issue and to create tactical plans. Specific improvement activity plans (that is, operational plans) would be created for projects agreeing to test new approaches or technologies.

The second approach is to begin at the tactical level with a working group addressing one or two issues and creating a tactical plan. This plan could be used as the basis for an improvement activity pilot. The results from the pilot could be used to refine the tactical plan further and/or to create leverage for establishing additional working groups.

4. Describing and Defining the Software Process

Existing processes must be described before they can be well understood, managed, and improved. An organization must then define what these processes should be before attempting to support them with software engineering tools and methods; in fact, the definition forms the requirements for tool and method support. A variety of process description and definition technologies is available.

A process that is not well understood and articulated cannot be managed or improved. Just as manufacturers must select activities and tools for a particular product line, software organizations must also define and implement appropriate processes for each major development effort. Software, by its nature, is very easy to change. Because software practitioners are well-educated professionals who readily take independent action, a well-articulated and understood process is essential to the orderly and predictable operation of a software development organization.

A well-defined and articulated description is prerequisite to process improvement. As Card and Berg [Card89] state, "periodic acquisition of improved methods and tools, by itself, does not ensure continual improvement. To be effective, technology must be integrated into an underlying process. That integration must be managed explicitly." Increases in quality and productivity, along with the successful use of technology, depend directly on a well-articulated and well-understood process.

As an organization's process becomes mature, the process definition is increasingly critical because it is used to determine measures for each step in the process, and it is these measures that will be used to manage and improve the process. A defined process, then, yields the opportunity to employ a quantitative approach to management. Japanese and United States manufacturing firms have demonstrated that the quality of a product directly depends on the quality of the process that produced it.[7] In Appendix A, Humphrey describes relative levels of process sophistication and manageability; attendant effects on productivity and quality can be expected at each level.

4.1. Describing the Existing Process

Building a process definition for an organization begins with describing what exists. Eventually, the definition of what *should* exist is also created, and replaces the initial description.[8]

All software development organizations have a process in place for doing their work. This

[7]For details see Appendix B.

[8]Recommended reading for those beginning the task of describing or defining software process: [Humphrey89], Ch. 13, "Defining the Software Process."

process is not always documented in adequate detail. Corporate or national standards that apply may be too abstract for use without further modification; for example, they may require, as in the case of [IEEE-SQ86], several different types of reviews, but not provide checklists or agendas. A first step in process improvement, therefore, is to prepare a detailed written description of the process as it currently is practiced. The process group and the working groups first describe the existing processes and later define the desired processes, tune them as needed, and maintain the descriptions.

Any of a number of different methods may be used; one of the simplest, ETVX [Radice85], is described below. Another straightforward method with tool support is described in [Kellner89]. Whatever the method, describing the existing process should result in documentation of how software products are actually developed and maintained in a given organization.

4.1.1. Documenting the Process: One Approach

One very accessible approach to writing a process description is presented in [Radice85]. Radice considers a process to be made up of software *activities,* which must be defined. A defined activity is one which has:

> (1) a list of entry criteria that should be satisfied before beginning the tasks, (2) a set of task descriptions that indicate what is to be accomplished, (3) a validation procedure to verify the quality of the work items produced by the tasks, and (4) a checklist of exit criteria that should be satisfied before the activity is viewed as complete. (p. 83)

Radice calls this the E (Entry Criteria) T (Tasks) V (Validations) X (Exit Criteria) process architecture. Using ETVX to describe each phase of software activity should result in a consistent description of software development processes. Gaps—missing activities or a missing E, T, V, or X for an activity—that appear in the process description indicate where immediate process improvement may be needed. For example, the exit criteria for module design may not have been defined or may not match the entry criteria for the next activity.

On the most abstract level, process definition is simply the determination of what stages and activities will be used by a given software organization. [Radice85] lists the following phases for a commercial product:

- Requirements and planning
- Product-level design
- Component-level design
- Module-level design
- Code
- Unit test
- Functional verification test

- Product verification test
- System verification test
- Package and release
- Early support program
- General availability

E, T, V, and X must be defined for each activity. Organization-wide process definitions provide general definitions, noting where individual programs and projects must provide specifics. For example, the general definition of a component design activity might have as entry criteria the availability of an inspected specification document; as a task, the requirement to produce a design document; as validation, a design document inspection; and as exit criteria, the baseline and configuration control of the inspected document. The details of how inspections are conducted, including who and how many individuals should attend, the roles of the attendees, how much is inspected at a time, etc., are left to individual projects. The particular design method and notation, as well as the document creation and maintenance tool, are also left to the individual project.

Standards, templates, and examples are simple and worthwhile complements to the ETVX documentation. *Standards* describe required procedures, practices, methods, and technologies, such as the use of software inspections as peer reviews or a particular Ada compiler. *Templates* are detailed outlines for activity work products (Radice's "work items"). Templates for component-level designs and for test plans and scripts, for example, encourage consistency of format. Templates can be used to enforce standard reporting formats, with some sections required and others optional. *Examples* are filled-in templates, preferably actual instances of a work product, such as a test script. Examples give practitioners guidance about style, level of detail, and vocabulary.

The ultimate goal is to create a description of an organization's entire software engineering process. This is a long-term, ongoing effort which can take years. More limited efforts can pay off in the short run while the larger effort continues. For example, it may be useful to place all intermediate work products—design documents, test plans, and source code—under configuration control, thus enforcing exit criteria throughout the existing process. This should lead to increased stability and greater visibility of work products.

Another useful tactic, once configuration management has been established, is to completely define one phase of the software engineering process, preferably the one most widely used by project personnel. This is typically the coding phase. Once the process for one phase is well defined, an interfacing phase—perhaps unit testing or module design—can be more easily defined.

4.2. Defining the Desired Process

When the existing process has been described, the work to define the desired process can begin. This effort can require extensive resources and is a long-term effort. The process group must coordinate the concurrent efforts of the working groups to create the definition, and everyday use of the defined process cannot be expected for some time. This is one reason why executive sponsorship and the steering committee are so important: a long view and the resources to support it are essential.

Process group members work with the technical working groups initially to prepare descriptions of process phases, and then, over time, to define activities within process phases to the desired level of specificity, including the creation and review of standards, templates, and examples. The process group acts as editor and librarian of the growing collection of process definition material, ensuring, for example, that exit criteria from one phase or activity match entry criteria for the next. The process group also coordinates the efforts to tailor a process definition for a new project. As new projects begin, they draw material from the process definition library and adapt it. Working group representatives work through the process group to provide training and consulting for project members who will be using the new material.

What makes a good process definition? It should be all of the following:

- **Broad**. It should define the software life cycle. This includes all activities, and internal and deliverable documents, related standards, and constraints.
- **Deep**. It should define the process aspects at different levels of abstraction. It should include all formal and informal connections among tasks, phases, and work products.
- **Flexible**. The process definition should be able to describe phases that never end (such as requirements definition), as well as tasks that have crisp terminations (such as inspections). It should describe flows, exceptions, and constraints.
- **Practical**. The definition should accommodate adaptation. For example, if a process definition requires that a baselined component be placed under configuration control, it might require projects to decide how baselines are obtained and how configuration control will occur, to document their decisions, and to submit these plans for review, perhaps to the configuration control working group. The result is high-level consistency across projects, and consistency at a more detailed level within projects.
- **Measurable**. The process definition should be measurable so that statistical control of the process can be obtained.
- **Auditable**. It should be specific—concrete enough for an independent agency to use in making an objective, repeatable judgment about whether the defined process has been followed.
- **Evolvable**. The definition must include a provision for orderly change.

There is a growing body of literature about process definition and process modeling. [Freeman87] treats these subjects extensively, under the name "system development system." An SEI curriculum module [Scacchi87] is dedicated to these topics. Readers should also examine DoD-STD-2167A [2167A] and the IEEE *Standard for Software Life Cycle Processes* [IEEE-SLCP89], now available for public review. In addition, the international workshops on software process contain considerable material on definition; the latest proceedings is [SoftProcWork89]. Each of these references will mention others. As mentioned earlier, Chapter 13, "Defining the Software Process," in [Humphrey89] is recommended for reading prior to beginning a description or definition of an organization's process.

4.3. Process Definition and CASE Technology

Until an organization's software engineering process has been described and at least partially defined, it is premature to evaluate the applicability of a particular method or tool. Tools and methods are, in effect, instantiations of aspects of the development process. After the process has been described—and preferably not until it has been defined—methods and tools can be evaluated based on their fit and contribution to process improvement. In fact, a process definition can point to the need for a method or tool and not only help to justify it, but provide detailed requirements for its functionality. (See the discussion in Appendix D for other factors to consider in acquiring new technology, including how it may fit within the context of organizational culture and software process maturity.)

5. The Process Database

The process database is a central collection of files containing all critical information on the process and product aspects of software development and maintenance. It is maintained by the process group. Using simple definitions, the measurement portion of the database should contain a few indicative, useful measures of the products and processes. In addition to a quantitative database, the process group should maintain a file of process improvement lessons learned.

The process database is a central collection of files containing all critical information on the process and product aspects of software development and maintenance. It is maintained by the process group, which may work with a metrics working group, a defect prevention working group, and the steering committee to determine what the database should contain and how the data should be used.

Measurement motivates people to change their behavior—they agree on a target and work toward it. As progress becomes reflected in the measured results, people make modifications to improve the outcome; this is why practitioners need to be involved in defining the measures used.

5.1. Measurement

This section provides introductory material on establishing the measurement portion of a process database. Further guidance can be found in the following references, ordered from general to detailed: [Grady87], [Humphrey89], and [Flaherty85].

The measurement files of the process database contain quantitative information about how effective the process is. This information can be collected for a number of reasons and used in a number of ways. [Humphrey89] cites the following:

- **Understanding**. Data may be gathered as part of a study to learn about a particular item or process.
- **Evaluation**. The data may be used to determine if a process, product, or activity meets acceptance criteria.
- **Control**. The data may be used to set limits and goals for activities.
- **Prediction**. The data may be used to develop rate and trend parameters for project planning.

The ultimate goal of measuring the software process is to reach the point of statistically verifiable control similar to that available in a manufacturing process.[9] Control over a process is obtained by:

[9] [Cho87] explores this analogy at length, focusing primarily on product quality.

- Characterizing the process and making it more visible.
- Using simple measures directly related to what needs to be improved.
- Measuring the process as well as the product.

5.1.1. Visibility

Once the software development and maintenance process is defined, it can become dynamically visible by quantitatively characterizing it at points along the life cycle. A certain range of each measure is targeted as a reasonable goal, and progress against the goal is typically posted or announced publicly. Making the process public has the beneficial effect of making it a team effort [Weinberg71].

5.1.2. Types of Measurement

In general, there are two types of measures: product and process. Product measures describe the characteristics of each system component being produced or maintained. Typical product measures are size, cost, complexity, quality (number of defects), and resources consumed to develop or maintain the component. Process measures describe aspects of the process used to produce or maintain those components, and are attributable to the human processes that create the product. Typical process measures are defect rates, repair rates, and production rates. Broader measures such as labor turnover, learning curve, communications overhead, and degree of overall adherence to the defined process also relate to process. In fact, a single measure may apply to both process and product, depending upon its usage and one's point of view. Both types of measures may be collected on individual components or across groups of components; when product measures are aggregated, they often provide information about the process.

5.1.3. Definition of Measures

To be useful, a measure must be clearly defined. Following are the general characteristics of such a definition:

- It is practical—it can be directly related to the work.
- It is easy to apply, so subjective judgment is not required.
- The same things are treated in the same way.

The experience of those establishing a process database tells us that good definitions require iterative enhancement. It will be difficult to arrive immediately at the best definition.

Because of the difficulty in adequately defining terms (line of code, defect, documentation change vs. design change, etc.), experimentation is required. The purpose of the experiments should be to define terms precisely so that they are understood the same way by all users and can be collected by machine.

Measures that are subjectively weighted, complicated, or removed from the actual software production process are not likely to be effective. Accordingly, the first task will be to "gain agreement on a set of software measurement criteria which managers feel are meaningful, reasonable to collect, and can be used to measure progress and predict results" [Grady87].[10]

5.1.4. Suggestions

Keep the data simple. Even very simple measures have been shown to be effective. For example, it may be unnecessary to collect "hours worked" to any decimal points since variations of up to two times in effort rarely affect measures of the underlying process. (However, that variation might distort the numerical value of productivity.) Additional examples can be found in Chapter 9 of [Grady87].

There is an expense to collecting and validating data, but one of the factors that can offset the expense is turnaround that is quick enough to take advantage of the collected information. One must be careful when making claims about quick turnaround, though, because it takes quite a while to validate and process such data. The quick turnaround—while difficult to achieve in practice—will enable managers to see value added to the collection overhead.

Start modestly. Use the tools already available for data collection, presentation, and storage. For example, one can use a window in a development environment as a means of collecting information online, more or less in real time, regarding effort and schedule. This data can be transmitted via electronic mail and placed in a spreadsheet for "first-pass" validation, consolidation, stripping of the identity of its source, and presentation in tabular and graphic form. One does not need a sophisticated (read "relational") database in which to store elementary measures.

Remember that the development of a useful process database is a long-term process that requires the consensus and feedback of users to refine the definitions, the measures, and the requirements for raw data collection.

5.2. The Database

The purposes of the process database are these:

- To indicate trends and long-term effects.
- To experiment with definitions in order to arrive at a meaningful set.
- To indicate trouble spots that are evident using (even rudimentary) measures.
- To support a basis for estimates.

[10]The SEI has recently begun an initiative to provide standard definitions for size (such as source lines of code), effort (such as person-months), quality (defects), and schedule (elapsed time). This initiative includes industry, government, and academia.

5.3. Defect Prevention

Several process information files are needed to support the defect prevention process described by [Mays90, Gale90].

- **The defect file** contains a record of each defect and information such as originator, description, resolution, disposition, and effort to repair.
- **The action item file** contains the action items that arise out of a causal analysis of the defects. These items are suggestions for improving the process so that future defects would not be manifest in the software product.
- **The kickoff file** contains process descriptions of development phases. A process description package is printed by selecting the appropriate descriptions for a specific project and phase from among all those available in the file; this package is then presented at the meeting that kicks off the phase. The elements of the process are usually drawn from separate files that collectively represent the current best thinking regarding that phase.

5.4. Accessibility

Care must be taken never to use database information for personnel or individual project evaluations. There are many factors that cannot be taken into account by our primitive measures (for example, the best programmers often get the hardest assignments, so they may show the lowest productivity). Additionally, if the measures were used for evaluations, the temptation to manipulate the figures to show performance in a favorable light could make the data less useful for predictive and problem-solving activities. Thus, the database could no longer be used for process improvement.

On the other hand, access to composite results is necessary. Managers can better characterize and understand the process for which they are responsible when they can compare it to the values of central indicators.

Easy access to summary data achieves several objectives:

- It gives the software practitioner and manager some "ownership" without the concern of custody and maintenance. Although there is some cost to create the input to the database, easy access is a counterbalancing benefit.
- It gives the manager a free hand to experiment with measures that may be meaningful to him or her, but are not part of the "canned" suite.
- It allows managers to compare their performance with others in order to understand how they differ and offers them the opportunity to find out why they differ (for example, a manager might call another project manager that has interesting numbers).

As an example, Hewlett Packard maintains a central database and regularly distributes a summary version of the whole database to every project manager for use on his or her personal computer spreadsheet program. There is a suite of stock reports and graphs; in addition, the manager can use the spreadsheet program to analyze and present the information in any manner that is useful. A sophisticated (read "expensive"), distributed, shared, relational database management system does not seem to be necessary.

5.5. Lessons Learned: Beyond the Database

In addition to a machine-readable database, the process group should capture the history of the improvement efforts. The group should collect, catalog, and maintain reports and other artifacts related to attempts to improve the process. In many organizations, these products are lost when personnel change, and the improvement lessons must be learned all over again. Records should include: an indication of what went well and what did not go well; a description of how problems were handled; and suggestions for improving performance on similar tasks in the future.

Capturing and reviewing lessons learned can be a part of the development life cycle [SMAP4.3]. The challenge is to make this information available to those who need it, past the life of a project.

The development of a useful measurement program and process database is a long-term process in itself. It will require the consensus and feedback of users to refine the definitions, the measures, and the requirements for raw data collection.

Part II — Ongoing Activities of the Process Group

6. Beginning Continuous Improvement

The process of introducing an improvement includes selecting a candidate technology, tailoring the technology and the training for its use, and using the technology on a pilot basis. Feedback should be collected and future plans modified in light of the pilot experience.

The activities discussed thus far constitute the *beginning* of the process group's mission: continuous process improvement. These activities can be compared to the blueprint and frame of a house; now the building must be completed, occupied, and maintained.

The process group and its partners—the steering committee, the working groups, and the projects themselves—will have steady-state, ongoing jobs in accomplishing process improvement.

One key long-term activity is the installation, over and over again, of new procedures and technology that support process improvement. These installations should begin with a pilot (prototype) installation—a controlled experiment. Pilots are essential when an organization has no experience with a technology, or when the technology will be applied in a new domain or with inexperienced staff. This is true even if the technology seems to be mature, as with software inspections, or even if the organization has excellent in-house resources, such as a technical working group with some experience in cost estimation. If a technology is new to an organization, it cannot be considered mature in that context. Appendix D presents an extended discussion of this phenomenon. This chapter describes some considerations for executing a pilot effort.

6.1. Introducing the Change

Pilots are often initiated in response to activities of a working group. For example, the assessment and resulting action plan might identify the need for a new configuration control system. The working group would spend time investigating possible systems and choose the most likely one; and a project would be identified as a candidate to try the new configuration control system. The sponsor, the steering committee, and some members of the project targeted for the new technology then agree to proceed. To provide information to those involved, the process group leader organizes a briefing session in cooperation with a representative of the technical working group for the technology in question. The following people should be included: anyone who may be affected by the change; *all* members of the candidate project who can be identified as strongly supporting or opposing the change; and representatives from internal training and education organizations. If standards or quality assurance practices will change, representatives of those groups should also be invited.

At the briefing, information about the particular software technology being considered for use should be presented by a person who has experience in helping similar organizations adopt that technology. This might be a member of the technical working group that has been studying the technology; an in-house expert can certainly save time and expense, and has the added advantage of familiarity with the technical context and application domain of the target. If an in-house expert is not available, this is the time to bring in an outside consultant. The technical working group can probably recommend candidates; they may already have established consulting relationships with particular experts.

At the briefing, the expert selected should objectively discuss the pros and cons of the particular technology. He or she should discuss both the technology and the process of putting it into place. Time should be allowed for attendees to ask questions and express concerns. The briefing should address the following:

1. What problem is the technology intended to alleviate?
2. How does the technology solve the problem?
3. How well does the technology solve the problem? What are the benefits of the technology, both qualitative and quantitative?
4. What are the costs; for example, product cost, installation, training, time away from the job, and start-up costs?
5. Where else has the technology been used, both inside and outside this organization? Are there sources of information on use in comparable contexts?
6. Will the technology require revision of the current technological environment? Areas to be considered include: standards; other technology such as software tools or hardware; databases; and schedule and cost-estimation procedures.
7. Will the technology require revision of the current organizational environment? Areas to be considered include: reward systems; job structure; responsibilities and authority; policy; and budgeting.
8. How similar or different is this technology from what is in place now? Is training required? How much? Will it need tailoring? Will people be given the time necessary for classes?
9. Is this a good time to install new technology? Can it be used immediately? How does the technology relate to development and maintenance schedules? Is there enough lead time for people to try out the new technology prior to using it in production?
10. Can the technology be tailored? Will tailoring be necessary? If so, what will it cost? Are resources available from the working group, vendor, or other source?
11. What are other possible consequences of using the technology?

Many of these questions may be identified, but not answered, during this meeting. A list of questions not answered can be used as a guide for further investigation.

Finally, the pilot test site for both the technology and the implementation process should be confirmed. This site should be representative of the projects that are likely to use the new technology, so that the pilot installation serves as an adequate test. To be considered typical, it should be neither too simple nor too complex, so that the results of the technology installation are not attributed to the peculiar nature of the pilot. The pilot should have an appropriate point in its schedule for beginning the implementation process; and, most important, the pilot project members should be willing and able to collaborate with the process group in the implementation effort.

6.2. Pilot Procedures

The installation of technology must be carefully planned. Prerequisites to beginning actual use of the technology are listed below, in the order in which they should occur.

1. **Obtain the technology.** If a tool or method must be purchased, the process group, collaborating with the pilot project and a working group, should work with suppliers to select and obtain it. Suppliers may be in-house tool builders, projects from other divisions, or outside sources such as vendors and government repositories.

2. **Create in-house expertise.** Early use of the technology should be limited to process group members, members of the technical working group, and pilot project engineers who will help draft guidelines for use. (Broader use should follow preparation of the guidelines, which in turn is an outcome of the piloting process.) If skills and knowledge needed for preparing the guidelines for use are not available, training should be obtained for those involved in the task. Training should be specific to the technology coming in. For example, a general course on configuration control is good background if the technology in question is a tool to support this. However, such a course may need to be supplemented by tool-specific training before meaningful guidelines can be written. *This specific training is absolutely essential, and, if omitted, will seriously jeopardize the success of the prototype effort.* Lack of training for key people inside the organization can result in misapplication and underuse of technology; it can unnecessarily extend the learning curve for effective use.

3. **Draft guidelines for using the technology.** Ultimately, all pilot project engineers must be able to use the new technology. Training may not be necessary for each engineer, but all must have guidelines for use. If example guidelines are available from the technology supplier, the technical working group, aided by the process group and coordinating with the training and education function, can work to tailor these for trial use. Guidelines should specify procedures for each category of technology user, including managers who may use the new technology indirectly.

4. **Tailor the training to the needs of project members.** If training is necessary for the pilot project engineers, it should be provided in a form compatible with the guidelines. Students should not spend valuable class time attempting to reconcile differences between training materials and guidelines. Since training, if needed, is the last step prior to "live" use of the technology by all pilot engineers, it should result in skills and knowledge that can be applied immediately upon completion of the course. Tailoring the training materials so that they reflect the vocabulary and technical environment of the pilot project helps ensure immediate applicability of skills mastered in the course. Tailoring should be done jointly by the working group and the training and education function; the process group should be kept informed.

5. **Prepare the technology for use.** Perform any work needed prior to allowing broad access to the technology. This may require broader installation than during the development of guidelines and tailoring of training. If the configuration or performance of the technology can be tuned, that should be completed before new users become involved.

6. **Provide the training.** In the long run it pays to train *all* potential users of the new technology. One misapplication can cost far more than a few staff hours of training. It is best to train each team together, and to train them immediately before they will begin to apply the technology; if more than two weeks elapse between the training course and application, performance will be seriously degraded. The process group leader, the sponsor, and the pilot project manager should begin the training session with information on the pilot objectives and background on the technology selection process. Trainees should be encouraged to provide feedback on the training, the guidelines (distributed as part of the training materials), and the technology itself. Training sessions thus offer management an opportunity to explain objectives and demonstrate commitment, and give trainees the opportunity to "buy in" and/or discuss problems with those who are in a position to solve them.

7. **Begin using the technology.** During the first weeks of use, the process group and the technical working group should be available to consult with the new users. Some initial difficulties are to be expected; the process group should note these and correct the guidelines or training, or tune the technology. They should be careful to differentiate problems of the technology from problems of use.

8. **Provide ongoing support.** Important means for providing ongoing support include feedback mechanisms such as hot lines, ongoing consulting, user group meetings, and systematic data gathering to determine how well the new technology is being used and how the customers are doing.

6.3. Beyond Pilots to Routine Use

After the pilot project has used the new technology for about six weeks, and again after a few months, the users and process group make an evaluation. (The time periods for check-ups depend on the nature of the technology, the size of the pilot project, etc.) The process group and the working group members who have been involved should summarize the results and confer with the steering committee about the advisability of attempting another pilot. Results should be reviewed in terms of both the new technology and the process for putting it into place. It is most helpful if quantitative data can be obtained to indicate the effect of the new technology, but other forms of feedback—hot line records, memos, trouble reports, notes from focus groups, minutes of user group meetings, or customer complaints that can be traced to the new technology—can be useful as well. After the need for refinement has leveled off and key objectives have been met, more pilots can be initiated.

Usually a minimum of three pilots is advised before attempting organization-wide use. Completion of the piloting process is indicated when installation of the new technology can be estimated and scheduled predictably. When the experience in one pilot differs significantly from that of another, more pilots are required.

7. Mechanisms for Information Transfer and Implementing Change

Many mechanisms are available to aid the implementation of technological change. The most common one, information transfer, is a good starting point but is insufficient by itself. For change to last, organizations need additional mechanisms to assist in the routine use of new technologies.

There are many mechanisms that support the transfer of technological information and the implementation of technological change. Most are so common that they are taken for granted. For example, some organizations establish a policy that each professional employee will have several days of continuing education per year. Some routinely develop and maintain standards and procedures to ensure consistent engineering practice. Others sponsor university research and go on to incorporate the resulting technology. And there are many other approaches in everyday use. Two fundamental problems often limit the success of information transfer and change activities: mechanisms are not purposefully combined in the context of a strategy for implementing a particular objective, and one or two mechanisms are expected to do the work of several.

The success of process improvement depends on the adroit application of appropriate mechanisms. In this chapter, we will discuss mechanisms for both the transfer of technological information and the implementation of technological change. We will also discuss strategies for using these mechanisms.

7.1. Information Transfer Mechanisms

Information transfer mechanisms are those that serve to disseminate information or knowledge about a particular technology. These mechanisms, summarized in Figure 7-1, allow prospective users of a technology to proceed through the first four stages of commitment to organizational change [Conner82]: contact with new technology, awareness of its implications in general, understanding of how it might change an individual's particular environment, and positive perception of the possibility of using it. Typically, these information transfer mechanisms present a technology in a generic or unadapted state. People participating in conferences or seminars, for example, must hypothetically "map" the technology to their specific context. These mechanisms might provide, for example, the opportunity to gain an overview of object-oriented software design or cost-estimation models for software projects; but other mechanisms, such as those discussed in the next section, are prerequisite to actual use.

Organizations which employ professional engineers or which engage in research and development work often assume that it is sufficient to make mechanisms available, such as those listed in Figure 7-1. This laissez-faire approach to information transfer is effective only to the degree that employees take advantage of the opportunities provided.

Mechanism / Audience	Briefings	Executive Seminars	Libraries or External Literature	Company Newspapers	Company Journals	Internal & External Seminars &Conferences	Peers	Consultants	Vender Demos	User Newsletters or Electronic Bulletin Boards	Brown Bag Lunches	Sponsored Research Reports
Upper Management	X	X		X	X	X	X	X				X
Middle Management	X		X	X	X	X	X	X	X			X
First-line Supervisors & Engineers			X	X	X	X	X	X	X	X	X	

Figure 7-1: Mechanisms for Information Transfer

An organization may wish to proceed more deliberately and systematically, as in the case of setting goals to move toward increasing levels of process maturity. If this is true, it must have a planned and purposeful dissemination of information throughout the organization.

Once the assessment and action planning have identified areas for improvement, including technological needs, the organization needs something analogous to an internal marketing campaign. Market segments (such as levels of management, potential sponsors, and various communities of engineers) can be targeted and, after research, appropriate information transfer mechanisms chosen for each. The process group, the technical working groups, and the steering committee should plan this program and cooperate in implementing it. A specific overall goal must drive the strategy; for example, moving up one full level of the process maturity framework within 24 months.

OBJECTIVE: Create awareness of technology that can support software process improvement resulting in a Level 3 by a self-assessment in 18 months.

OVERALL STATEGY: Determine candidate technologies. Identify potential sponsors and collaborators for meeting this objective. Prepare a schedule of opportunities to provide sponsors and collaborators with exposure to these technologies.

STRATEGY FOR EXECUTIVE SPONSORS: Analyze opportunities to make contact with senior executives. Lay out a systematic "campaign" to provide them with information on the relevant technology (including its relationship to business conditions and information on what other organizations are doing) at the right times.

Tactics might include:

1) Briefing potential sponsors or getting consultants to do so, and then having the sponsors brief or meet with peers.

2) Getting a place on the agenda of annual executive seminars, or internal technical conferences where senior executives may appear, for an appropriate level speaker who can discuss both the strategy and candidate technologies.

3) Getting coverage of candidate technologies in the company newspaper; providing writers with appropriate material as needed.

4) Getting on the agenda of meetings of standing committees, e.g. corporate quality committees or task forces, or software technology steering groups, to discuss strategy and candidate technologies.

5) Getting a task force set up to provide input on the strategy.

6) Lobbying to have executives set policy for managers and engineers to attend relevant outside training and conferences addressing the technology areas identified.

STRATEGY FOR MANAGERS: Analyze opportunities to make contact with middle managers and first-line supervisors. Plan a systematic "campaign" to provide them with information on the relevant technology (including information on what other organizations are doing, the connection to business conditions, and resource requirements) in a timely way.

Tactics might include:

1) Briefing potential sponsors individually or in standing committees or organizational groupings, and then having them brief their peers.

2) Requesting the technical library or its equivalent to subscribe to journals and then having them periodically circulate tables of contents to a distribution list you have created. Circulating research reports to the same list.

3) Getting coverage in the company newspaper; providing writers with material appropriate to the interests and concerns of managers.

4) Getting knowledgeable sponsors or consultants to appear at internal management seminars or conferences to discuss the objectives and candidate technologies.

5) Having vendors conduct special management demonstrations of products.

6) Running focus groups with managers to bring concerns to the surface, gather ideas, issues.

7) Suggesting that managers attend certain outside conferences or training workshops, and/or have their people attend these.

STRATEGY FOR COLLABORATORS: Analyze opportunities to provide information to and gather ideas from engineers who may use or help implement the new technologies. Use these same occasions to identify "opinion leaders", potential process group members, potential working group leaders and members, special needs, and internal experts in the technology areas.

Tactics might include:

1) Having the technical library or its equivalent subscribe to appropriate journals; making a distribution list and then having the library circulate tables of contents periodically.

2) Getting coverage in the company newspaper; provide material targeted to the engineers who may use a new technology.

3) Providing experts from outside and inside the organization to present talks at internal seminars, colloquia, and brown bag lunches.

4) Running focus groups, inviting both key advocates and skeptics, to solicit ideas, concerns.

5) Arranging for demonstrations by vendors of processes, methods, and tools that are candidate technologies.

6) Setting up electronic bulletin boards or newsletters providing news of and ideas about candidate technologies, and also information on local experts and users, sources of reference material, and descriptions of conferences and training workshops.

Figure 7-2: Example of Strategy for Software Process Improvement

The specifics of the strategy will vary widely depending on the organization. A strawman strategy for process improvement within a particular division of a large corporation is illustrated in Figure 7-2. The major difference between this strategy and a more usual technology-driven approach to information transfer is simply that the proposed strategy uses ordinary mechanisms very systematically with the goal of meeting a focused objective with a far-reaching effect.

7.2. Mechanisms for Implementing Technological Change

Mechanisms for implementing change are those that support the processes of adapting, installing, and routinely using a new technology. According to [Conner82], the next four stages of commitment to change involve beginning and ongoing use of a new technology; he labels these installation, adoption, institutionalization, and internalization. *Installation* is self-explanatory. Although pilot efforts are not discussed per se, the definition of *adoption* does imply the need for a shakedown period. *Institutionalization* is defined as broad and routine use across the whole organization (which takes an extended period of time to achieve). *Internalization* is defined as the stage where the change has been incorporated, not just into job performance, but also into the personal values of employees. Internalization of the need for and underlying principles of process improvement is probably a desirable goal; that degree of commitment may not apply to a particular implementation because many different technologies can support the same principles.

Most of the day-to-day work of the process group and the working groups has to do with implementing change; the groups use many of the mechanisms listed in Figure 7-3, such as consulting, planning and implementing pilots, screening and acquiring specific tools, revising standards, suggesting policy changes, arranging apprenticeships, running hot lines, and editing newsletters. The essential difference between these mechanisms and information transfer mechanisms is that the latter deal with *information about* generic technology, and the former deal with *adapting and using* technologies in a very specific context.

There are many variations on the mechanisms listed in Figure 7-3 and many additional mechanisms that can be used. Good sources of ideas are [Przybylinski88] and [Morton83], both of which document the experience and thinking of engineers and consultants actively working on software engineering technology transfer issues.

7.3. Applying Both Types of Mechanisms

Whether a change effort involves a major change such as beginning an improvement effort, or a particular software improvement activity such as software inspections, its effectiveness depends upon careful planning and awareness of available, applicable technology. The process group, with the help especially of the working groups, must keep information about potentially useful technology "in the air." Drawing attention to available technology long before any specific application of it is contemplated can reduce the effort involved in justifying action planning for using it or introducing it in a pilot. These efforts help educate potential sponsors and create a well-informed and interested constituency.

Mechanism / Audience	Internal or External Consulting	Pilot Use	On the job, Standard or Custom Training	Tools	Funding	Standards	Policy Revision	New or Revised Procedures	Apprenticeship	Hot lines	User Newsletters or Electronic Bulletin Boards
Upper Management	X				X		X				
Middle Management	X	X			X	X	X	X			X
First-line Supervisors & Engineers	X	X	X	X		X		X	X	X	X

Figure 7-3: Mechanisms for Implementing Technological Change

To accomplish this part of their process improvement work, the process group and the technical working groups must keep themselves informed about current software engineering technology. They do this by joining professional societies, reading the professional literature, attending conferences, and maintaining contact with internal and external peers, experts, and vendors. They filter information from technology producers and advocates, continually attempting to find the best match between the available technology and the needs of their organization. And they use their knowledge of software engineering technology and of change and information transfer mechanisms to orchestrate the best combination of activities in support of their organization's improvement goals.

8. Process Consultation

The process group spends a significant proportion of its time coaching others and problem solving, while being alert to examples of best practices. This consulting and broad awareness of the quality of particular efforts is indispensable for the success of the overall process improvement program.

As the focal point for process improvements, the process group spends a significant proportion of its resources meeting with those whom it serves. The group's consulting activities include:

- Helping to set realistic expectations for improvement activity results.
- Tailoring process priorities, definitions, standards, training, and other process materials.
- Suggesting methods for improving a specific process if a working group has not already done so.
- Analyzing process measurement data.
- Facilitating improvement planning meetings and activities.
- Demonstrating improvement technology. (For example, serving as a moderator for inspections until a project can develop its own inspection moderators.)
- Referring groups with similar interests to each other so that they can help each other.

Process group members must have or develop good consulting skills; they need the ability to listen and clarify, and to collaborate in problem solving with those who seek their help. If process group members have access to training or expert advice (especially in the form of "shadow" consulting) in this area, they should take advantage of it.

In the organization development (OD) community within management science, there is considerable emphasis on systems—in the sense of an organization being a system—and on engaging the entire system in any change processes. The thinking of modern OD consultants thus is quite compatible with process improvement. Process group members teaming with OD consultants should expect that there will be much value in such an arrangement; but the OD consultant may need as much indoctrination in software technology and culture as the process group members do in OD skills and practice.

An emerging source of process consultation experience and expertise is the community of augmentation groups.[11] As part of this community, process group members are able to

[11]Douglas Engelbart at the Conference on Computer Support for Cooperative Work, Portland, Oregon, September 27, 1988, suggested this term; his augmentation groups are roughly equivalent to process groups and are not limited to the software industry.

share experiences across disciplines and organizations. Those involved in software engineering process improvement could learn from other process improvement fields (e.g., dentistry, nursing, education, printing, VLSI design and manufacture, and flexible manufacturing). This may result in the development of tools that aid process improvement; for example, structured ways to set expectations, assess needs and create improvement plans, manage resistance, and reuse improvement experience (by sharing).

Part III — Membership and Placement

9. Process Group Membership

Process group members should collectively have experience from throughout the software life cycle. They should have experience with multiple frames of reference, and their backgrounds should complement each other. Membership is usually (a) full-time, (b) for a fixed period of time, and (c) recommended as a condition of promotion to a senior position.

Members of the process group are advocates for improvement. They are software professionals assigned, generally full time, to help the organization increase the maturity of its software process. Members should be carefully selected with the goal of balancing the experience and educational background of the group as a whole. Suggested selection criteria (see below) should be tempered by an awareness of the technical and social characteristics and corporate culture of the organization that the process group will be supporting.

The process group works with sponsors (executive management), a steering committee (line and program managers), working groups (usually senior practitioners), and project staff. The group works with domains such as systems engineering, marketing, field support and maintenance, and training. Process group members are *boundary spanners* (see Appendix D) who bridge layers of organization, function, structure, culture, discipline, and competing interests. They translate the terms of one group into the terms of the other.

9.1. Selecting the Process Group Leader

The sponsor, with the advice of the primary advocate for the process group or the steering committee, selects the process group leader. The selection should be endorsed by key members of technical staff, especially influential members of working groups and key members of management. The process group leader *must* be an acknowledged technical leader, with these characteristics:

- Extensive experience in or knowledge of the software process.
- Experience advocating improved software development processes, methods, and tools—that is, improved quality and productivity.
- Experience in management or project leadership.
- Knowledge of the software development environment.

Candidates for process group leadership may be found among senior technical staff who have lobbied for improvements within their software projects. If working groups or other standing technical committees exist, these are good sources of candidates. The quality and appropriateness of the choice of process group leader will clearly indicate the degree of management commitment to software engineering process improvement.

9.2. Selecting the Process Group Members

Working with the sponsor and the steering committee, the process group leader then helps to select the other members.

Each process group member should meet this fundamental set of qualifications:

- Be a regular employee of the organization, with experience within the software projects that will be served by the process group.
- Have application domain expertise.
- Have a basic understanding of the software development process.
- Have knowledge of local software development methods, tools, and practices.

The experience of process group members should be in keeping with the standard of excellence for software practitioners in the organization overall.

Experience in a process group or its equivalent is extremely desirable but very rare. Experience in planning for and installing new software methods and tools is also very helpful and more common. The important principle underlying the types of experience is that process group members should have worked in several different contexts and have experience with multiple frames of reference (see Appendix D).

In addition, process group members need consulting skills and strong oral and written communication skills. They must be able to communicate effectively with peers and superiors, and must have a reputation for productive work relationships with both. An open mind is essential, as group members must be able to rapidly understand and appreciate many different technical and organizational contexts.

The composite background of the process group must be considered. The group should have experience that supports objectivity about the organization's technical environment, culture, management style, and reward systems. This experience includes:

- Work on other software development projects.
- Work in other companies or government organizations.
- Experience in a variety of software development or support/staff positions, for example, in testing, design, and product assurance (rather than in just one of these).
- Experience with other target and host computers, as well as other software development tools and environments.
- Use of other software development processes.
- Work in other application domains.
- Attendance at professional society conferences and workshops.

A process group having members experienced in software design, testing, management, tools, maintenance, proposal preparation, and quality assurance will be able to work credibly with the steering committee, a range of working groups, and other software practitioners.

The process group is also a good place to use experienced, very senior people who have held top management positions. These people have accumulated many lessons of value to their organizations; and if they are willing to do engineering work, they offer a good foundation on which the process group can build. Their understanding of corporate culture and management style and their broad base of personal contacts and alliances can be particularly useful.

One pitfall in selecting process group members is a tendency to place people in a process group who are not successful elsewhere. This would be a clear signal of management's ambivalence about the importance of the process group. Competition for people with the characteristics needed for process group membership is to be expected. The advantages of a rotation through the process group should be presented in response to concern about use of valuable staff members in the process group (see Section 9.4).

9.3. Length of Membership

Tenure of two to three years is recommended for process group members. Membership should be staggered, allowing a month or two of overlap as members are replaced. Ideally, process group members should come from and return to line organizations; i.e., organizations building or revising software.

9.4. Advantages of Membership

The organization gains an advantage when key technical staff join a process group: the central role of the group allows great leverage of valuable talent. For members of the process group, the experience offers exposure to a broad collection of line organizations, management groups, business areas, and software development processes and practices.

The work of the process group work requires members to exercise communication skills in preparing and refining plans and briefings. Group members develop leadership ability as they work in cooperation with peers and management. Because of this, process group members may become qualified for promotion to management in a short time.

The process group also provides excellent experience for tool builders and computing system environment staff. Environments and tools become more important as an organization's software process becomes more refined and predictable, that is, more mature. Work with a range of projects and software development processes gives group members prerequisite experience for effective tool and environment design and support.

10. Placing the Process Group in the Organization

The process group must be located where it can honor its working relationships, foster collaboration among the various groups it supports, and help integrate the many aspects of an overall improvement effort. The process group works at multiple levels in the organization; its position and scope of influence are important considerations.

The actual placement of the process group in the formal organization chart will depend on the particular organization. The first rule of placement is this: *the process group must be located where it can honor its working relationships*. In some organizations, the process group reports to the director of engineering and is part of the engineering staff organization that also manages standards, computing facilities, and advanced development. In others, the process group is part of a large software engineering tools development effort which also serves the broad software engineering community within the organization. In still others, the process group reports to the manager of software development for a particular product line. Wherever it is, the group must be able to:

- Work with managers at multiple levels.
- Work with both staff and line organizations.
- Work across project boundaries.

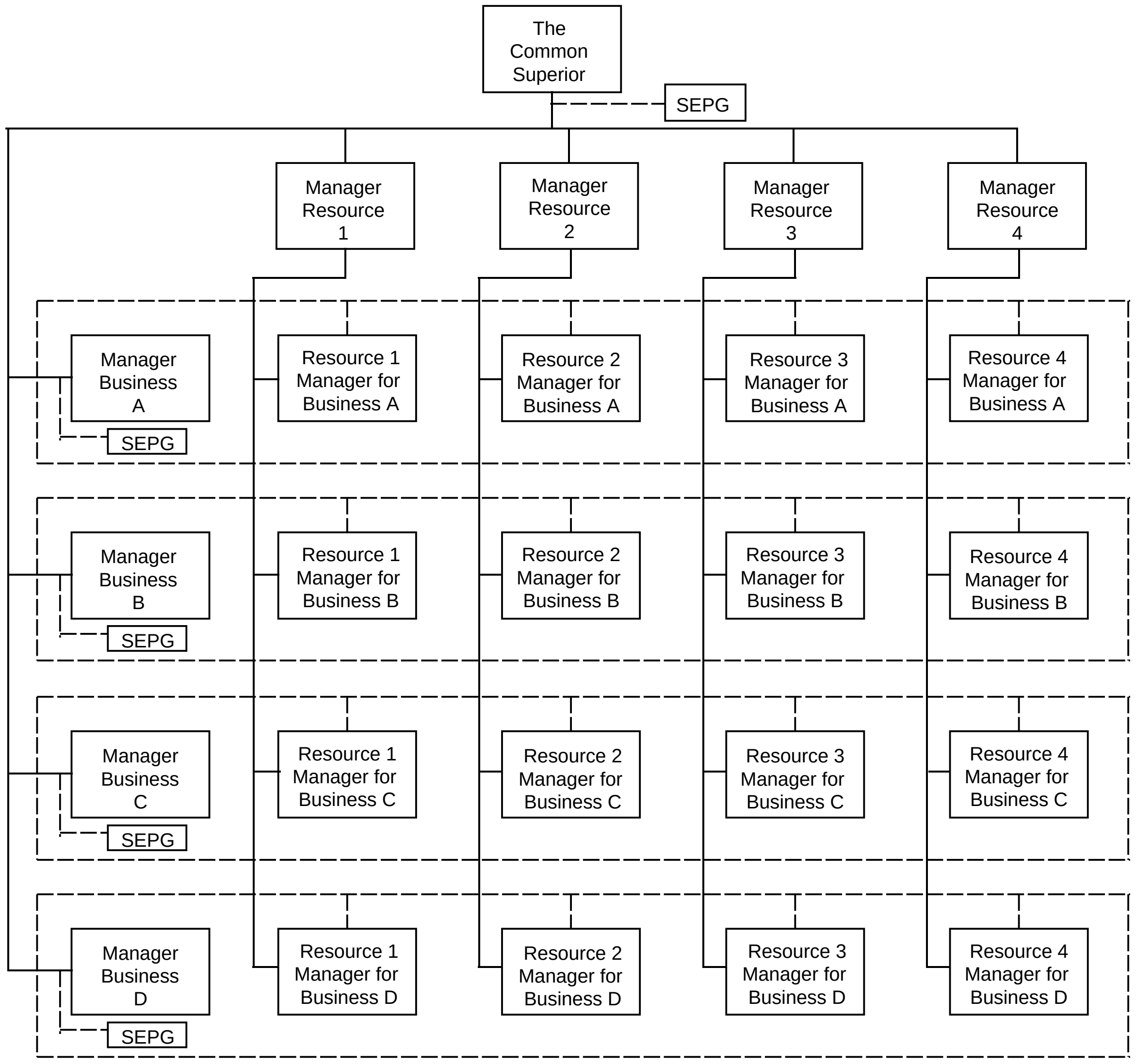
The process group, steering committee, and working groups are an organizational coalition for process improvement. Viewed in terms of ordinary organizational structures, however, this coalition is a collection of task forces, a management committee, and a relatively low-level staff group. It is, in effect, a largely informal parallel organization [Stein80]; and like most informal organizational entities, it is at risk because it is outside direct product-producing activities. Recognizing the importance of groups to the competitiveness of the organizations they serve, Kanter [Kanter83] suggests the following solution:

> The idea behind having a second, or parallel, organization alongside routine operations only makes explicit what is already implicit in an integrative, innovating company: the capacity to work together cooperatively regardless of field or level to tackle the unknown, the uncertain. In a formal, explicit parallel organization, this is not left to happenstance but is guided—managed—to get the best results for both the company and the people involved. (p. 359)

Many scenarios are possible for having a parallel organization for process improvement. One used by Westinghouse suggests a steering committee reporting to a group vice president in charge of a product line, the process group's administrative core reporting to the manager in charge of software engineering standards and tools, and the working groups reporting to the steering committee. This steering committee meets monthly with the working group, and monthly alone. Placing the process group in a matrix organization is a special case; a proposal for this follows.

10.1. Special Considerations in a Matrix Organization

A matrix organization is one in which most practitioners have two supervisors: a product or project manager and a resource manager. Product or project management (referred to here also as program management) is the function responsible for the cost, schedule, and delivered product. Resource management is the function responsible for the development; in software, this includes analysis, design, programming, and testing. Such an arrangement is depicted in Figure 10-1.



The Common Superior
SEPG
Manager Resource 1
Manager Resource 2
Manager Resource 3
Manager Resource 4
Manager Business A
SEPG
Resource 1 Manager for Business A
Resource 2 Manager for Business A
Resource 3 Manager for Business A
Resource 4 Manager for Business A
Manager Business B
SEPG
Resource 1 Manager for Business B
Resource 2 Manager for Business B
Resource 3 Manager for Business B
Resource 4 Manager for Business B
Manager Business C
SEPG
Resource 1 Manager for Business C
Resource 2 Manager for Business C
Resource 3 Manager for Business C
Resource 4 Manager for Business C
Manager Business D
SEPG
Resource 1 Manager for Business D
Resource 2 Manager for Business D
Resource 3 Manager for Business D
Resource 4 Manager for Business D

Figure 10-1: Placement in a Typical Matrix Organization

Adapted from [Janger79].

There is a special challenge in placing a process group where it will be of greatest value to the most people. Should it be in the program office, where the final responsibility for quality rests, or in the resource office, which covers all software developers, not just those on certain projects or products?

Typically, resource managers have small operating budgets to cover their internal overhead functions; they receive the bulk of funding by "selling" resources to the program offices. Accordingly, it is the program offices that have most of the funds in a matrix organization. In addition to the preponderance of funds, the final responsibility for the quality of the finished product rests with the program office. Therefore, the program office would be a natural place to house the process group but for a single objection: the program office is responsible for just one program, not all programs. Improvement should be attempted corporate-wide—in all programs—and the experience across programs shared for the common good. Improvement gains in only one program are lost after the program is completed; for improvement to become institutionalized, there needs to be a function that continuously represents the collective experiences of numerous programs and lives longer than and across all of them.

In the search for the proper place, we must also consider a function to which both program managers and resource managers report—the *common superior*. Typically this function does not make decisions but is more judicial in character, arbitrating interactions among the competing needs of program and resource managers [Janger79]. Organizations often keep the "common boss" budget low. This assures that studies are done and decisions are made by those who have to live with the consequences, i.e., the operating units themselves.

There is the very real risk that placing a process group in a corporate staff office may lead to the perception, particularly by program offices, that it will not stay strongly rooted in day-to-day realities. On the other hand, such corporate-level placement indicates upper management's strong commitment by providing high visibility.

One effective arrangement (illustrated in Figure 10-1) is to have a small corporate staff office whose tasks are to: (1) ensure program-level compliance with a corporate mandate that all programs have an improvement program and (2) provide a central means to collect and disseminate improvement history. Under this arrangement, every program has a process group to manage its improvement effort. Part of the program-level improvement program can be administered by the resource office (e.g., basic training), but the funds and responsibility come from the program office. The program-level process group could obtain some of its human resources from a pool administered centrally by the corporate process group; this would ensure that experienced members were available.

Having a local process group in every program allows program-specific priorities to be directly addressed and funding directly justified. This implies that each program would have a maturity level somewhat independent of other programs. The corporate goal would be to increase process maturity across programs, with every program required to set and fund its maturity level goals during each planning cycle. So in addition to the usual indicators of program success, such as cost and schedule, program offices would also be held responsible for progress on their process improvement plan.

Conclusion: A Final Word from the Authors

The quality of the software product is governed by the quality of the process used to develop it. Improving quality may be the most difficult task we undertake as professionals. It takes tremendous energy to counter our own and others' resistance, to establish and sustain high standards, and to inspire others to improve the way they conduct their daily professional lives. As difficult as this task is, it seems clear that approaching it in an organized, systematic manner with the help of management (sponsors, members of the steering committee, and managers of pilot projects) and colleagues (members of the technical working groups, practitioners, and people from other departments and disciplines) offers the best hope of success.

Process group members can look to each other as a source of energy for this demanding task, and they can look to the growing body of process groups and other improvement groups that are emerging. Process groups can also look to the considerable literature, some of which is summarized or cited in the following appendices, for information and guidance. While the road to improvement is a difficult one—requiring, as Deming counsels, constancy of purpose—it is an increasingly populated one, as more and more individuals and organizations recognize that working on quality is not optional but fundamental to a viable future.

The job of the process group is to create and communicate the vision of a preferred future. The very act of creating a vision requires considerable energy. And to hold it out for all to see, criticize, and claim as their own requires even more. What makes it worth the effort is seeing improvement actually take place. We hope that the *SEPG Guide* helps you make improvement happen, and we wish you well on your journey toward that end.

Acknowledgements

In developing the *SEPG Guide*, we have been educated and positively influenced by many people.

We thank Watts Humphrey, director of the Software Process Program at the SEI, an authority on software engineering process improvement and a persuasive advocate of process groups, whose idea it was to write the *SEPG Guide*. Watts supported us with ideas, guidance, and resources. We also thank Dick Martin, director of the SEI Technology Transition Program, and Cecil Martin, manager of Technology Applications, for their understanding of the importance of process groups to software engineering technology transfer, and their provision of resources to support our work.

We are grateful to Dave Card, former SEI resident affiliate from Computer Sciences Corporation (CSC) and now director of Software Process and Measurement for CSC's System Sciences Division, for writing Appendix B, "The Quality Movement and Software Process Improvement," and to Kevin Kendryna, former SEI resident affiliate from the Naval Undersea Warfare Engineering Station (NUWES) and now a TQM trainer there, for his research and ideas on action plans for Chapter 3 and Appendix F.

We are grateful to our colleagues: Rodger Blair provided steady encouragement and thorough review; John Maher gave thoughtful review of our work, tutored us in organization change theory and practice, and introduced us to Marvin Weisbord's work and to ODR Inc.'s *Managing Technological Change*; Don O'Neill improved our understanding of some essential principles of software management, provided the architecture of process improvement, and encouraged us to freely adapt his work in [ONeill80]. We also thank Jim Withey for his thoughts on process improvement and his ideas on how to structure this document; Stan Przybylinski for his willingness to collaborate on roughly hewn ideas and for allowing us to regularly tap his knowledge of the literature on diffusion of innovation and technology management; and Anita Carleton for helping us to process and summarize the data and write the appendix on the results of the 1989 SEPG Workshop.

We are grateful to those who allowed us to interview them about their process group and/or process improvement work: Glenn Secor of the Software Technology Lab at AT&T Bell Laboratories; Gary Wigle of the Software Automation Organization at Boeing Aerospace; Howard Quaife and Gail Sailer of Application Systems Technology at Boeing Computer Services; Judah Mogilensky of the Process Enhancement Program at Contel Federal Systems; Kathy Hornbach of CASE Products Marketing, and the VMS process group at Digital Equipment Corporation; Jerry L. Atkison of Advanced Technology Product Software, Data Systems Division, at General Dynamics; Jim Perry, Fred Yonda, other members of the process group, and their management sponsors, Helen Dyer and Connie Ahara, at GTE Government Systems; Guy Cox and the Corporate Engineering group at Hewlett Packard; Rebecca Smith, formerly of the Software Methods Lab in the Data and Languages Division at Hewlett Packard (now with Tandem Computers); Robert Mays at IBM Communications Products; Kyle Rone of IBM Federal Sector Division in Houston; members of the Software Engineering

Process Office at the Naval Ocean System Center; Cathy Libby, Fred Titsworth, and other members of Raytheon's Missile Systems Division process group; David Kriegman of Systems Research and Applications; Walker Royce at TRW Space and Defense Sector, Systems Integration Group; and Vicky Mosley, Fred Joh, and other members of the Software Engineering Process Organization at Westinghouse Electronic Systems Group. From each person we obtained important ideas to share with our readers, and from the group collectively we greatly enhanced our understanding of the overall of what process groups are and can do.

We thank Dan Clayton (now with Bellcore), members of the original System Development Technology group, and the many software development projects who together pioneered the process group work at AT&T Bell Labs. They were the source of many of the ideas in the guide.

We thank Peter Freeman, then of the University of California at Irvine (now with Georgia Tech), a long-standing champion of software engineering technology transfer and source of the idea of managing this process like a project. We also thank Walt Scacchi, of the University of Southern California, who first introduced us to the literature on *software* technology transfer.

Finally, we are grateful to several authors in particular:

- Robert L. Grady and Deborah L. Caswell, *Software Metrics: Establishing a Company-Wide Program*, Prentice-Hall, 1987.
- H. J. Harrington, *The Improvement Process: How America's Leading Companies Improve Quality*, McGraw-Hill, 1987.
- Watts Humphrey, *Managing the Software Process*, Addison-Wesley, 1989.
- Rosabeth Moss Kanter, *The Change Masters: Innovation for Productivity in the American Corporation*, Simon and Schuster, 1983.
- Tom Peters, *Thriving on Chaos: Handbook for a Management Revolution*, Harper & Row, 1987.
- Marvin Weisbord, *Productive Workplaces: Organizing and Managing for Dignity, Meaning, and Community,* Jossey-Bass, 1987.

References

[2167A]
Military Standard, Defense System Software Development.
February 29, 1988
DOD-STD-2167A.

[Ackerman83] A. F. Ackerman, Priscilla J. Fowler, and Robert G. Ebenau.
Software inspections and the industrial production of software.
In Hans-Ludwig Hausen (editor), *Proceedings, Symposium on Software Validation*, pages 13-40. North-Holland, September 25-30, 1983.

[ASD800-5] Philip S. Babel.
Software Development Capability/Capacity Review (SDCCR).
Technical Report ASD Pamphlet 800-5, U.S. Department of the Air Force, Headquarters Aeronautical Systems Division, Acquisition Management, Wright-Patterson AFB, OH, September 10, 1987.

[Bayer89] Judy Bayer and Nancy Melone.
Adoption of Software Engineering Innovations in Organizations.
Technical Report CMU/SEI-89-TR-17, ADA211573, Software Engineering Institute, April 1989.

[Card89] D.N. Card and R.A. Berg.
An industrial engineering approach to software development.
J. of Syst. and Software 10(3):159-168, October 1989.

[Card90] D.N. Card.
Measuring Software Design Quality.
Prentice-Hall, 1990.

[Carlyle88] Ralph Emmett Carlyle.
Advanced technology groups.
Datamation 34(21):18-24, November 1, 1988.

[Cho87] Chin-Keui Cho.
Quality Programming: Developing and Testing Software with Statistical Quality Control.
John Wiley & Sons, Inc., 1987.

[Conner82] Daryl R. Conner and Robert W. Patterson.
Building commitment to organzational change.
Training and Development Journal :18-30, April 1982.

[Crawford85] S.G. Crawford and M.H. Fallah.
Software development process audits - A general procedure.
In *Proceedings, 8th International Conference on Software Engineering*, pages 137-141. August 1985.

[Crosby79] Philip B. Crosby.
Quality is Free: the Art of Making Quality Certain.
New American Library, 1979.

[Currit86] P.A. Currit, M. Dyer, and H.D. Mills.
Certifying the reliability of software.
IEEE Trans. Software Engineering 12(1):3-11, January 1986.

[Deming82] W. Edwards Deming.
Quality, Productivity, and Competitive Position.
MIT, 1982.

[Deming86] W. Edwards Deming.
Out of the Crisis.
MIT, 1986.

[Ebenau83] Robert G. Ebenau, A.S. Ackerman, M. S. Easterling, P. J. Fowler, P. Freeman, P. M. Mellema, and V. S. Whitis.
Report of the group on training as a technology transfer vehicle.
In *Proceedings, IEEE Computer Society Workshop on Software Engineering Technology Transfer*, pages 6-8. IEEE Computer Society Press, 1983.

[Fagan76] M. E. Fagan.
Design and code inspections to reduce errors in program development.
IBM Systems J. 15(3):182-211, July 1976.

[Feigenbaum83] Armand V. Feigenbaum.
Total Quality Control.
McGraw-Hill, 1983.

[Felix83] G.H. Felix and J.L. Riggs.
Productivity measurement by objectives.
National Productivity Rev. , Autumn 1983.

[Flaherty85] M.J. Flaherty.
Programming process productivity measurement system for System/370.
IBM Systems J. 24(2):168-175, 1985.

[Fortuna88] R.M. Fortuna.
Beyond quality: taking SPC upstream.
ASQC Quality Progress , June 1988.

[Fowler86] Priscilla J. Fowler.
In-process inspections of workproducts at AT&T.
AT&T Technical J. 65(2):102-112, March-April, 1986.

[Freeman87] Peter Freeman.
Software Perspectives: The System is the Message.
Addison-Wesley Publishing Company, 1987.

[Gale90] J.L. Gale, J.R. Tirso, and C.A. Burchfield.
Implementing the defect prevention process in the MVS interactive programming organization.
IBM Systems J. 29(1):33-43, 1990.

[Gardiner87] J.S. Gardiner and D.C. Montgomery.
Using statistical control charts for software quality control.
Quality and Reliability Engineering Intl. :15-20, January 1987.

[Gershon85] Gary Gershon (ed.).
Special issue on quality and producivity.
IBM Systems J. 24(2):74-175, 1985.

[Gessner84] Robert A. Gessner.
Manufacturing Information Systems Implementation Planning.
John Wiley & Sons, Inc., 1984.

[Grady87] Robert L. Grady and Deborah L. Caswell.
Software Metrics: Establishing a Company-Wide Program.
Prentice-Hall, 1987.

[Harrington87] H.J. Harrington.
The Improvement Process: How America's Leading Companies Improve Quality.
McGraw-Hill, 1987.

[Humphrey88] Watts Humphrey.
Characterizing the software process: a maturity framework.
IEEE Software :73-79, March 1988.

[Humphrey89] Watts Humphrey.
Managing the Software Process.
Addison-Wesley, 1989.

[IEEE-PMP88] IEEE.
Standard for Software Project Management Plans, ANSI/IEEE Std. 1058.1-1987.
Institute of Electrical and Electronics Engineers, 1988.

[IEEE-SLCP89] IEEE.
Draft Standard for Software Life Cycle Processes, P1074/D4.
Institute of Electrical and Electronics Engineers, August 8, 1989.

[IEEE-SQ86] IEEE.
Guide for Software Quality Planning, ANSI/IEEE Std. 983-1986.
Institute of Electrical and Electronics Engineers, 1986.

[Ishikawa85] Kaoru Ishikawa.
What is Total Quality Control? The Japanese Way.
Prentice-Hall, 1985.

[Jacobson89] John M. Jacobson.
Value-based contracting.
In *Proceedings, SOFTCON '89: Managing Software into the 90's ... Acquiring the Competitive Edge*, pages 189-203. American Defense Preparedness Association, 1989.

[Janger79] Allen R. Janger.
Matrix Organization of Complex Businesses.
The Conference Board, 1979.

[Juran81] J.M. Juran.
Product quality - a prescription for the West.
Management Rev. , June 1981.

[Kanter83] Rosabeth Moss Kanter.
The Change Masters: Innovation for Productivity in the American Corporation.
Simon and Schuster, 1983.

[Kellner89] Marc I. Kellner.
Software process modeling: value and experience.
SEI Technical Review :23-54, 1989.

[Kenkyusha54] Senkichiro Katsumata (editor).
Kenkyusha's New Japanese-English Dictionary.
Kenkyusha, 1954.

[Lawler85] E.E. Lawler III and Susan A. Mohrman.
Quality circles after the fad.
Harvard Bus. Rev. :65-71, January 1985.

[Leonard85] Dorothy Leonard-Barton.
Implementing new technology.
Harvard Bus. Rev. :102-110, November-December 1985.

[Leonard88] Dorothy Leonard-Barton.
Implementation as mutual adaptation of technology and organization.
Research Policy 17(5):251-267, October 1988.

[Lorange84] Peter Lorange and Declan Murphy.
Considerations in implementing strategic control.
J. of Bus. Strategy 4(4):27-35, Spring 1984.

[Martin88] Charles F. Martin.
Second-generation CASE tools: a challenge to vendors.
IEEE Software :46-49, March, 1988.

[Mason81] Richard O. Mason and Ian I. Mitroff.
Challenging Strategic Planning Assumptions - Theory, Cases and Techniques.
John Wiley & Sons, Inc., 1981.

[Mays90] R.G. Mays, C.L. Jones, G.J. Holloway, and D.P. Studinski.
Experiences with defect prevention.
IBM Systems J. 29(1):4-32, 1990.

[Morton83] Richard Morton (editor).
IEEE Computer Society Workshop on Software Engineering Technology Transfer.
IEEE Computer Society Press, 1983.

[Myers86] Andrew B. Myers (ed.).
Special issue on quality: theory and practice.
AT&T Technical J. 65(2):4-118, March/April 1986.

[ONeill80] D. O'Neill.
The management of software engineering, Part II: Software engineering program.
IBM Systems J. 19(4):421-431, 1980.

[Persico89] J. Persico, Jr.
Team up for quality improvement.
ASQC Quality Progress :33-37, January 1989.

[Pinto88] Jeffrey Pinto and John Prescott.
Variations in critical success factors over the stages in the project life cycle.
J. of Management 14(1):5-18, 1988.

[Pressman88] Roger S. Pressman.
Making Software Engineering Happen: A Guide for Instituting the Technology.
Prentice-Hall, 1988.

[Przybylinski88] Stan Przybylinski and Priscilla Fowler (editors).
Transferring Software Engineering Tool Technology.
The Computer Society of the IEEE, 1988.

[Quinn77] James Brian Quinn.
Strategic goals: process and politics.
Sloan Mgmt. Rev. 19(1):21-37, Fall 1977.

[Radice85] R.A. Radice, N.K. Roth, A.C. O'Hara, Jr., and W.A. Ciarfella.
A programming process architecture.
IBM Systems J. 24(2):79-90, 1985.

[Rogers83] Everett M. Rogers.
Diffusion of Innovation.
The Free Press, 1983.

[Scacchi87] Walt Scacchi.
SEI Curriculum Module SEI-CM-10-1.0.
Models of Software Evolution: Life Cycle and Process, Software Engineering Institute, October 1987.

[Schmidt87] Terry Schmidt and Merlyn Kettering.
Planning for Successful Project Implementation: Guidelines for the Project Team.
Technical Report, United States Department of Agriculture, 1987.

[Schultz87] Randall Schultz, Dennis Slevin, and Jeffrey Pinto.
Strategy and tactics in a process model of project implementation.
Interfaces 17(3):34-46, May-June 1987.

[Skrabec89] Q.R. Skrabec.
The transition from 100% inspection to process control.
ASQC Quality Progress :35-36, April 1989.

[SMAP4.3] *Management Control and Status Reports Documentation Standard and Data Item Descriptions.*
Release 4.3 edition, Office of Safety, Reliability, Maintainability, and Quality Assurance, Software Management and Assurance Program, National Aeronautics and Space Administration, Washington, DC, February 28, 1989.

[SoftProcWork89] Colin Tully.
Representing and enacting the software process.
In *Proceedings of the 4th International Software Process Workshop held at Moretonhampstead, Devon, UK, May 11-13, 1988*. Also available as ACM *Software Engineering Notes*, June 1989.

[Stein80] Barry A. Stein and Rosabeth Moss Kanter.
Building the parallel organization: Toward mechanisms for permanent quality of work life.
J. Applied Behavioral Science (16):371-88, July 1980.

[Thomas88] .
Most companies fail software test.
Advanced Military Computing 4(7), 1988.

[Tichy83] Noel Tichy.
The essentials of strategic change management.
J. of Bus. Strategy 3(4):55-67, Spring 1983.

[Tornatzky90] Louis G. Tornatzky and Mitchell Fleischer.
The Processes of Technological Innovation.
Lexington Books, 1990.

[TQM88] Department of Defense.
Total Quality Management Master Plan.
August 1988

[TR23] Watts Humphrey and William Sweet.
A Method for Assessing the Software Engineering Capability of Contractors.
Technical Report CMU/SEI-87-TR-23, ADA187230, Software Engineering Institute, September 1987.

[TR7] Timothy G. Olson, Watts Humphrey, and David Kitson.
Conducting SEI-Assisted Software Process Assessments.
Technical Report CMU/SEI-89-TR-7, ADA219065, Software Engineering Institute, February 1989.

[Trainor89] Michael Sullivan-Trainor.
IBM's Ford suggests optimum IS spending.
Computerworld (23):17, March 27, 1989.

[Turner78] W.C. Turner, J.H. Mize, and K.E. Case.
Introduction to Industrial and Systems Engineering.
Prentice-Hall, 1978.

[Weinberg71] Gerald M. Weinberg.
The Psychology of Computer Programming.
Van Nostrand Reinhold, 1971.

[Weisbord87] Marvin Weisbord.
Productive Workplaces: Organizing and Managing for Dignity, Meaning, and Community.
Jossey-Bass, 1987.

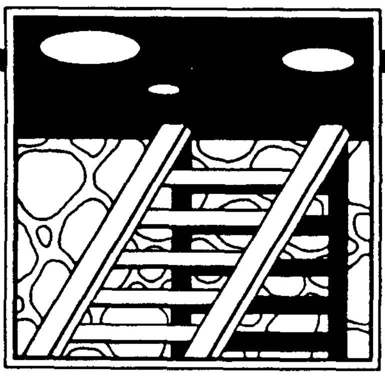
[Willborn88] W. Willborn.
Registration of quality programs.
ASQC Quality Progress :56-68, September 1988.

Appendix A: Characterizing the Software Process: A Maturity Framework

This is a reprint of [Humphrey88].

Characterizing the Software Process: A Maturity Framework

Watts S. Humphrey
Software Engineering Institute



Characterizing the Software Process: A Maturity Framework

Watts S. Humphrey, *Software Engineering Institute*

Software Quality and productivity must improve. But where to start? This model helps organizations identify their highest priority problems and start making improvements.

The amount of money spent on software in the US grows approximately 12 percent each year, and the demand for added software functions grows even faster. Software is a major and increasing portion of US Defense Dept. procurement costs, and software often adversely affects the schedules and effectiveness of weapons systems.

In recognition of the need to improve the development of military software, the Defense Dept. has launched several initiatives on software reliability, maintainability, and testing, including the Ada Joint Program Office and the STARS program. The Defense Dept. formed the Software Engineering Institute at Carnegie Mellon University in 1984 to establish standards of excellence for software engineering and to accelerate the transition of advanced technology and methods into practice.

One SEI project is to provide the Defense Dept. with some way to characterize the capabilities of software-development organizations. The result is this software-process maturity framework, which can be used by any software organization to assess its own capabilities and identify the most important areas for improvement.

Ideal Software process

It is worthwhile to examine the characteristics of a truly effective software process. First, it is predictable: Cost estimates and schedule commitments are met with reasonable consistency and the quality of the resulting products generally meet user needs.

Statistical control. The basic principle of software process management is that if

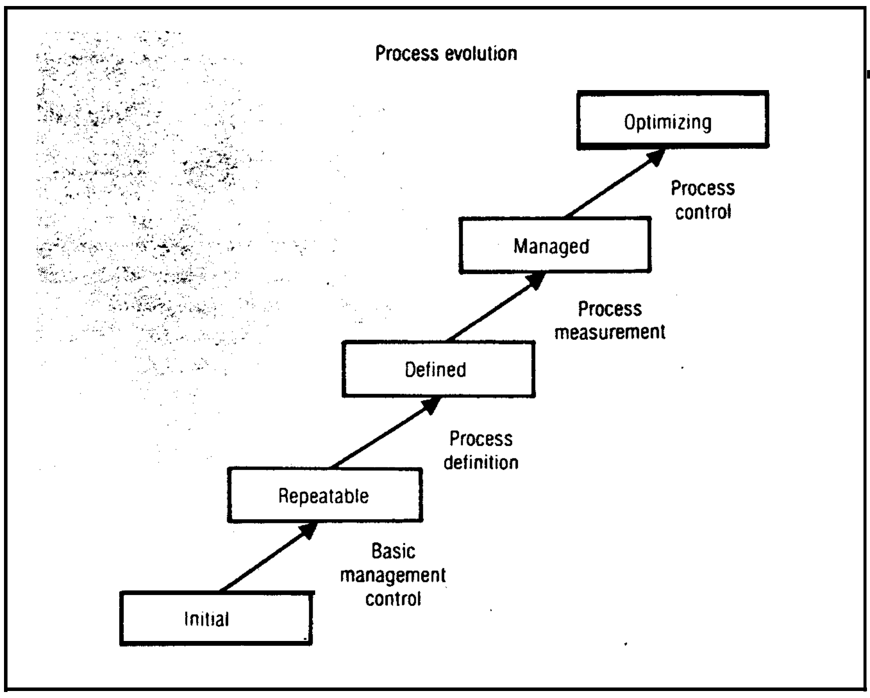


Figure 1. The five levels of process maturity.

the development process is under statistical control, a consistently better result can be achieved only by improving the process. If the process is not under statistical control, sustained progress is not possible until it is.'

When a process is under statistical control, repeating the work iv roughly the same way will produce roughly the same result.

W.E. Deming, in his work with the Japanese industry after World War II, applied the concepts of statistical process control to industry.' While there are important differences, these concepts are just as applicable to software as they are to automobiles, cameras, wristwatches, and steel. A software-development process that is under statistical control will produce the desired results within the anticipated limits of cost, schedule, and quality.

Measurement. The basic principle behind statistical control is measurement. As Lord Kelvin said a century ago, " . . . when you can measure what you are speaking about. and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely in your thoughts advanced to the stage of science. . . . "[2]

There are several factors to consider in measuring the programming process. Perhaps most important is that the mere act of measuring human processes changes them. Since people's fears and motivations are involved, the results must be viewed in a different light than data on natural phenomena.

It is also essential to limit the measurements to those few items that will really be used. Measurements are both expensive and disruptive; overzealous measuring can degrade the processes we are trying to improve.

Development-process improvement

An important first step in addressing software problems is to treat the entire development task as a process that can be controlled, measured, and improved. We define a process as a sequence of tasks that, when properly performed, produces the desired result. Clearly, a fully effective software process must consider the relationships of all the required tasks, the tools and methods used, and the skill, training, and motivation of the people involved.

To improve their software capabilities, organizations must take five steps:

(1) understand the current status of their development process or processes,

(2) develop a vision of the desired process,

(3) establish a list of required process improvement actions in order of priority,

(4) produce a plan to accomplish these actions, and

(5) commit the resources to execute the plan.

The maturity framework developed at the SEI addresses these five steps by characterizing a software process into one of five maturity levels. By establishing their organization's position in this maturity structure, software professionals and management can more readily identify those areas where improvement actions are most likely to produce results.

Process maturity levels

As Figure 1 shows, the five levels of process maturity are:

1. Initial. Until the process is under statistical control, no orderly progress in process improvement is possible.

2. Repeatable. The organization has achieved a stable process with a repeatable level of statistical control by initiating rigorous project management of commitments, cost, schedule, and changes.

3. Defined. The organization has defined the process, to ensure consistent implementation and provide a basis for better understanding of the process. At this point, advanced technology can usefully be introduced.

4. Managed. The organization has initiated comprehensive process measurements, beyond those of cost and schedule performance. This is when the most significant quality improvements begin.

5. Optimizing. The organization now has a foundation for continued improvement and optimization of the process.

These levels have been selected because they

*reasonably represent theactual historical phases of evolutionary improvement of real software organizations,

- represent a measure of improvement that is reasonable IO achieve from 1 he prior level,
- suggest interim improvement goals and progress measures. and
- make obvious a set of immediate improvement priorities, once an organization's status in this framework is known.

While there are many other elements to these maturity-level transitions, the basic objective is to achieve a controlled and measured process as the scientific foundation for continuous improvement. This structure is intended to be used with an assessment and management methodology, as outlined in the box on pp. 76-77.

Initial Process

The Initial Process could properly be called ad hoc, and it is often even chaotic. Here, the organization typically operates without formalizied procedures, cost estimates, and project plans. Tools are neither well integrated with the process nor uniformly applied. Change control is lax and there is little senior management exposure to or understanding of the problems and issues. Since problems are often deferred or even forgotten, software installation and maintenance often present serious problems.

While organizations at this level may have formal procedures for project control, there is no management mechanism to ensure they are used. The best test is to observe how such an organization behaves in a crisis. If it abandons established procedures and reverts to merely coding and testing, it is likely to be at the Initial Process level. After all, if the techniques and methods are appropriate. they must be used in a crisis and if they are not appropriate, they should not be used at all.

One reason organizations behave chaotically is that they have not gained sufficient experience to understand the consequences of such behavior. Because many effective software actions such as design and code reviews or test data analysis do not appear to directly support shipping the product, they seem expendable.

It is much like driving an automobile. Few drivers with any experience will continue driving for very long when the engine warning light comes on, regardless of their rush. Similarly, most drivers starting on a new journey will, regardless of their hurry, pause to consult a map. They have learned the difference between speed and progress.

In software, coding and testing seem like progress, but they are often only wheel-spinning. While they must be done, there is always the danger of going in the wrong direction. Without a sound plan and a thoughtful analysis of the problems, there is no way to know.

Organizations at the Initial Process level can improve their performance by instituting basic project controls. The most important are:

- Project management. The fundamental role of a project-management system is to ensure effective control of commitments. This requires adequate preparation, clear responsibility, a public declaration, and a dedication to performance.[3]

For software, this starts with an understanding of the job's magnitude. In any but the simplest projects, a plan must then 'be developed to determine the best schedule and the resources required. In the absence of such an orderly plan, no commitment can be better than an educated guess.

A disciplined software development organization must have senior management oversight.

- Management oversight. A disciplined software-development organization must have senior management oversight. This includes review and approval of all major development plans before official commitment.

Also, a quarterly review should be conducted of facility-wide process compliance, installed-quality performance, schedule tracking, cost trends, computing service. and quality and productivity goals by project. The lack of such reviews typically results in uneven and generally inadequate implementation of the process as well as in frequent overcommitments and cost surprises.

- Quality assurance. A quality-assurance group is charged with assuring management that the software-development work is actually done the way it is supposed to be done. To be effective, the assurance organization must have an independent reporting line to senior management and sufficient resources to monitor performance of ail key planning, implementation, and verification activities. This generally requires an organization of about 5 to 6 percent the size of the development organization.

- Change control. Control of changes in software development is fundamental to business and financial control as well as to technical stability. To develop quality software on a predictable schedule, the requirements must be established and maintained with reasonable stability throughout the development cycle. Changes will have to be made, but they must be managed and introduced in an orderly way.

While occasional requirements changes are needed, historical evidence demonstrates that many of them can be deferred and phased in later. If all changes are not controlled, orderly design, implementation, and testing is impossible and no quality plan can be effective.

Repeatable Process

The Repeatable Process has one important strength over the Initial Process: It provides commitment control.

This is such an enormous advance over the Initial Process that the people in the organization tend to believe they have mastered the software problem. They do not realize that their strength stems from their prior experience at similar work. Organizations at the Repeatable Process level thus face major risks when they are presented with new challenges.

Examples of the changes that represent the highest risk at this level are:

- New tools and methods will likely affect how the process is performed, thus destroying the relevance of the intuitive historical base on which the organization relies. Without a defined process framework in which to address these risks, it is even possible for a new technology to do more harm than good.
- When the organization must develop a new kind of product, it is entering new territory. For example, a software group that has experience developing compilers will likely have design, scheduling, and estimating problems if assigned to write a control program. Similarly, a group that has developed small, self-contained programs will not understand the interface and integration issues involved in large-scale projects. These changes again

destroy the relevance of the intuitive historical basis for the organization's work.

l Major organization changes can be highly disruptive. In the Repeatable Process organization, a new manager has no orderly basis for understanding what is going on and new team members must learn the ropes through word of mouth.

The key actions required to advance from the Repeatable Process to the Defined Process are:

1. Establish a process group. A process group is a technical group that focuses exclusively on improving the software-development process. In most software organizations, people are entirely devoted to product work. Until someone is given a full-time assignment to work on the process, little orderly progress can be made in improving it.

The responsibilities of process groups include defining the development process, identifying technology needs and opportunities, advising the projects, and conducting quarterly management reviews of process status and performance. Typically, the process group should be about 1 to 3 percent the size of the development organization. Because of the need for a nucleus of skills, groups smaller than about four are unlikely to be fully effective. Small organizations that lack the experience base to form a process group should address these issues through specially formed committees of experienced professionals or by retaining consultants.

2. Establish a software-development process architecture that describes the technical and management activities required for proper execution of the development process.' The architecture is a structural decomposition of the development cycle into tasks, each of which has entry criteria, functional descriptions, verification procedures, and exit criteria. The decomposition continues until each defined task is performed by an individual or single management unit.

3. If they are not already in place, introduce a family of software-engineering methods and technologies. These include design and code inspections, formal design methods, library-control systems, and comprehensive testing methods. Prototyping should also be considered. along with the adoption of modern implementation languages.

Defined Process

With the Defined Process, the organization has achieved the foundation for major and continuing progress. For example, the development group, when faced with a crisis, will likely continue to use the Defined Process. The foundation has now been established for examining the process and deciding how to improve it.

As powerful as the Defined Process is, it is still only qualitative: There is little data to indicate what is going on or how effective the process really is. There is considerable debate about the value of software-process measurements and the best ones to use. This uncertainty generally stems from a lack of process definition and the consequent confusion about the specific items to be measured. With a defined process, we can focus the measurements on specific tasks. The process architecture is thus an essential prerequisite to effective measurement.

The key steps[3,4] to advance to the Managed Process are:

1. Establish a minimum, basic set of process measurements to identify the quality and cost parameters of each process step. The objective is to quantify the relative costs and benefits of each major pro-

How to use this framework

This process-maturity structure is intended to be used with an assessment methodology and a management system.'.'

Assessment lets you identify the organization's specific maturity status. A management system establishes a structure for actually implementing the priority actions necessary to improve the organization. Once its position in this maturity structure is defined, the organization can concentrate on those items that will let it advance to the next level.

When, for example, a software organization does not have an effective project-planning system, it may be difficult or even impossible to introduce advanced methods and technology. Poor project planning generally leads to unrealistic schedules, inadequate resources, and frequent crises. In such circumstances, new methods are usually ignored, and the focus is on coding and testing.

Using this maturity framework, the SEI has developed an assessment questionnaire and methodology, a portion of which is shown in Figure A.[4,5] The questionnaire has been reviewed by more than 400 governmental and industrial organizations. Also, it has been completed by more than 50 programming professionals from nearly as many software organizations. A section of our questionsionals from nearly as many software organizations.

The SEI has also used the assessment methodology to conduct in-depth technical reviews of 25 programming projects in four large programming organizations.

Through this work, the assessment methodology and questionnaire have evolved, but the five-level maturity framework has remained essentially unchanged. We have found that it portrays, with reasonable accuracy, the status and problems as seen by the managers and professionals in the organizations reviewed.

These early results indicate that the model reasonably represents the state of such organizations and providesa mechanism to rapidly identify the key improvement issues they face. At this time, the data is loo limited to provide any more detailed information as to maturity distribution by industry, organization size, or type of work.

References

1. W.S. Humphrey, *Managing for Innovation - Leading Technical People,* Prentice-Hall. Englewood Cliffs, N.J.. 1987.
2. R.A. Radice et al., "A Programming Process Study." IBM Systems J.. Vol. *24. No. 2.* 1985. pp. 91-101.
3. R.A. Radice et al.. "A Programming Process Architecture," ISM Systems J., Vol. 24. No. *2.1985.* pp. 7940.

cess activity, such as the cost and yield of error detection and correction methods.

2. Establish a process database with the resources to manage and maintain it. Cost and yield data should be maintained centrally to guard against loss, to make it available for all projects, and to facilitate process quality and productivity analysis.

3. Provide sufficient process resources to gather and maintain this data and to advise project members on its use. Assign skilled professionals to monitor the quality of the data before entry in the database and to provide guidance on analysis methods and interpretation.

4. Assess the relative quality of each product and inform management where quality targets are not being met. An independent quality-assurance group should assess the quality actions of each project and track its progress against its quality plan. When this progress is compared with the historical experience on similar projects, an informed assessment generally can be made.

Managed Process

In advancing from the Initial Process via the Repeatable and Defined Processes to the Managed Process, software organizations typically will experience substantial quality improvements. The greatest potential problem with the Managed Process is the cost of gathering data. There are an enormous number of potentially valuable measures of software development and support, but such data is expensive to gather and maintain.

Therefore, approach data gathering with care and precisely define each piece of data in advance. Productivity data is generally meaningless unless explicitly defined. For example, the simple measure of lines of source code per development month can vary by 100 times of more, depending on the interpretation of the parameters. The code count could include only new and changed code or all shipped instructions. For modified programs, this can cause a ten-times variation. Similarly, you can use noncomment, nonblank lines, executable instructions, or equivalent assembler instructions, with variations again of up to seven times.[5] Management, test, documentation, and support personnel may or may not be counted when calculating labor months expended. Again, the variations can run at least as high as seven times.[6]

When different groups gather data but do not use identical definitions, the results are not comparable, even if it made sense tocompare them. The tendency with such data is to use it to compare several groups and put pressureon those with the lowest ranking. This is a misapplication of process data.

First, it is rare that two projects are comparable by any simple measures. The variations in task complexity caused by different product types can exceed five to one. Similarly, the cost per line of code of small modifications is often two to three times that for new programs. The degree of requirements change can make an enormous difference, as can the design status of the base program in the case of enhancements.

Process data must not be used to compare projects or individuals. Its purpose is to illuminate the product being developed and to provide an informed basis for improving the process. When such data is used by management to evaluate individuals or teams, the reliability of the data itself will deteriorate. The US Constitution's Fifth Amendment, which protects against self-incrimination. is based on sound principles: Few people can be counted on to provide reliable data on

4. W.S. Humphrey and D.H. Kitson. "Preliminary Report on Conducting SEI-Assisted Assessments of Software Engineering Capability," Tech. Report SEI-87-TR-16, Software Eng. Inst.. Pittsburgh, July 1987.
5. W.S. Humphrey and W.L. Sweet, "A Method for Assessing the Software Engineering Capability of Contractors," Tech. Report SEI-867-TR-23, Software Eng. inst., Pittsburgh, Sept. 1987.

2.3. Data Management and Analysis

Data management deals with the gathering and retention of process metrics. Data management requires standardized data definitions, data management facilities, and a staff to ensure that data is promptly obtained, properly checked, accurately entered into the database, and effectively managed.

Analysis deals with the subsequent manipulation of the process data to answer questions such as, "Is there is a relatively high correlation between error densities found in test and those found in use?" Other types of analyses can assist in determining the optimum use of reviews and resources, the tools most needed, testing priorities. and needed education.

2.3.1. Has a managed and controlled process database been established for process metrics data across all projects?

2.3.2. Are the review data gathered during design reviews analyzed?

2.3.3. Is the error data from code reviews and tests analyzed to determine the likely distribution and characteristics of the errors remaining in the product?

2.3.4. Are analyses of errors conducted to determine their process related causes?

2.3.5. Is a mechanism used for error cause analysis?

2.3.6. Are the error causes reviewed to determine the process changes required to prevent them?

2.3.7. Is a mechanism used for initiating error prevention actions?

2.3.8. Is review efficiency analyzed for each project?

2.3.9. Is software productivity analyzed for major process steps?

Figure A. A portion of the SEI's assessment questionnaire.

their own performance.

The two fundamental requirements to advance from the Managed Process to the Optimizing Process are:

1. Support automatic gathering of process data. Some data cannot be gathered by hand, and all manually gathered data is subject to error and omission.
2. Use this data to both analyze and modify the process to prevent problems and improve efficiency.

Optimizing Process

In varying degrees, process optimization goes on at all levels of process maturity. With the step from the Managed to the Optimizing Process, however, there is a paradigm shift. Up to this point, software-development managers have largely focused on their products and will typically only gather and analyze data that directly relates to product improvement. In the Optimizing Process, the data is available to actually tune the process itself. With a little experience, management will soon see that process optimization can produce major quality and productivity improvements.

For example, many errors can be identified and fixed far more economically by code inspections than through testing. Unfortunately, there is little published data on the costs of finding and fixing errors.' However, I have developed a useful rule of thumb from experience: It takes about one to four working hours to find and fix a bug through inspections and about 15 to 20 working hours to find and fix a bug in function or system test. It is thus clear that testing is not a cost-effective way to find and fix most bugs.

However, some kinds of errors are either uneconomical or almost impossible to find except by machine. Examples are errors involving spelling and syntax, interfaces, performance, human factors, and error recovery. It would thus be unwise to eliminate testing completely because it provides a useful check against human frailties.

The data that is available with the Optimizing Process gives us a new perspective on testing. For most projects, a little analysis shows that there are two distinct activities involved. The first is the removal of bugs. To reduce this cost, inspections should be emphasized together with any other cost-effective techniques. The role of functional and system testing should then be changed to one of finding symptoms that are further explored to see if the bug is an isolated problem or if it indicates design problems that require more comprehensive analysis.

In the Optimizing Process, the organization has the means to identify the weakest elements of the process and fix them. At this point in process improvement, data is available to justify the application of technology to various critical tasks and numerical evidence is available on the effectiveness with which the process has been applied to any given product. We no longer need reams of paper to describe what is happening because simple yield curves and statistical plots provide clear and concise indicators. It is now possible to assure the process and hence have confidence in the quality of the resulting products.

People in the process. Any software-development process is dependent on the quality of the people who implement it. Even with the best people, however, there is always a limit to what they can accomplish. When engineers are already working 50 to 60 hours a week, it is hard to see how they could handle the vastly greater challenges of the future.

The Optimizing Process helps in several ways:

l It helps managers understand where help is needed and how best to provide the people with the support they require.

l It lets professionals communicate in concise, quantitative terms. This facilitates the transfer of knowledge and minimizes the likelihood of their wasting time on problems that have already been solved.

l It provides' the framework for the professionals to understand their work performance and to see how to improve it. This results in a highly professional environment and substantial productivity benefits, and it avoids the enormous amount of effort that is generally expended in fixing and patching other people's mistakes.

The Optimizing Process provides a disciplined environment for professional work. Process discipline must be handled with care, however, for it can easily become regimentation. The difference between a disciplined environment and a regimented one is that discipline controls the environment and methods to specific standards while regimentation defines the actual conduct of the work.

Discipline is required in large software projects to ensure, for example, that the people involved use the same conventions, don't damage each other's products, and properly synchronize their work. Discipline thus enables creativity by freeing the most talented software professionals from the many crises that others have created.

The need. There are many examples of disasters caused by software problems, ranging from expensive missile aborts to enormous financial losses. As the computerization of our society continues, the public risks due to poor-quality code will become untenable. Not only are our systems being used in increasingly sensitive applications, but they are also becoming much larger and more complex.

While proper questions can be raised about the size and complexity of current systems, they are human creations and they will, alas, continue to be produced by humans - with all their failings and creative talents. While many of the currently promising technologies will undoubtedly help, there is an enormous backlog of needed functions that will inevitably translate into vast amounts of code.

More code means increased risk of error and, when coupled with more complexity, these systems will become progressively less testable. The risks will thus increase astronomically as we become more efficient at producing prodigious amounts of new code.

As well as being a management issue, quality is an economic one. It is always possible to do more inspections or to run more tests, but it costs time and money to do so. It is only with the Optimizing Process that the data is available to understand the costs and benefits of such work. The Optimizing Process thus provides the foundation for significant advances in software quality and simultaneous improvements in productivity.

There is little data on how long it takes for software organizations to advance through these maturity levels toward the Optimizing Process. Based on my experience, transition from level 1 to level 2 or from level 2 to level 3 take from one to three years, even with a dedicated management commitment to process. improvement. To date, no complete organizations have been observed at levels 4 or 5.

To meet society's needs for increased system functions while simultaneously addressing the problems of quality and productivity, software managers and professionals must establish the goal of moving to the Optimizing Process.

This software-development process-maturity model reasonably represents the actual ways in which software-development organizations improve. It provides a framework for assessing these organizations and identifying the priority areas for immediate improvement. It also helps identify those places where advanced technology can be most valuable in improving the software-development process.

The SEI is using this model as a foundation for a continuing program of assessments and software process development. These assessment methods have been made public,[8,9] and preliminary data is now available from several dozen software organizations.

Figure 2 shows the maturity distribution of these organizations and the three leading problems faced at each level. At level one, the distribution is shown by quartile. There is not yet sufficient data to provide this detail for levels 2 or 3. As further data is gathered, additional reports will be published on the results obtained.

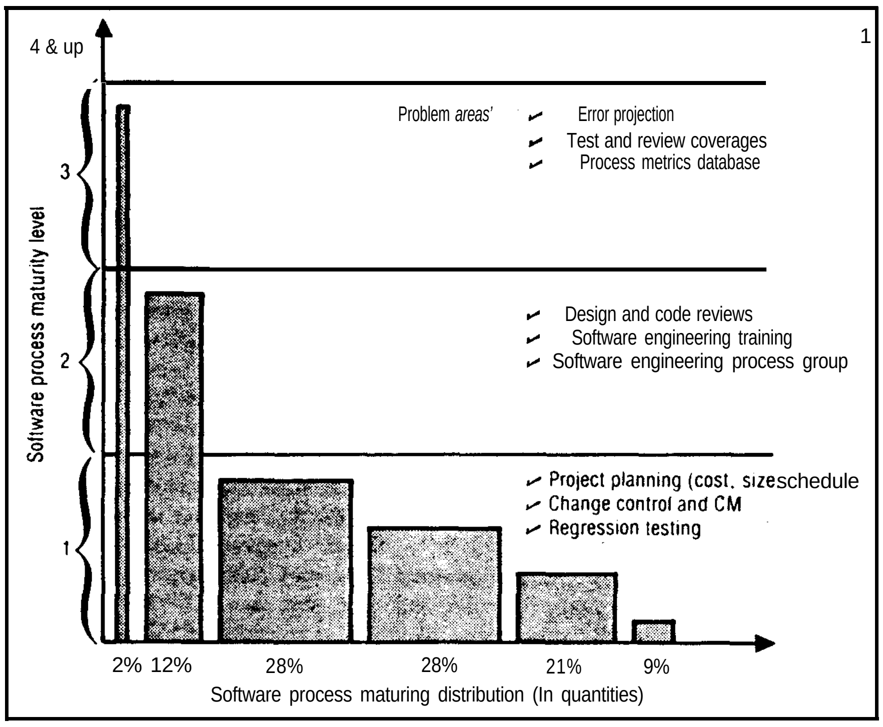


Figure 2. Early results from several dozen software organizations queried by the SEI shows the maturity distribution and the three leading problems faced at each level. At level one, the distribution is shown by quartile. There is not yet sufficient data to provide this detail for levels 2 or 3. To date, no complete organizations have been observed at levels 4 or 5.

Acknowledgments

Much of the early work on software process maturity was suggested by my former colleagues at IBM. I am particularly indebted to Ron Radice and Jack Harding for their insights and support. In addition, William Sweet of the SEI and Martin Owens and Herman Schultz of Mitre Corp. have made valuable contributions to this work. I am also indebted to my colleagues at the SEI, particularly Rodger Blair, Larry Druffel. and Greg Hansen, for their helpful comments and suggestions. This work was supported by the Defense Dept.

References

I. W.E. Deming, "Quality, Productivity, and Competitive Position," tech. report, MIT Center for Advanced Eng. Study, Cambridge, Mass., 1982.

2. J.R. Dunham and E. Kruesi, "The Measurement Task Area," *Computer, Nov.* 1983. pp. 41-54.

3. W.S. Humphrey, *Managing for Innovation: Leading Technical People,* Prentice-Hall, Englewood Cliffs, N.J., 1987.

4. R.A. Radice et al., "A Programming Process Architecture," *IBM Systems J.,* Vol. 24, No. 2. 1985. pp. 79-90.

5. M.L. Shooman. *Software Engineering: Design, Reliability, and Management,* McGraw-Hill. New York, 1983.

6. R.W. Wolverton. "The Cost of Developing Large-Scale Software," *IEEE Trans. Computers,* June 1974, pp 615-636.

7. M.L. Shooman and M.I. Bolsky, "Types, Distribution, and Test and Correction Times for Programming Errors," Proc. *Int'l Conf. Reliable Software,* IEEE. New York, 1975, pp. 347-357.

8. W.S. Humphrey and D.H. Kitson, "Preliminary Report on Conducting SEI-Assisted Assessments of Software-Engineering Capability," Tech. Report SEI-87-TR-16. Software Eng. Inst., Pittsburgh, July 1987.

9. W.S. Humphrey and W.L. Sweet. "A Method for Assessing the Software Engineering Capability of Contractors," Tech. Report SEI-87-TR-23, Software Eng. Inst.. Pittsburgh, Sept. 1987.

Watts S. Humphrey is director of the software process program for the Software Engineering Institute. This group provides leadership in establishing advanced software engineering processes, metrics, methods, and quality programs for the US government and its contractors.

He worked at IBM from 1959 to 1986, where he was director of programming quality and process. Humphrey has written two books, *Managing for Innovation: Leading Technical People* and *Switching Circuits with Computer Applications.*

Humphrey received a BS in physics from the University of Chicago, an MS in physics from the Illinois Institute of Technology, and an MBA from the University of Chicago. He has taught graduate electrical engineering at Northeastern University. An IEEE Fellow, he is also a member of the ACM.

Questions about this article can be addressed to Humphrey at the SEI, Carnegie Mellon University, Pittsburgh, PA 15213.

Appendix B: The Quality Movement and Software Engineering Process Improvement

by David N. Card
Computer Sciences Corporation

In recent years, concern for quality has become a national and international movement. Today, quality is a key factor in international competition. Throughout government and industry, new policies, audit programs, and the offering of prizes promote quality improvement actions. While this movement first became popular in Japan, many of the underlying concepts and implementing techniques were developed in the United States.

In the United States, the Department of Commerce and NASA give major awards to companies that demonstrate significant improvements and a commitment to quality. Japan has offered the Deming Prize for many years. The United Kingdom requires quality programs to be audited and certified [Willborn88]. Italy has also made moves in this direction.

In recognition of the increasing importance of quality, the Department of Defense recently developed a Total Quality Management (TQM) Policy [TQM88] that aims to achieve continuous improvement of products and services acquired for and produced by DoD. TQM practices will be inserted gradually into agency and contractor organizations on a seven-year schedule. In addition, the SEI has developed software capability evaluation criteria [TR23] to provide a basis for assessing software engineering organizations.

The remainder of this appendix discusses how the quality movement and TQM apply to software engineering. The appendix summarizes basic quality improvement strategies and identifies milestones in quality technology. Most of the discussion deals with the process management approach to quality improvement and the software quality technology required to apply that approach to software engineering.

B.1. Quality Technology Milestones

The milestones in the development of quality technology in manufacturing are widely recognized, as shown in Figure B-1. Product inspection began soon after the introduction of the assembly line (c. 1920). Japan started to adopt statistical methods of process control about 1960, when it embarked on a major effort to penetrate the world electronics market. Since then, most manufacturers have been driven to adopt similar methods. Process control requires understanding the capability of a process and setting control limits [Ishikawa85] for key process variables. More recently (c. 1980), attention has shifted to improving the underlying process and product designs.

The new quality technology has supplemented rather than replaced older quality technology. During this 70-year period, the key innovation was the shift from the product to the process as the focus for quality control and improvement. The transition from a product-centered to a process-centered view of quality is not easy. Experience from manufacturing indicates that it takes 8 to 10 years for an organization to move from product inspection to process control [Skrabec89]. Changing organizational behavior takes time, as recognized in the 7-year schedule of the TQM Master Plan [TQM88].

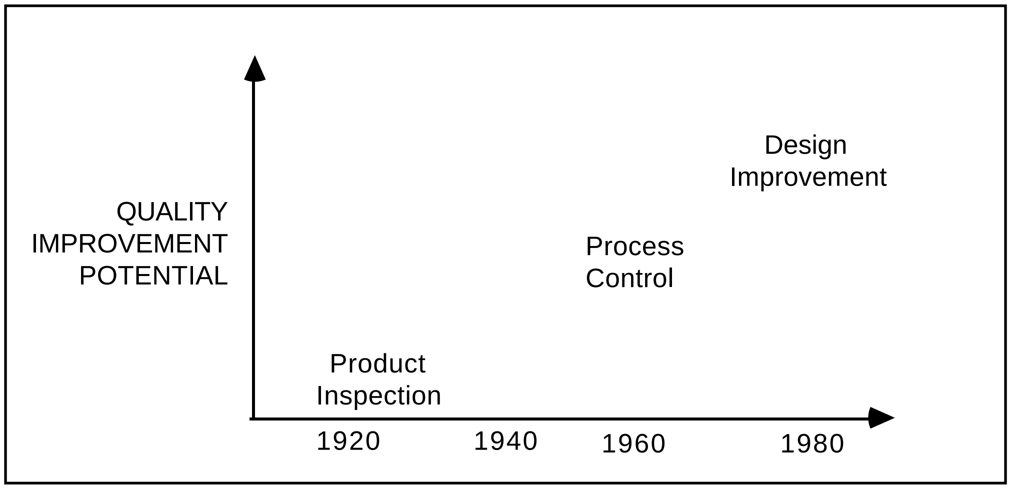


Figure B-1: The Evolution of Quality Assurance Technology
Adapted from [Fortuna88].

B.2. Quality Improvement Strategies

The significance and advantage of the process-centered approach to quality improvement can be seen by comparing it with two other commonly practiced approaches: exhortation and management by objectives. Understanding the deficiencies and limitations of these approaches may help to overcome the drawbacks associated with using them.

Exhortation relies on slogans, posters, buttons, and speeches to encourage workers to perform better. Usually, the notion of "better performance" is not well-defined. Moreover, management typically has no objective measure of the quality actually achieved. This approach is inexpensive to implement and does not require much technical expertise to operate.

Management by objectives involves identifying specific productivity and quality indicators. Management then sets improvement objectives for workers to achieve (for example, the objective matrix in [Felix83]). The measures are not tied to any specific process definition. Consequently, it is not always clear whether an improvement has occurred, or whether costs and problems have simply been shifted to an activity that is not measured. To compensate for this, several measures may be used simultaneously. More importantly, setting numerical objectives, although it provides an incentive, does not help ascertain how to make improvements, which is the hardest part of the task.

In contrast, process management provides a systematic approach to controlling and improving quality. Studying the process and analyzing its performance does help to develop improvement strategies. Juran recommends four steps [Juran81]:

1. Study the symptoms of poor quality (defects and failures).
2. Develop a theory or explanation of the cause of poor quality.
3. Test the theory in production to establish the cause.
4. Implement corrective or improvement action.

Deming recommends a 14-point organization-wide approach [Deming82]. The DoD TQM initiative is based on Deming's principles. The first and most important point of the Deming approach is long-term management commitment to quality improvement. Deming recognizes two types of causes of poor quality. *Special causes* show up as failures in the established process; the process is not working as intended and needs to be tuned or repaired because performance is less than expected. *Common causes* are basic weaknesses in the underlying process. Properly trained workers can identify and correct special causes. Only management has the authority to make the basic changes necessary to deal with common causes.

Both types of problems can be addressed with the same approach: the Shewart plan-do-check-act cycle shown in Figure B-2 [Deming82]. The cycle shows obvious similarity to Juran's four steps. The key message from Juran, Deming, and others in the quality movement is that long-term improvement results only from systematic study and action, not from slogans or arbitrary objectives.

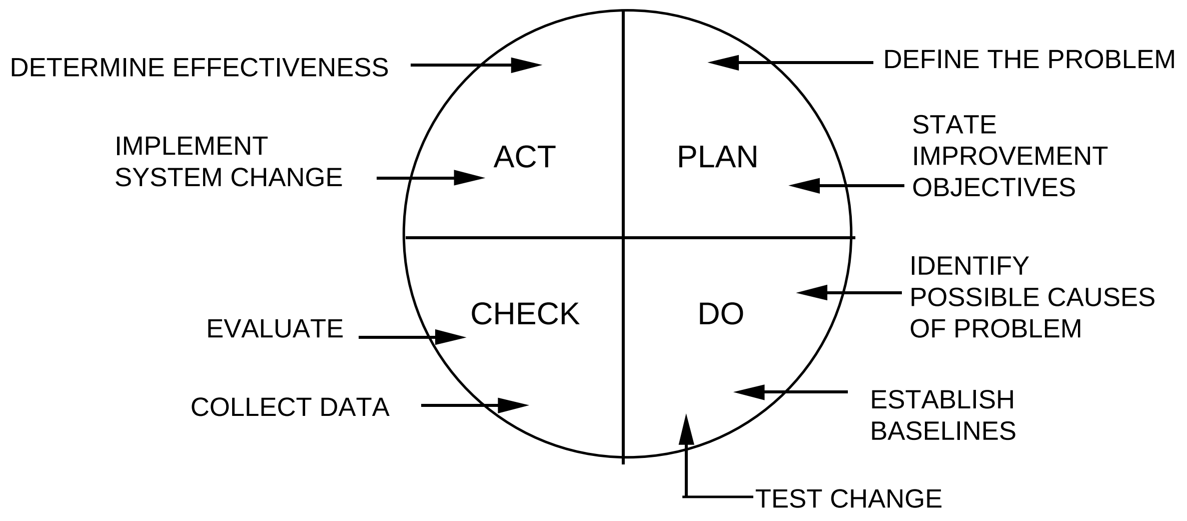
B.3. Process Management Approach

The process management approach includes three essential activities: process definition, process control, and process improvement. An undefined process cannot be controlled. An uncontrolled process cannot be improved consistently. Because improvement means change, attempting to improve an unstable process often leads to further instability. Figure B-3 shows how these elements are connected by the common threads of performance data and corrective action.

The process definition activity provides a prescription for performing work. Measuring the initial process performance establishes a baseline against which subsequent performance can be compared. The process control activity is concerned with identifying and correcting *special causes* of poor quality, to keep the process performing as intended. That is, it seeks to maintain key quality parameters within pre-defined control limits. The process improvement activity seeks to identify and rectify *common causes* of poor quality by making basic changes in the underlying process.

Each of these activities requires different skills and may be performed by different organizational elements. For example, the quality assurance organization may be charged with process control while a separate quality improvement team (such as a process group) deals with process improvement.

MANAGEMENT'S NEW JOB



- •Identify opportunities for improvement
- •Develop plan for improvement
- •Take corrective action on common causes
- •Pursue continuous improvement

Figure B-2: The Shewart Plan-Do-Check-Act Cycle

The three activities of process management can be mapped to the SEI process maturity levels (see Appendix A). The process definition step coincides with the Defined level, the process control step corresponds to the Managed level, and the process improvement step matches the Optimizing level.

From Figure B-3, one can get the impression that the process management approach appears to ignore the software product. In fact, product quality is an integral part of process management; the software engineering process is defined around work products—they are the interfaces between activities. Moreover, process performance is monitored in part by looking at products. Examining the software quality functions in a mature process will clarify this connection.

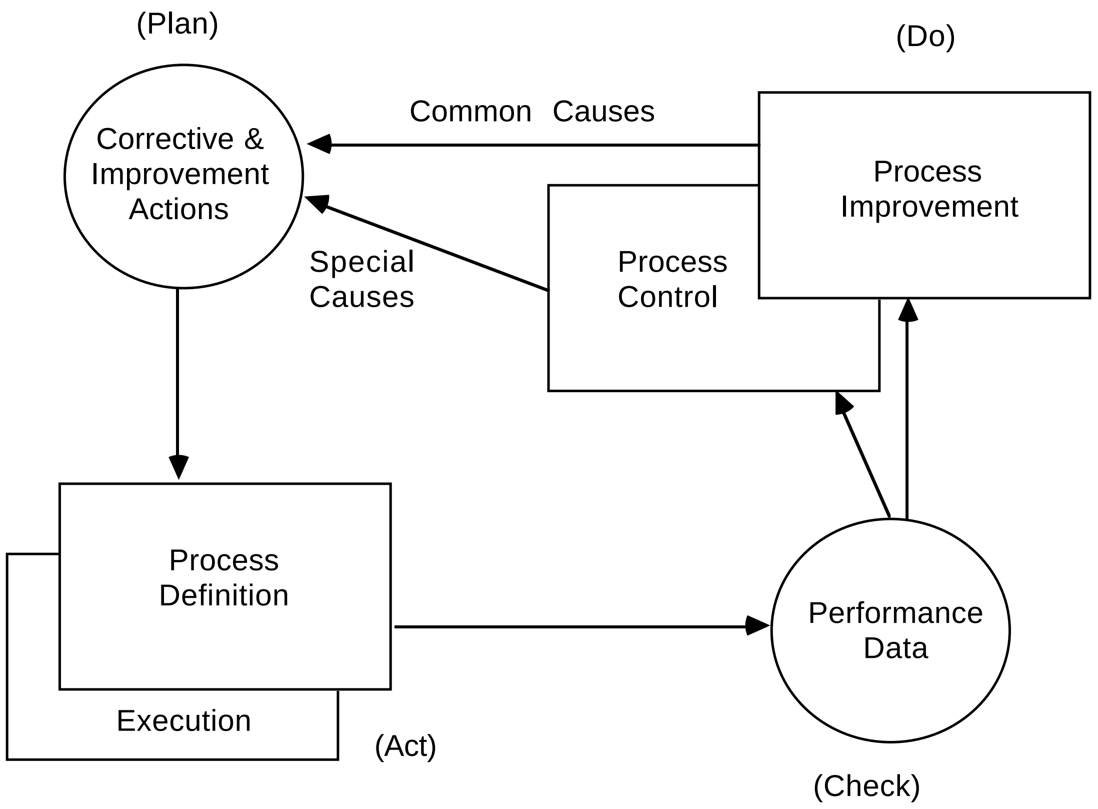


Figure B-3: Process Management Approach

B.4. Software Quality Functions

Combining the historical view of quality technology with the process management approach to quality improvement suggests that a fully mature software engineering process includes seven quality functions (see Figure B-4):

1. Process definition
2. Product inspections
3. Process audits
4. Software quality control
5. Process quality control
6. Software design improvement
7. Process design improvement

These functions affect both process quality and product quality, and responsibility for them must be distributed throughout the software enterprise.[12]

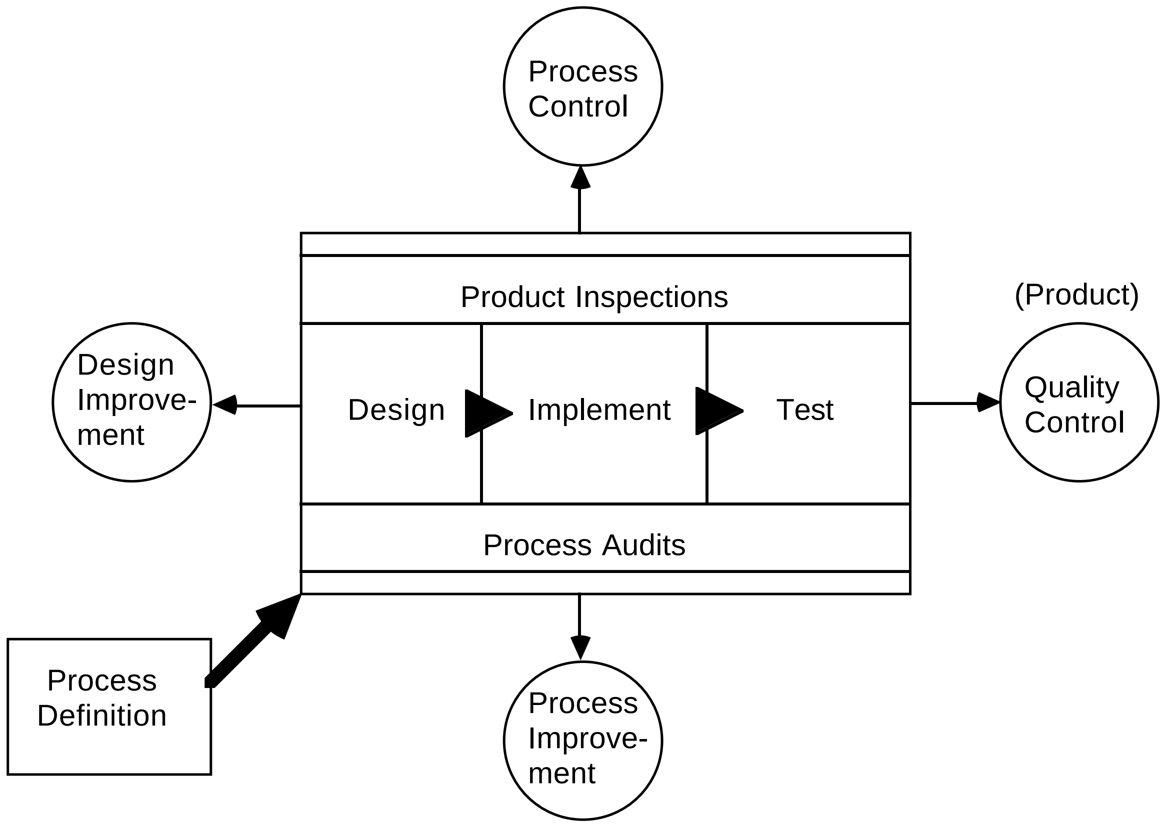


Figure B-4: Software Quality Functions

B.4.1. Process Definition

Because the software engineering process is intellectual rather than physical, explicitly defining that process—the steps, sequence, requirements, team composition, interfaces, etc.—can be challenging. Process definition leads to process control and improvement.

B.4.2. Product Inspections

The simplest quality technology involves performing peer reviews and inspections of both work in progress and final software products and documents. An inspection is a structured technique for comparing a software work product against its specification and other quality criteria [Fagan76]. Inspections result in corrections to software work products. Effective inspections require planning and training. Quantifying the results of inspections establishes a basis for process control and improvement.

[12]Responsibility for facilitation rests with the process group.

B.4.3. Process Audits

An audit of the software engineering process focuses on the adequacy of the methods, tools, standards, and procedures in place in a project [Crawford85]. It also considers the conformance of the project to the prescribed defined process. The actual processes may, after all, depart significantly from the intended process. To be effective, process audits should suggest corrections to the software engineering process. Process audits should be oriented to improvement, rather than adversarial and oriented to problems. Poor quality and productivity may signal the need for an audit. An evaluation or assessment using the maturity model [TR23] can be considered a process audit.

B.4.4. Software Quality Control

Software quality control involves measuring the quality of a software work product and determining the need for corrective action. For example, if reliability is less than required, then rework and further testing must be performed [Currit86]. Often the customer will specify acceptance criteria for software products. Alternatively, the software enterprise may have internal criteria. These may be stated in terms of complexity, module size, reliability, etc.

B.4.5. Process Quality Control

Once established, a software engineering process can be expected to perform at a relatively constant level until some unanticipated problem or change occurs (a *special cause* of poor quality). Tracking actual versus expected performance on a control chart can highlight process problems [Gardiner87]. Once the software enterprise recognizes that performance has departed substantially from expectations, the search for a cause can begin. Maintaining process control means continuously correcting process problems.

B.4.6. Software Design Improvement

Until recently, hardware designers left many important product quality considerations to be handled by manufacturing engineers. Because software does not go through a corresponding manufacturing phase, the software engineer must deal with those producibility concerns directly [Card90]. That is, the software system must be designed to be easy to implement and maintain. This is in addition to satisfying the customer’s functional requirements.

B.4.7. Process Design Improvement

Once the process has been defined and controlled, management can turn its attention to improving the underlying process. This can be done by simplifying the process and by inserting appropriate new technology. [Turner78] suggests five questions that should be asked about each process element:

1. Is this activity necessary or can it be eliminated?
2. Can this activity be combined with another or others?
3. Is this the proper sequence of activities?

4. Can this activity be improved?
5. Is the proper person doing this activity?

Initial process improvement efforts should be concentrated at *leverage points*: those activities that require the most effort and produce the most problems (see, for example, Pareto analysis in [Grady87]). Improvement actions should be evaluated in situ by studying their effect on actual process performance.

B.5. Quality Improvement Teams

To be effective, responsibility for the quality functions described in the preceding section must be distributed throughout the software organization. No single department has all the skills and scope to deal with myriad issues. Management must provide leadership and disseminate common goals to all levels of the organization [Deming82]. Coordination, cooperation, collaboration, and communication within the organization can be facilitated by properly structured quality improvement teams, such as process groups.

One team approach, about which much has been written, is the quality circle (QC). Quality circles are voluntary groups of workers studying and trying to improve their own tasks. In Japan, quality circles function as group suggestion programs. They are easy to implement and inexpensive to operate.

Although the approach has been widely adopted in Japan, it has had limited success in the U.S. [Lawler85], and the payoff from quality circles can be small. Because quality circles typically are formed at a low level in the organization, they lack the resources to implement major changes and can easily be ignored by management. Moreover, the homogeneity of membership often limits the innovativeness of a quality circle. Serious process problems often affect more than one task, so may be unsolvable by a single quality circle.

Overcoming these obstacles means recognizing four principles of team-building for improvement [Persico89]:

- Management must be involved.
- Teams are social and technical entities.
- Teams require diverse skills and experience.
- Members need specialized training in process analysis, statistics, and problem solving.

The software engineering process group concept is an attempt to adapt these principles to software quality improvement. The result is a high-level, skilled team with dedicated resources and a mission to make improvements.

B.6. Conclusion

The quality movement is an international force. Its influence has begun to affect the way software is developed and maintained. Many issues regarding the application of quality improvement principles to software engineering can be resolved only through experience. The SEI maturity model of the software engineering process, assessment procedure, and process group concept are tools that can help to install TQM in government and industry organizations, where it can continue to evolve.

Appendix C: Candidate Action Planning Process: The Search Conference

Marvin Weisbord, in [Weisbord87], Chapter 14, "Inventing the Future: Search Strategies for Whole Systems Improvement," suggests a method for developing a strategic portion of the action plan that could also be used to create the framework for the tactical action portion.[13] That method—a search conference—is summarized here. It was selected because it blends the excitement of visible improvements with the need for a written, formal plan for implementing those improvements.

The result of a search conference is a shared vision of the future of software engineering in an organization; this vision is recorded primarily in a strategic action plan. The plan, therefore, is not a study; it is a structured report on the future, on the expectations and method of attaining that envisioned future.

Focusing on the future has several benefits:

- The focus removes some of the energy from the natural tendency to place blame on those who are believed to be responsible for the current state.
- The future provides a rallying point. It is, after all, the future in which everyone is interested. Improvements, by their nature, live in the future.

In a search conference, all the stakeholders gather off-site for several days. Trained facilitators lead the group to discover and create their vision of the future. Before giving more details about this activity, it might be helpful to describe what it is not: it is not a listing of problems and corresponding solutions. These lists tend to be so long that they are overwhelming. And the solutions are piecemeal, not systemic.

Rather, the conference is a search for, in Weisbord's words, "images of potential, an image of aspiration, a preferred future." Weisbord cites a 1979 study that reports that when people plan present actions by working backward from what is really desired, they develop energy, enthusiasm, optimism, and high commitment. Says Weisbord,

> Search conferences excite, engage, produce new insights, and build a sense of common values and purpose. They have been especially attractive to organizations faced with significant change: markets, mergers, reorganization, new technologies, new leadership, the wish for a coherent culture and corporate philosophy. (p. 285)

According to Weisbord, the search conference is based on three assumptions:

1. Change is so rapid that we need *more*, not less, face-to-face discussion to make intelligent strategic decisions.

[13]A similar method can be found in *The Action-Training Approach to Project Improvement: Guidelines for the Trainer*, by Merlyn Kettering and Terry Dean Schmidt, U.S. Agency for International Development, Washington, D.C., July 1987.

2. Successful strategies...come from *envisioning preferred futures*. Problem solving old dilemmas doesn't work under fast-changing conditions. Each narrow solution begets two new problems. [Emphasis in original.]
3. People will commit to plans they have helped to develop. (pp. 285 - 286)

C.1. Organizing the Conference

A committee composed of the facilitator(s) and four to six potential participants meet to decide on dates, place, schedule, and the menu for meals and breaks, along with participant selection method, group tasks, and goals. They might select a theme, such as "Software Development Here in 1995."

A group of 50 or 60 might be invited. A diversified group provides a high potential for creative and synergistic solutions, increased likelihood of follow-up, and more linkages outside the department conducting the conference. The key is to involve only people who have a stake in the sponsoring organization's future. Any group important to implementation should be represented. Top management always participates. The overriding guideline is diversity.

One effective schedule begins with dinner on Wednesday evening, followed by the first work session. This makes for a fast start-up the next morning, the participants having had the night to gather their thoughts for the next two days of meetings. This schedule provides yet more time, the weekend, for reflection before returning to the pressures of everyday work.

C.2. The Conference

The participants first focus on the past: which significant events, technologies, breakthroughs, and processes have had the greatest impact on the present state of the persons participating, on the enterprise, on the industry as a whole, and on society at large? Weisbord speaks of organizing the perspectives by decade, looking for patterns, meanings, and emerging trends. The group can respond to the direction of some of these past events as a way of discovering important—shared or not shared—values in an objective way.

The present is then dealt with from two perspectives: external and internal. Those forces that impinge most on the enterprise from the outside are listed in priority order. Deciding on the forces and their priority helps build teamwork; additional values surface. Then a list of "prouds" and "sorries" is drawn up; this is the perspective from the inside. It is a list of those things going on right now about which the participants feel good and bad. The group votes on the "proudest prouds" and "sorriest sorries" as a way to mutually acknowledge the mistakes and shortcomings and to build commitment to do something about them.

Participants consider the future next. Groups are given a few hours to develop a rough draft of a preferred future: the most desirable, attainable future five years hence. After the groups have presented their visions of the future, each participant prepares three lists: suggested actions for themselves, for their function or department, and for the whole organization. Individuals keep the list of personal actions for their own use. Departments review the lists by function and develop departmental action plans. The conference steering committee—or, better, top management—reviews the suggestions for the organization as a whole, yielding the strategic action plan for the enterprise.

The result of the search conference, then, is a layered set of action items for process improvement, each set aimed at its appropriate level. Since the action items were developed by the community of stakeholders, there is already an investment in their successful implementation.

Weisbord reports:

> I do believe that anyone who has attended one of these events remembers it for a lifetime. The search conference links values and action in real time. It promotes productive workplaces by using more of each person's reality. It joins people who need one another, yet rarely interact, in a new kind of relationship. It stimulates creativity and innovative thinking. It offers a unique...springboard for planning and goal setting. (p. 295)

These are probably the most difficult aspects of the search conference:

- Management may feel threatened by appearing to give up decision-making to a larger group.
- The outcome depends upon facilitation. What if a few powerful individuals dominate the conference? Who would contradict one's supervisor in such a meeting? Managing the participants, rather than managing the content, may be the principal challenge.
- Convergence—or even enough agreement to enlist support—is not assured. In fact, even the assessment results may come into question.

In the end, the decision to try a search conference is an issue of tailoring and reaching consistency with the corporate culture, as must be done with other activities suggested in this guide.

Appendix D: An Introduction to Technological Change

This appendix addresses the fundamentals of implementing technological[14] change in software organizations. The process group can greatly improve its odds for success if it understands and acquires some skills in managing the technological and organizational change attendant to process improvement.

This material is intended for software professionals. It is an introduction to the basic knowledge that is prerequisite to effective and predictable technological change in the context of software engineering. The field of management science, and in particular the areas of organization behavior and organization development, are the primary sources for this material.[15]

This appendix contains two sections. The first section, which is tutorial in nature, introduces basic vocabulary and concepts. It is adapted from the lecture material in the course, *Managing Technological Change*, developed by ODR, Inc. Through a licensing arrangement, the SEI has taught this course to software professionals who act as change advocates and change agents.

The second section presents a discussion of information transfer, which we distinguish from technological change. Technological change involves efforts internal to an organization and implies adoption and everyday use of a technology. Information transfer involves systematic access to "outside" technology to prepare for long-term, ongoing change.

D.1. A Brief Tutorial on Implementing Technological Change

Software engineering is such a new discipline that constant procedural and technological change is a way of life in most software organizations. Perhaps because change is taken for granted, it is not often specifically addressed. But change needs to be managed. At a minimum, managing the process of introducing new approaches will make that process more predictable, thereby reducing the surprises and thus the stress involved. Reduced stress will help minimize negative impact on productivity.

The idea of managing a process such as *change* implies that there are predictable aspects of that process that can be addressed. Some of these aspects are familiar, but because they deal with the human side of change rather than with the technological side, they are

---

[14]As noted earlier, the term *technology* is used here and throughout the guide in the broadest sense of the term. It refers not only to CASE tools and compilers, for example, but to project management, technical review methods, and any other change that affects the technical aspects of how an organization goes about its business.

[15]Resources include [Tornatzky90], an interesting and readable interdisciplinary survey of many aspects of technology transfer, and the annotated bibliography included in [Przybylinski88], which was especially prepared for software engineers and managers.

often oversimplified and are not directly addressed by software engineering organizations. When a technological change takes place, the following human changes may occur as well: need for new skills and knowledge, gained either through training or on-the-job experience; revised reporting schemes; redefined work roles and relationships; revised schedules and deliverables; gain or loss of status. When these changes are carefully and systematically addressed, the resulting plans become part of a project-like road map from the present state to the new, desired state.

One way to begin to recognize these predictable aspects of change is by learning something about the nature of change. The following sections provide concepts and vocabulary that may be helpful.

D.1.1. Characteristics of the Change Process

In even the most effective organizations, evolving business conditions and technology can create the need for change. The process of discovering and accepting the need for change is called *unfreezing*. This process can be very distressing, because it can involve changing carefully crafted and sometimes long-standing policies and procedures, and because it appears to be a move away from stability and equilibrium. People and organizations usually change only when it is more painful to stay in their current condition than to accommodate change.

Moving from a less desirable state to a more desirable state is called *transition*. The transition period is characterized by disequilibrium, during which the status quo undergoes alteration to incorporate new procedures and/or technology. High stress and reduced productivity are the norm, but disequilibrium can be minimized if the transition process is carefully anticipated and planned. In the case of technological changes, such as adding new software tools or procedures, it is especially important that transition planning address changes to both the organizational and the technical environment. It is possible to reduce the unpredictability of the transition period by preparing revised guidelines, offering training, and running pilot efforts to test the change prior to broadly instituting it.

As new technology and processes become part of routine organizational behavior, equilibrium is re-established. Some refinement of the use of the new technology may be necessary during this period, but by the time *refreezing*—a refocusing upon the product rather than the process—begins, the change should be generally accepted and tested. If an organization is planning a series of changes, it can soon begin to consider the next change effort. If the change has occurred in a pilot effort, lessons from the experience can reduce the disruption caused by subsequent similar changes. Although change efforts are discussed here as if they occur discretely, it is an oversimplification to say that change occurs "one change at a time," especially in a field as new and dynamic as software engineering. The fact that several—or even many—changes may be underway simultaneously is one reason a process group is so important. This group can maintain expertise in change management, and track and facilitate several changes simultaneously. It can also learn, on behalf of the organization, from each change experience and transfer that learning to the next change effort, gaining effectiveness and predictability each time.

D.1.2. Key Roles in the Change Process

Successful change efforts are usually characterized by certain actions or roles taken by people within the organization, as summarized below:

- **Change advocate.** The change advocate, sometimes known as the champion, must be in place first. This is the person who sees the need for change earliest, who can envision the effect of the change, and who can convincingly represent and convey that vision to other members of the organization. The change advocate often becomes a change agent.

- **Change agent.** When advocacy moves from proselytizing for change in a general area of need—for example, software project schedule overruns—to implementing a particular change, such as the use of cost-estimation models, the person who implements change is called a change agent. An agent needs resources and support in order to actually effect change.

- **Sponsor.** A person who can provide approval and resources to support the agent is called a sponsor. The sponsor who allocates resources and supports the beginning of the change effort is called the *initiating sponsor*. Other sponsors are called *sustaining sponsors*.

  Typically a sponsor is a manager. The sponsor of a process group, for example, should head the organization served by the process group. The sponsor of a change effort ought to be the manager of the organization that will undergo the change. Both the process group and a particular change effort may, however, need additional sponsors. For example, a steering committee is institutionalized sponsorship. The steering committee consists of managers whose organizations will be affected by the changes advocated by the process group, and the committee's sponsorship is prerequisite to successful transition of the changes. In another example, if the manager heading the organization acquiring a new configuration management (CM) tool is not also in charge of corporate CM software standards, the change agent may have to seek complementary sponsorship from the standards organization.

  Sponsors must exist at all affected levels of the organization. Here is the test for adequate sponsorship: if a particular sponsor were no longer available, would a particular change effort, or even the process group itself, be in jeopardy?

- **Change target.** The organization (or individual) that will change is called the change target. The size of the change target should be decided by the level of available sponsorship, the resources that can be applied to the change effort, the possibility and necessity of pilot efforts, and the breadth of application of the new technology or procedure. (It may be difficult, for example, to use configuration management on only part of a system under development.)

All roles are necessary, and they must be played in the right balance. [Bayer89] indicates that neither upper-level management alone nor practitioner support alone bode well for a successful adoption of new technology. The need for multi-level support is also discussed in [Ebenau83] and [Leonard85].

D.1.3. Frames of Reference

One of the most troublesome aspects of change is communicating the nature and implications of the change to those it will affect. And one of the most difficult aspects of communication is bridging *frames of reference*. A frame of reference consists of ideas, theories, beliefs, feelings, values, and assumptions which, taken together, allow people to meaningfully interpret experience. Frames of reference simplify the processing of the many stimuli that bombard us constantly. Two familiar examples from outside software engineering are labor and management frames of reference, and male and female frames of reference. Both examples have provided many classic anecdotes of miscommunication. Software organizations may need to address the frames of reference of management, customers, software engineering, systems engineering, hardware engineering, and different application domains.

In the context of implementing new software technology, a common occurrence is a clash between a manager's frame of reference and an engineer's frame of reference. For example, management may see great value in improved cost estimation because it will allow them to predict resources more effectively. Engineers may view this as added overhead that takes time away from their "real" work, that of designing and building systems. Whether or not a particular constituency views the change with favor, concerns and assumptions need to be articulated and addressed. Very early in planning, the change agent (i.e., the process group) should actively solicit responses to the proposed change. Informal interviews, meetings, and suggestion boxes (perhaps electronic) can be used to gather input. The change agent must remain carefully open-minded, as often these responses hint at implementation strategies and transfer mechanisms. Engineers faced with using a new cost-estimation procedure may be concerned about its calibration to their work. An initial argument about the overhead required for the procedure may mask real technical issues that must be addressed.

D.1.4. Resistance to Change

People resist change even if it is perceived to contribute to their own well-being. Change is a digression and *should* be resisted unless its impact can be projected to be beneficial within a reasonable period of time and the process of change can be reasonably orderly.

Individuals resist change for a variety of reasons. A new technology or procedure may threaten their stature in an area affected by a proposed change. They may feel that the new technology or procedure will make them and their jobs obsolete. They may be aware of a negative impact a change could have on the organization's product costs or delivery dates. Or, very simply, they may prefer not to deal with a change in their usual work routine.

The key to successfully addressing resistance to change is to address each issue specifically, perhaps within a particular frame of reference. Particular attention should be paid to the influential individuals—called *opinion leaders*—who feel strongly against or in favor of the change. These individuals often can affect the plan for change in important ways. For example, when a process group is preparing an operational-level action plan for a particular project, key members within that project should serve as sounding boards, providing specific information about the "fit" between the plan and existing technology and practice. If technical working groups are organized, opinion leaders should be invited to join.

D.1.5. How Much Change?

Organizations, like people, are subject to stress. Too much or too rapid change creates stress and jeopardizes the success of any change effort. It is not unusual for even a relatively small change such as a new technical review process to take several months to implement [Ackerman83, Fowler86] considering time required for planning, technology selection or procedure definition, actual change activities such as training and guidelines preparation, and evaluation. Attempting to implement change too rapidly can result in large quantities of rework, and usually is much more costly than doing it right the first time.

Introducing too many changes too quickly can also increase the risk of failure of a particular change effort. Scheduling individual change efforts to allow enough time for a return to equilibrium after each one allows an organization to continue to acclimate to changes comfortably and almost indefinitely.[16]

D.2. Technology Transfer

It is useful to distinguish *technological change* from the *transfer of technological information*. For the sake of this discussion, the former is defined as what happens when a cohesive organizational unit installs and adopts a new technology, and the latter is what happens when technology becomes part of the knowledge base—although not necessarily part of the skill set—of a technical community, be that an organization, an application domain, or a broader community such as the embedded, real-time systems community. Technological change and information transfer, mechanisms for which were discussed in Chapter 7, comprise technology transfer. This section provides a conceptual framework to show how these two elements of technology transfer complement and support each other. The section presents key aspects of technology transfer inside and outside the organizations, including *context analysis*, *transfer project selection*, *mapping* of technology across contexts, and *boundary spanners*—people who perform the mapping process.

The preceding section explained technological change as what takes place when a technology is carefully and purposefully implemented in a particular organizational unit. For example, change is what happens when all software engineers in a program learn how to use software inspections and begin inspecting code on a regular basis. Change is also what happens when managers in a particular division agree on and begin to use a particular cost-estimating tool routinely. Essentially, change occurs when a group of people whose work is

[16]While this might seem to preclude simultaneous change efforts, most software technology changes will affect one group of engineers primarily, and others secondarily. If multiple change efforts are managed well, no one group undergoes more than one major change at a time, and it should be possible to have several efforts smoothly underway.

closely related join together to learn about and implement a new way of working together using a new procedure or technology.

If change were necessary only occasionally, the organization could muddle through the process informally. But with the goal of ongoing software process improvement, and the fact of constantly changing technology in the new field of software engineering, many such changes may occur at once. It is critical, therefore, that an organization plan for these changes, frame the ongoing change within a strong strategy and clear goals, systematically facilitate the change process, and carefully monitor the changes that are underway. The process group plays a key role and acts as a focal point in planning.

A process group could plan, successfully facilitate, and monitor a specific change effort within an organization with the knowledge described in the previous section. Preparing the strategy and long-term goals for ongoing change, however, requires a different perspective. The process maturity model offers a scale by which to calibrate goals in terms of process and technology. For example, an organization at the "repeatable process" level on the maturity scale should "...introduce a family of software engineering methods and techniques. These include design and code inspections, formal design methods, library-control systems, and comprehensive testing methods" [Humphrey88]. These actions are part of moving toward the next, or "defined process" level of maturity. Collectively, however, they represent a massive, multi-year effort across a multiple-unit organization. The process group and its steering committee need a way to decide which technology to implement when, the best place to put it first, and then when to implement it more broadly. An understanding of the dynamics of information transfer provides a basis for designing an action plan that will have the greatest likelihood of success.

D.2.1. Context Analysis

Context is the set of all social and technical aspects of an environment. It includes everything that surrounds and pervades the organization in which a technology will be implemented, such as:

- Application domain
- Project size
- Organization software development standards
- Government software development standards
- Professional standards
- Software engineering process definition
- Economic and business conditions
- Budgeting practices
- Financial status
- Managerial style and reward system

- Technical style and reward system
- Geography of organization locations
- Technological environment (hardware and software)
- Equipment for each software engineer
- Networking sophistication
- Availability of continuing education and professional training
- Homogeneity of staff
- Education and experience of staff

Some of these aspects are addressed as part of assessing the software process; others fall outside the process assessment and need to be examined separately. For example, if training in software project management is considered critical to process improvement but there is no precedent for it other than management development training for project managers, it may be difficult to find a way to budget time or money for the training. Or, if the corporate culture advocates using custom tools, the organization may resist purchasing an off-the-shelf cost-estimation model even if ample funds are available. Context analysis is a useful next step after process assessment in preparation for identifying tactics for specific process improvement activities.

Context also includes the dominant frames of reference within an organization. Analyzing context helps articulate these frames of reference so they can be explicitly addressed in the transfer process. The greater the similarity between the native context of the change agent and that of the change target, the greater the likelihood the change will succeed. The less similarity there is, the greater the need to carefully analyze the context and examine all assumptions [Rogers83].[17] Context analysis can help clarify dissimilarities, provide criteria for change agent selection, and anchor the planning of change in the specifics of the organization's situation.

Aspects of context that are outside the realm of influence of the process group and its sponsor need to be acknowledged and taken into account in planning, even when they cannot be modified.[18] For example, standards and policies that cannot be changed locally need to be carefully considered. Often there are creative ways to work within the constraints. Limited budget for formal education and training can be compensated for through judicious use of libraries and through sharing knowledge by using simple mechanisms such as "brown bag" seminars during lunch hours. Local upgrades of networking capability may be constrained

[17]Rogers defines the terms *homophily* and *heterophily* to describe the degree to which a change agent is more like or less like the change target. [Rogers83] is one of the classics in the study of technology transfer.

[18]DoD standards are a good example here. However, if a standard limits an industry's ability to upgrade technologically, an organization may want to allocate resources to lobby with others for broader change, via industry or professional associations.

by a budgeting cycle, but a high-level sponsor's help might be enlisted to find sources of funds outside the local budget limits or cycle. Changes in reward systems may require broad corporate initiative; local options can be enhanced by conscious attention to how the corporate systems affect local practice. Managers typically have some discretion in using existing systems; at a minimum, for example, they can exercise leadership and sponsor informal celebrations to acknowledge a particular success. For each change attempted, the context should be analyzed at enough levels to be sure that each way it affects the change effort has been acknowledged and addressed. One approach to such an analysis is shown in Figure D-1. Note that this diagram is greatly simplified: several categories of individuals and functional groups are likely, and a chart should be made for each important category in an organization.

Categories of Context / Area Affected	**Technical**	**Business**	**Cultural**
Individual	Equipment	Education and Training Policy	Privacy and Size of Workspace
Functional Group	Software Development Standards	Schedule	Professional Homogeneity
Organization	Electronic Communications Capability	Co-location	Reward System

Figure D-1: Context Analysis

D.2.2. Mapping

Mapping is a simple but powerful way to determine whether a technology is likely to succeed in an organization. The mapping process essentially involves: 1) examining the technology to be implemented for aspects of context that have been built into it and 2) comparing ("mapping") the results of this examination to the results of the organizational context analysis. If there is minimal overlap, the risk is high that implementing the technology will be complex and unpredictable and will require more resources, especially in pilot efforts. If there is a significant overlap, the chances of success are greater. The comparison should take into account the fact that some aspects of context may carry more weight than others.

Context analysis is a useful step to take just prior to beginning a search for a technology. It can help narrow the candidate field by noting major aspects of context, such as application domain, technical compatibility, and budget, which would seriously constrain possible matches. Once the list is pared down to a half dozen or so technologies, mapping is a major step in the final selection process.

Mapping (see Figure D-2) is necessary because all software engineering technologies are based on a set of assumptions. Certain application domains or classes of application domains—for example, real-time embedded, MIS, scientific—may be assumed when a technology is developed. Certain types of users—for example, ordinary citizen, engineer, scientist, bank teller—may also be assumed. Certain technical requirements—machine type and size, operating system, network type—are usually stated. The easiest technologies to eliminate from a candidate list are those with the least overlap with the organization's context; as the list shrinks, the mapping process requires more detective work and attention to subtleties such as software engineering terminology, life-cycle phase coverage and definition, and style of work in groups. Even a technology that is well targeted to a particular market niche cannot take all variations of an organization in that niche into account. The best test of a new technology is a series of pilot uses, thoroughly evaluated.[19]

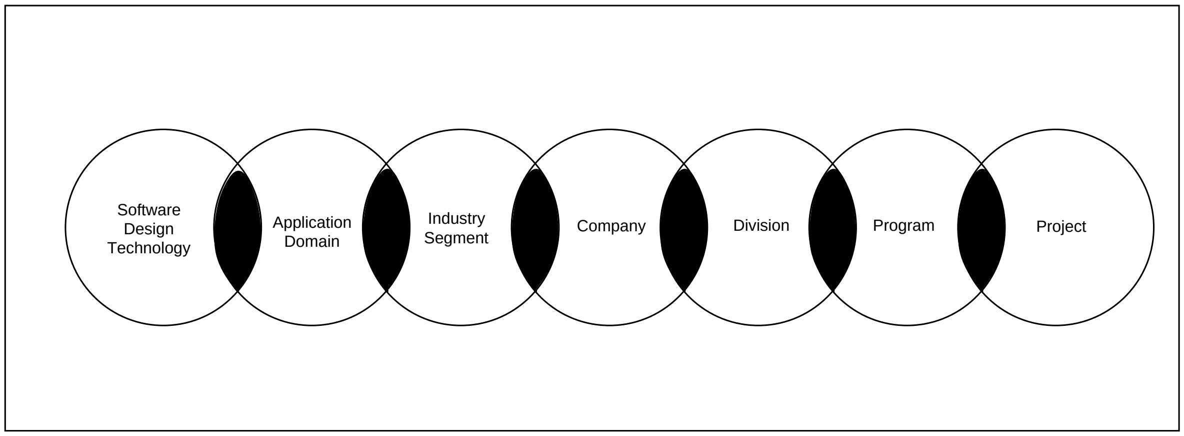


Figure D-2: Mapping Technology

Mapping is less critical with a more mature technology such as UNIX; the technology has been used in many different contexts and a body of experience exists. Many books are available, and user organizations and conferences provide ongoing access to the experience base. Newer technologies such as Ada and reuse are not yet mature in this sense. Nor are "softer" technologies such as technical reviews and software metrics. In these cases, successful technology implementation depends predominantly on people and people-based mechanisms such as sponsorship, training and education, articles and conference presentations, and consulting. Even with books such as [Grady87] providing excellent "how-to" advice, success is still heavily dependent on the "right" people using that advice. The same is true even for computer-aided software engineering (CASE) tools, despite the burgeoning marketplace. [Martin88] notes the need for a new position, a CASE administrator, to institutionalize expertise. Without such a person, designers using the tools must perform design and process improvement in parallel with system design.

[19] A very useful discussion of the "mutual adaptation" of organization and technology, which must occur for successful technological change, can be found in [Leonard88].

When an organization is dealing with newer and softer technologies, mapping becomes critical. If transfer mechanisms are people-based, they are heavily influenced by frame of reference and by the skills of the people who must translate between frames of reference in the transfer process.

D.2.3. Boundary Spanners

Boundary spanners are people who are comfortable in multiple frames of reference; they can work effectively as members of a business organization, with management and engineers, and on professional society committees. Examining the process of communicating in different languages provides a useful analogy to the services boundary spanners perform.

Because the mapping process is similar to translation between languages, it is best done by boundary sponsors who are the equivalent of multi-lingual. If these people have extensive experience in translation, they will know when to proceed cautiously; they will understand the need, for example, to define carefully terms that most people will assume are well understood.

Boundary spanners are common *outside* organizations in the technology advocate roles described in Figure D-3. These boundary spanners play an important role in getting new products from research into the marketplace, in articulating the new technology in universities and professional communities, in marketing the technology once it becomes a product, and in supporting new users. They use many of the transfer mechanisms we have discussed in this guide. Marketing people in particular have important skills in combining various transfer mechanisms to support systematic change.

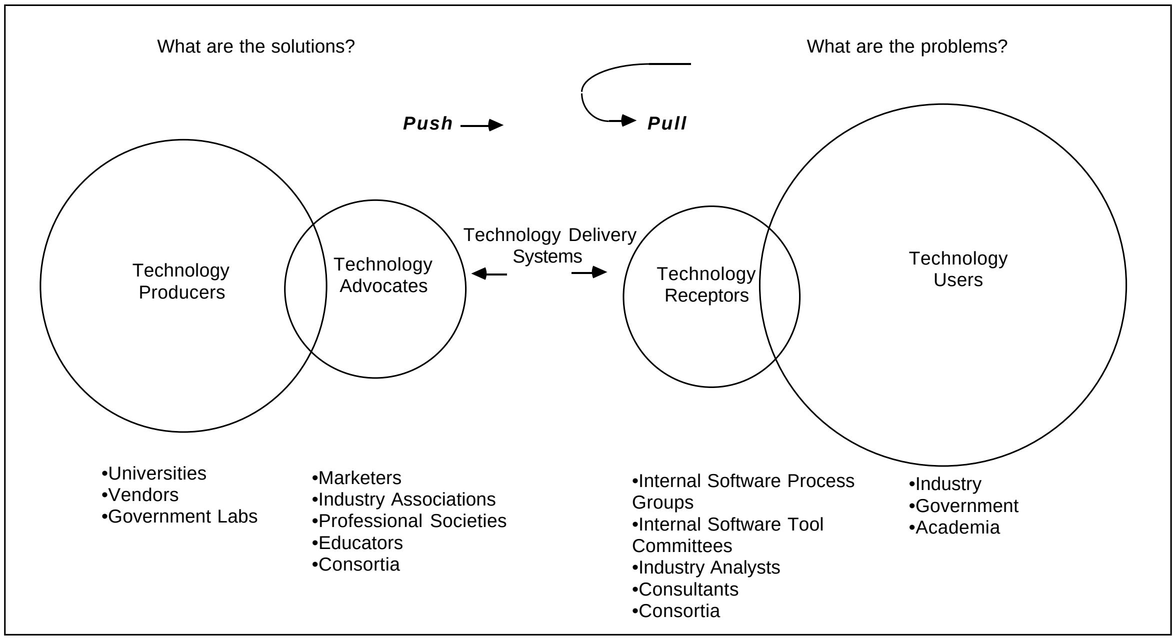


Figure D-3: Boundary Spanner Roles

The functions performed by change agents usually have not been replicated inside organizations. This must change, first, because the problems that exist in the outside world also exist in microcosm within all organizations interested in, and needful of, technology transfer and change to support efforts such as software engineering process improvement. Secondly, boundary spanners, or technology receptors, working on behalf of an organization, are necessary for reasons of symmetry and interface to the outside world. These are critical issues and, together, are the reason a process group is an essential part of the change process for software organizations.

Boundary spanners working on behalf of an organization represent that organization's *problem set*. They are needed to provide symmetry with those working on behalf of technology, the *solution set*. The symmetry in this case results when a group of people act on behalf of the organization, and serve to map (translate) as well as *filter* technologies from outside.

The filtering process is implicit in the process of mapping; however, it deserves explicit discussion. Boundary spanners focus both on the internal transfer process and on the outside world. In fact, in many ways they sit on the "surface" of the organization. In their role, boundary spanners typically serve on technical working groups or as members of the process group "porting" technology experience from one organizational group to another; they connect the inside of the organization to the outside world. They track technology directly [Carlyle88]; they may also track it by serving on outside working groups such as the IEEE Computer Society standards committees. Having largely internalized context analysis results for their own organization, they effectively scan for and select candidate technologies.[20] People who perform this role best have had experience in a number of contexts; criteria for boundary spanners are much the same as for members of process groups.

Without boundary spanners to act as a filter, naive users are confronted directly with new technology. They may be unskilled at either the boundary spanning work of technology selection or at implementing technological change. Mistakes made here can be costly. In the absence of the filtering function provided by experienced boundary spanners, the organization is bombarded directly and randomly with opportunities to adopt technology. Selection is not made carefully, and failure is often the result. Boundary spanners are especially necessary in a new field such as software engineering, where technology is often immature and changes constantly. They are, in a sense, acting as advocates for the consumers of new technology within their organization.

[20]It is the job of the process group and steering committee, however, to take the broader view; the process group members, having worked with many projects, and the managers, also having done so and having had management experience and being able to represent concerns of the organization, must determine how all the technologies brought back by various categories of boundary spanners fit together and serve together for the greater good.

Appendix E: Training and Education

"If you think training is expensive, try ignorance." - Derek Bok, Harvard University

Training and education can be powerful instruments for change. For the purpose of this discussion, *education* refers to the process of providing information—especially concepts and theory—about technology, while *training* refers to providing skills in the use of a technology or procedure.

Education of those who will participate in the change initiates the *unfreezing* process (see Appendix D) and is prerequisite to implementing actual change. Motivation to acquire information through education or other means may come from shifts in business conditions or from new technology that makes it necessary to consider change. Key employees, both managers and technical staff (especially those in working groups), need to get enough information about a new technology or set of procedures to determine its applicability to their situation. Their education should include a description of the new technology or procedure, its use in similar applications elsewhere, and estimated costs and benefits. Since sponsors and key people in the target organization need information as input to decision making prior to beginning any change, education of this sort should occur very soon after it appears that change may be needed. In fact, an ongoing strategy to provide information continuously may be needed; this is discussed at greater length in Chapter 7.

Once a technology is selected for a specific improvement activity, new skills must be provided to those who will use the new technology. Training—instructor-led, embedded, video-based, or any combination—can efficiently provide these skills. The training must take place no more than one or two weeks prior to the initial use of the technology and should take place after any new technology has been acquired and installed by support staff.

Thus, the need for training must be identified early, as part of the action planning process. At the same time, certain training-related activities should begin, such as preparing course descriptions and selecting vendors. The training itself must occur "just in time" for people to acquire the necessary skills.

E.1. Sources of Education and Training

The process group does not necessarily develop or provide education and training relating to process. The group is, however, responsible for the provision of both. The current state of the market supply is such that the training and education needed for process improvement procedures and technology is often not available in a form suitable for immediate use. Thus, the process group, working with members of technical working groups, may need to tap both internal and external sources of appropriate instructors, course developers, and materials.

E.1.1. Internal sources

There may already be an internal training or education organization that provides engineering skills development and/or continuing education. Such an organization can support the process group's efforts with course development, delivery, and acquisition services. Even if the process group must prepare and deliver or acquire its own courses, the internal training and education organization should be kept apprised of these efforts.

Internal training and education organizations may be good sources of courses, if the courses can be tailored to suit the needs of particular improvement activities. Education and training will be most effective when tailored to suit the specific technical context in which the technology will be used; working group members and pilot project participants can assist in the tailoring process. Course material should provide examples from contexts similar to the one in which the technology is targeted for use. For example, if the targeted context is embedded software systems, examples might come from telecommunications, medical instrumentation, or aeronautics. Training materials should provide exercises to build skills, knowledge, and confidence; they should use languages and concepts familiar to the students (for example, Ada code inspections should not be taught with Pascal or C language examples).

Survey courses or general-purpose courses on broadly used programming languages and computer systems are most commonly available. These may need revision in order to provide answers to questions about the application of a technology or process in a specific project. The process group should be prepared to negotiate the tailoring of existing courses and to supply or help find resources for developing or acquiring and teaching new courses. Education and training are not the entire change process, as noted earlier, but are so important that the process group should be prepared to invest a significant percentage of its efforts in obtaining, providing, and maintaining appropriate training and education.

Technical working group members are valuable resources. They can help determine which education and training to offer, help tailor the material, and act as sources of expertise to supplement formal courses. Working group members can also be enlisted to attend potentially useful courses and evaluate them, and to provide informal tutorial and related consulting services. If the need for education or training is short term and for limited numbers, this can be an adequate overall solution. Otherwise, it can be used as a stop-gap measure until more suitable arrangements can be made.

E.1.2. External sources

If internal training and education sources do not exist, outside sources such as universities and vendors can be used. More and more commonly, universities are involved in continuing engineering education; and often, in addition to generally available courses, there are faculty who can be engaged to develop and deliver training and education courses. Some training vendors will contract to develop original courses or to tailor the courses they routinely offer. The same suggestions that apply to working with internal sources apply here.

Whether vendors or university faculty are being considered, it is important to select carefully. University faculty may not be accustomed to writing or tailoring courses that focus on skill building. Vendors who offer to tailor and provide "hands on" exercises should be asked for examples. In either case, the process group must be prepared to negotiate and pay for these additional services. Note that *references should be checked.* References can provide information on how easy a particular supplier is to work with, the quality of the supplier's product, lead time required, typical follow-up activities, and cost.

E.1.3. Acquiring Course Materials

The first step in any system development effort is requirements specification, and course development is similar to software development in many respects. The process group should be prepared to provide a specification for courses it wants, even when it is attempting to use internal and existing courses. The specification can be the result of an assessment or other formal needs analysis. It should include a description of the number and types of students who will take the course, the context of course use (it is part of the process improvement effort), the timing of the course offering(s), and the length of time students can be away from their work. It should also specify behavioral or attitudinal objectives (for example, "On completion of this course, the students will be able to moderate code inspections," or, "On completion of this seminar, the managers who attend will be able to sponsor a pilot test of configuration control technology"). Course suppliers should respond to the specification in writing and should identify course prerequisites and describe the student background and experience that is expected.

Course development that results in consistency of delivery independent of a specific instructor is a labor-intensive process. It can require up to 100 staff hours of effort for each classroom hour,[21] including workshop and exercise time, especially if a text does not exist. This approach assumes that instructors who meet certain basic prerequisites can be found or trained, and it provides for the creation of course materials and notes for both student and instructor. Lower effort figures may be possible if instructors have both extensive subject matter expertise and course development experience. Courses to be delivered in certain media take longer to develop in exchange for appealing to a wider audience or having broader distribution possibilities; a course using interactive video disk can take hundreds of hours of development time per course delivery hour.

If the course will be taught many times, the high front-end investment in a well-designed and documented course that can be taught by a number of different instructors may be cost-effective. Reputable suppliers will provide details on how courses will be developed and tailored in accordance with the level of detail in the course specification.

[21]This is the experience of a number of SEI technical staff who have developed courses.

E.1.4. Criteria for Selecting Materials

Materials and providers should be selected carefully, considering the investment in the development or outright acquisition of educational and training materials, students' time away from regular work assignments during in-class time, and the importance of education and training in the process improvement effort. The following questions should be helpful:

- **Fit**. Do the content, approach, prerequisites, and course objectives fit the requirements?
- **Flexibility**. What is the extent to which materials can be used with a variety of audiences and instructors? What materials are provided? How are they kept up to date?
- **Tailorability**. How readily can *all* the materials be customized for individual project needs? Is the course designed to be customized?
- **Track record**. Is there an objective way to evaluate past performance with these materials/courses? Can you examine student and instructor feedback from previous offerings?
- **Practical**. Will the material/course impart skills, or are they general overviews? Are there "hands-on" exercises?
- **Modularity**. Is the material packaged so that it can be learned in sections based upon student background, interest, need (i.e., level of expertise or skill to be obtained), and time available?

Appendix F: Action Plan Guidelines

The action plan, as discussed in Chapter 3, is actually a set of plans and guidelines. It includes a strategic plan and a two-part set of tactical plans, one containing guidelines for preparing operational plans, and the other, a set of operational plans derived from that guidance. This appendix offers direction in creating these sections of the action plan.[22]

F.1. Strategic Plan

F.1.1. Introduction to the Strategic Plan

Explain the intent of the plan and the policy that has motivated it. Give references for any policy documents and spokespersons.

Give a brief history of assessments and other events leading to the development of the plan. Cite any documents from those activities. List the types and positions of the participants.

F.1.2. Overview

Explain how the strategic portion of the action plan provides an overall framework for software engineering process improvement efforts. Use the diagram in Figure 3-1 in Chapter 3.

Explain that the strategic portion of the plan will provide answers to the following questions:

- What are our goals in beginning a major software engineering process improvement effort?
- What is our motivation to improve?
- What have we assumed?
- Who is directly responsible and how will we work together?
- How will we know when we succeed?
- What are we going to do to begin improving our software engineering process?
- How will we continue to improve?

F.1.3. Process Improvement Goals

List the strategic goals that have been developed as a result of the assessment.

[22]This material was aided by references to [Gessner84], [Mason81], and [Schmidt87].

F.1.4. Motivation

List the principal motivations that will drive the change effort. Use terms that are specific enough to enable people to understand what motivations are appropriate. Be clear about the need to improve.

F.1.5. Assumptions

List critical assumptions, describing how each affects the plan.

Discuss the risks implied by this set of assumptions.

F.1.6. Organizing for Process Improvement

Explain the functions of new organizational entities created in support of the process improvement efforts. The reader should be able to determine whom to ask a question of, submit an idea to, or give feedback to. Use Figure 3-3 and text from Chapter 3 as needed. Explain the following:

- The role of the executive sponsor and what resources he or she has committed.
- The role of the steering committee and how members are chosen.
- The role of the working groups and how members are chosen.
- How projects will participate in specific improvement efforts.
- The role of the process group.

F.1.7. Responsibility Matrix

Note that the steering committee will cross organizational boundaries to support the working groups in their implementation of software engineering process improvements.

Identify the final approving authority of the working groups.

List all process group coordinating activities with both working groups and the steering committee.

List all working group responsibilities.

Use Figure 3-2 as the basis for a matrix listing responsibilities, including approvals and reporting structure.

F.1.8. Criteria for Success

Describe how to measure and evaluate the success of improvement efforts at both the organization and the project levels. It would be helpful to tie this to the motivation for change.

F.1.9. Improvement Agenda

Describe the agenda for improvement that was developed as a result of the assessment and other data gathering activities. This should include any technical areas—configuration management, cost and schedule estimation, technical reviews, CASE—and organization-wide study efforts in areas such as software engineering training and education. The process group itself is an agenda item. If it already exists, the item might state how it will improve. If not, this agenda item is a statement chartering the process group. Other agenda items might note the need for working groups to support some of the process group's efforts; for example, research into useful metrics might be a joint effort of the process group and a working group prior to beginning to develop the process database and attendant procedures.

List the agenda items in priority order with the name of the responsible person or group. Note that the charter for each will be available in the tactical section of the action plan.

F.1.10. Guidelines for Planning Improvement Projects

Describe here the guidelines for planning specific improvement projects. Note that project teams are supported in this planning effort by members of the appropriate technical working group, the process group, and the steering committee, all of whom may join together in a planning workshop. Collectively, and prior to beginning detailed planning, this group should be able to answer all the questions listed in Chapter 6 about a new technology and the process of putting it in place.

Planning workshops should address five areas:

1. **Project activation.** Ensure that the project has met all legal, administrative, and bureaucratic requirements and is ready to begin operations.
2. **Project organization.** Plan how the project organization will function, clarify roles and responsibilities, and develop operating procedures.
3. **Schedules.** Agree on specific project goals and objectives, then establish realistic plans, budgets, and schedules.
4. **Resources.** Establish procedures for ensuring that the key project resources (money, people, training, and equipment) are available when needed.
5. **Information and control system.** Identify information needs, select measures and indicators, and establish a useful monitoring and reporting system.

The above process is iterative. The tasks can be completed in other sequences, but none of the five major steps can be neglected or the project will rest on a weak implementation foundation. At the end of the workshop, the participants will have prepared written plans, budgets, schedules, and responsibility descriptions to guide the implementation of specific process improvements. The planning workshop produces specific results and also builds team capacity for planning future improvements.

F.2. Tactical Plan

F.2.1. Introduction to the Tactical Plan

Note here that there will be several types of tactical plans and related guidelines. The content of the plans for the process group and the steering committee will differ to some extent from that of working groups in technical areas. Also, the plans for technical areas will contain material to use in creating a detailed operational plan for a specific improvement effort in that area.

There may be quite a few tactical plans and instantiations at the operational level. These need not be kept in one document. Each working group can maintain its own, keeping the technical library and process group informed of the date and location of the latest version.

The content of the tactical action plan is divided in two parts (refer to Figure 3-1 on page 30):

Part 1

- Working group charter.
- Plan for the review and selection of appropriate technology.
- Plan for the development of an operational plan template that supports the technology and transition approach.

Part 2

- Template for the operational plan.
- Guidelines and examples for completing the template.
- List of completed templates.
- Lessons learned from projects using these templates.
- List of contacts in the working group(s) and in projects with relevant experience.

F.2.2. Tactical Planning for Technical Working Groups

F.2.2.1. Technical Working Group Charter

Reference the agenda item in the strategic plan that will be implemented by this working group.

List the working group's responsibilities and functions. For technical working groups, these typically are (at a high level):

- Screening new technology, or technology new to this organization.
- Making recommendations on new technologies to pilot.
- Planning, in harmony with the process group, for the transition of new technologies from evaluation to pilots to broad usage.
- Participating in outside working groups (professional society or industry).
- Attending conferences and workshops related to this technical area.
- Organizing workshops and seminars inside the organization to pass along highlights and/or technical details about new technology.
- Contributing technical evaluations or trip reports to internal newsletters.

- Consulting with projects beginning to use a new technology in the working group's area.
- Reviewing papers for internal and external conferences in related areas.
- Working with the process group to design plans for improvement projects related to this technical area.
- Assisting with the execution and evaluation of these projects.
- Preparing templates to be used for operational plans, along with guidelines for and examples of their use.

Describe the level of management commitment to these functions. Note the name of the management sponsor(s) if other than the steering committee.

F.2.2.2. Template for Improvement Activity Plan

Note that this template is to be used to guide the development of a plan for a specific improvement activity, and that each template will differ, depending on the technical and administrative aspects of the activity.

Most templates will have the following entries:

- Statement of why the particular organization is adopting a new technology or procedure, citing related policy and needs analysis activities and reports, noting goals and objectives, and describing criteria against which success will be measured.
- Description of the technology or procedure to be adopted, along with a list of reference material and sources of expertise.
- Description of any enabling technologies or procedures required.
- Description of the source of technology or procedure itself and attendant services, such as education or training, consultants, and installation.
- Description of purchase procedures for both the technology or procedure and any enabling technology or procedure.
- Description of plans for tailoring the technology or procedure, and the technical or administrative environment that will receive it, as well as related materials such as user documents.
- Description of plans for educating management and practitioners about the new technology or procedure and the reason for its adoption; description of related training for those who will need new skills.
- Description of technology selection procedure, if applicable (some working groups may select two or three suppliers for a particular technology).
- Description of how the improvement activity will be evaluated, and how status of both adoption and evaluation will be reported.
- Schedule of all the above activities and a list of responsible individuals.

F.2.2.3. Guidelines for Completing the Template

Write guidelines for completing the template (a good way to approach writing these is to do the first several plans without a template, derive the template, and then convert the lessons learned in writing the plans into guidelines). Refer to the sources listed below for completed templates that can be used as examples. Note the possibility of doing this work in an improvement project planning workshop.

Because each improvement activity is a project, the IEEE *Standard for Software Project Management* [IEEE-PMP88] is a useful guideline for creating improvement activity plans. All sections will not apply, but most will, and the use of this standard helps prevent omission of key information.

F.2.2.4. Example of Completed Templates

Include here at least one example of a completed template.

F.2.2.5. List of Completed Templates

List the locations of completed templates created by similar improvement project work.

F.2.2.6. Lessons from Experienced Projects

Include here (if brief, otherwise list sources) descriptions of lessons learned by others in the process of planning and executing improvement projects.

Provide a checklist of questions to answer to capture lessons learned from this improvement project.

F.2.2.7. List of Contacts

Project Contacts: List the names, phone numbers, and electronic and physical addresses of others who have participated in similar improvement efforts and who are willing to serve as resources.

Working Group Contacts: List the same for working group contacts who can provide expertise or services in this technical area.

F.2.2.8. Record of Working Group Meetings

Include dates, chairpersons, attendees, minutes.

F.2.2.9. Record of Working Group Activities

Briefly summarize each activity—workshop, newsletter contribution, demonstration, seminar.

Note for each activity the date, attendees, and the agenda (if the activity was a meeting).

Note any future plans.

F.3. Tactical Planning for the Steering Committee

F.3.1. Steering Committee Charter

Describe the purpose of the steering committee and how it will oversee and support various process improvement activities.

Describe its relationship to the process group and the working groups.

List its members; describe how members are selected and how they retire.

Give the name of the committee's executive sponsor.

F.4. Tactical Planning for the Process Group

F.4.1. Process Group Charter

Describe how the process group acts as a focal point for software engineering process improvement. Mention the functions it serves:

1. Coordinates all self-assessment activity.
2. Coordinates all action planning activity.
3. Coordinates all working group activity.
4. Acts as secretary to steering committee, keeping it informed, setting meeting agendas, keeping minutes.
5. Recommends new working groups.
6. Maintains expertise in the implementation of new technology, and consults in same during process improvement projects.
7. Maintains the process database.
8. Provides process consulting.
9. Integrates the "big picture" across all improvement activity and reports on this to working groups and the steering committee.
10. Coordinates training and education related to process improvement.
11. Maintains the process improvement library of meeting minutes, lessons, references, proceedings, etc.

List process group members, how to reach them, and, if possible, their particular expertise.

Appendix G: Summary and Analysis of SEPG Workshop Data

Introduction

At its 1989 SEPG Workshop, the SEI created and administered an informal questionnaire for gathering information about process groups represented by workshop attendees. This was done at the request of the attendees, who indicated they needed the information for briefing their management on what other organizations were doing. Figure G-1 shows the questionnaire.

What follows is a summary and limited analysis of the data submitted for all the questions. Of 22 questionnaires, 20 were usable; that is, they were submitted by a representative of a process group that was already established, even if very new. The questionnaire was in every sense of the word a prototype: rough and not scientifically validated, but developed to get a quick sense of the process groups represented at the workshop.

Summary of SEPG Data From Attendees
2nd Annual SEI SEPG Workshop
June 21 - 22, 1989

1. What is the number of full time equivalent people assigned permanently to the SEPG? (How many personnel are you allowed? How many do you have now?)

2. What is the number of full time equivalent people assigned on a rotational basis to the SEPG? (How many personnel are you allowed? How many do you have now?)

3. If your SEPG is not full time, how many people are assigned and for what percentage of their time?

4. Do you have working groups for technical areas (reuse, Ada, education, etc.) or other special planning functions? Please list each one and note how many people participate for how much time how often. Please note the approximate date of origin of each group.

5. What is the date of origin of your SEPG?

6. How many software professionals do you serve?

7. Are you a corporate process group or are you focused just on one division? Please briefly describe the organization(s) you serve.

8.Do you have corporate, division and middle management sponsorship? Please relate the answer to this to question 7's answer.

9. Describe what special activities/planning you did to get started and how long these took for how many people.

10. List your near term tasks.

11. List your long term tasks.

12. How many geographic locations do you serve?

13. What are the key accomplishments of your group to date?

Figure G-1: Questionnaire Administered at the 1989 SEPG Workshop

Results: Questions 1 - 3

Figure G-2 summarizes the results of Questions 1 through 3. Of the 20 process groups, 14 were at their allowed staffing level; 6 were below. The largest group was allowed 20 full-time equivalent (FTE) staff; the smallest had no full-time members at all, but rather 5 people at less than 5% of their time. Only about one-third used rotational personnel. Ten process groups used only full-time staff, 6 used only part-time staff, and 4 used a mix of both full-time and part-time staff.

FTE* Allowed		**FTE Have Now**		**FTE Rotators Allowed**		**FTE Rotators Have Now**	
No. FTE	No. SEPGs	No. FTE	No.SEPGs	No. FTE	No. SEPGs	No. FTE	No. SEPGs
0	1	0	1	0	13	0	13
.8 - 2	5	1 - 2	6	1 - 2	2	1 - 2	3
3 - 4.5	5	3 - 3.5	6	3 - 4	3	3 - 4	2
5 - 7	7	5	3	5 - 6	2	5 - 6	2
9	1	6 - 7	4				
20	1						
Average = 4.73		Average = 3.39		Average = 1.2		Average = 1.1	

*FTE = Full-time Equivalent Staff

Figure G-2: SEPG Staffing

Results: Question 4

Figure G-3 lists types of working groups noted by respondents and the number of process groups having each type. A total of 37 working groups were reported. Most had been in existence at least briefly, but a few were listed as planned. Four SEPGs had no working groups; 2 SEPGs had only one, and the remainder had two or more (see Figure G-4). Eight SEPGs had working groups in the area of training and/or education; 7 had working groups in Ada.

We speculate that, because of the differences in naming working groups, more types are listed here than actually exist. For example, two process groups mentioned metrics as an area where they had working groups; other categories that might be predominantly metrics work are "availability," "process database," and "cost/size/schedule estimation." If these categories are added together, five SEPGs have working groups in the general category of metrics. Likewise, if "software engineering process" is combined with "standards/policies/ procedures," which seems very similar, five SEPGs have working groups in that category.

Technical Area	No. of SEPGs with working groups in that area
Training/Education	8
Ada	7
Standards/Policy/Procedures	4
Cost/Size/Schedule Estimation/Tracking	3
Reuse	3
Environments	2
Metrics	2
Tools and Methods	2
Software Engineering Process	2
Software Technology	2
Requirements Engineering	1
Process Database	1
PC communications	1
Availability	1
Simulation	1
None	4

Figure G-3: Technical Working Groups

The data on working groups raised some questions. With as much apparent activity in the CASE area as appears from informal feedback to the SEI and from both technical and business literature and magazines, it was surprising that only two groups listed tools and two listed environments. Several process groups noted that they had full-time "working groups." They are included in the count, but it is questionable whether they really fit the definition. Perhaps tools, environment, CASE, etc., are not mentioned more frequently because these areas have already become established as permanent parts of the organization. It would be interesting to know how people chose which working groups to establish, and to what extent process groups have played a role in originating and facilitating working groups.

No. Working Groups	No. SEPGs
0	6
1	2
2	9
3	2
4 - 5	3

Figure G-4: Summary of Working Group Data

Results: Questions 5, 6, and 12

Over half (11) process groups were 8 months old or less. The balance ranged in age from more than 8 months to 5 years (see Figure G-5).

Age	No. SEPGs
≤ 8 months	11
< 2 years	3
2 - 3 years	4
4 - 5 years	2

Figure G-5: Age of SEPGs

The number of software professionals served by the process groups ranged from 100 to 3500. Most groups served 500 or fewer (see Figure G-6).

No. Professionals	No. SEPGs
≤ 250	7
300 - 500	5
600 - 800	2
1000 - 2000	5
3000 - 3500	2

Figure G-6: Number of Software Professionals Served by SEPGs

A finding was anticipated: a relationship between the age of a process group and its size as a percentage of the population it served, the theory being that SEPGs prove their worth over time, and older groups should therefore be a larger percentage of the number of professionals they serve. No such relationship was found.

Most process groups served one or fewer locations (see Figure G-7). The size of the groups did not, however, increase with the number of locations served. Considering the increase in the number of interfaces that accompanies the increase in locations, this is surprising. It probably indicates that the SEPG staffing level is effectively lower than the percentage of staff served would indicate.

No. Locations	**No. SEPGs**
1	10
2 - 4	8
10 - 15	3
> 25	1

Figure G-7: Geographic Locations Served by SEPGs

Results: Questions 7 and 8

Questions 7 and 8 looked for information on the type of organization served and related sponsorship issues. The inquiry was whether process groups worked at a corporate or a division level (or some other level), and to what extent they had management commitment. Figures G-8 and G-9 show the results.

Corporate (C) or Division (D) SEPG	**Sponsorship**		
	Corporate	Division	† Management
D		*	*
D		*	
C	*	*	
C	*		*
D		*	*
C	*	*	
D	*	*	*
D	*		
D	*	*	*
D	*	*	*
D	*		*
D	*	*	*
C	*		
D		*	
C	*	*	*
D	*	*	
D	*		*
C	*	*	
other (large project)			*
D		*	*

†Middle Management

Figure G-8: Level and Sponsorship of SEPGs

Thirteen process groups worked at the division level and had two or three levels of sponsorship. The caveat here is that the term *division* may be not used consistently by all respondents. Also, terms such as *sector*, *group equivalent*, and *lab* were considered in our analysis to be equivalent to *division*, but may not actually be equivalent.

SEPGs	Sponsorship	
All SEPGs	1 level	5
	2 levels	10
	3 levels	5
Corporate SEPGs only	1 level	1
	2 levels	4
	3 levels	1
Division SEPGs only	1 level*	3
	2 levels	6
	3 levels	4

* the SEPG that served one large project had one level of sponsorship

Figure G-9: Summary of Level and Sponsorship of SEPGs

Results: Question 9

Question 9 asked how people got started in their process group and related work, and what human resources were needed. The most frequently taken action was an assessment of some sort. Five process groups did a self-assessment after SEI training. Two groups conducted SEI-assisted assessments. Two others had assessments of another sort. After assessment, the next most common initiating actions were planning (action, strategic, SEPG charter, other) and establishing working groups, task forces, networks, or tiger teams. Negotiation with management was mentioned by three process groups; it is likely that there was much more of this but that it was not mentioned because it was taken for granted. Figure G-10 lists all initiating actions and the number of process groups that performed each one.

Action	No. SEPGs using
Assessment (self, SEI-assisted, other)	9
Planning	7
Task force, working group, tiger team	7
Organization changes	4
Negotiation with management	3
Competitive analysis	1
Training (other than self-assessment)	1
Software engineering conference for managers	1
Pilot Project	1

Figure G-10: SEPG Actions to "Get Started"

Results: Question 10

Software engineering process groups were involved in a wide variety of near-term tasks, as listed in Figure G-11.

Task	No. SEPGs
Software engineering policies, procedures, standards, manuals	11
Metrics program or procedures	7
Advance to maturity level 2	5
Planning	3
Training	3
Size, cost, schedule estimation procedures	2
Write SEPG charter	2
Improve Ada skills	2
Obtain sponsorship	2
Improve environment	2
Pilot project	2
Establish process database	2
Tool evaluation	1
Expand SEPG scope	1
Publish SEPG newsletter	1
Support MIS development	1
Establish infrastructure	1

Figure G-11: Near-Term Tasks

The most popular task by far had to do with the development of software engineering policies and procedures. Four of the eleven SEPGs that mentioned this task specifically stated that they were preparing a manual or a handbook. Next most frequently noted was a metrics program or procedure for gathering data. Five groups mentioned advancing to process maturity level 2 as a "task."

Results: Question 11

Process groups were asked about long-term tasks that they planned to address. Again, advancing one or two maturity levels was mentioned by ten groups. Responses indicated that three groups wished to advance to maturity level 2, six to maturity level 3, and one to level 4 and eventually 5. The next most frequently mentioned long-term task was "continuous improvement of process." The others listed are in Figure G-12.

Tasks	No. SEPGs
Improve maturity level	8
Put metrics tools and procedures in place	6
Continuous improvement of progress	5
Insert technology	5
Establish training program	4
Expand SEPG charter	3
Implement action plan	2
Expand scope of SEPG efforts	2
Perform assessments	1
Increase degree of reuse	1
Instill a software quality culture	1
Do long range planning for the SEPG	1
Consult	1
Coordinate feedback from ongoing projects	1
Sell strategic plan to management	1

Figure G-12: Long-Term Tasks

Results: Question 13

Question 13 asked SEPG representatives to list key accomplishments to date. Standards, training courses, and improved technical software development environment were mentioned most often. Figure G-13 lists all responses.

Accomplishment	No. SEPGs
Developing and documenting standard process	9
Training course(s)	8
Improving the technical environment (tools, workstations, etc.)	6
Self-assessment/Re-assessment	5
Getting started, planning	4
Establish reviews or inspections	3
Obtain or improve sponsorship	3
Metrics teams or standards/process	3
Obtained funding	2
Pilot projects	2
Professional activities (papers, workshops)	2
Newsletter	1
Established infrastructure	1
Expanded SEPG charter	1

Figure G-13: Key Accomplishments to Date

The responses to Questions 9, 10, 11, and 13 were analyzed with respect to the age of the process group answering. The analysis sought to determine whether particular activities were more common to newer groups (8 months or less in age—11 SEPGs) or older groups (more than 8 months old—9 SEPGs).

Figure G-14 compares responses between newer and older process groups for some of the more frequently mentioned activities.

Activity	Question Number	Number of newer* SEPGs mentioning	Number of older† SEPGs mentioning
Self-assessment	9	5	4
Increase maturity level	10	4	1
Increase maturity level	11	8	2
Process definition and standards	10	7	4
Metrics	10	3	3
Metrics	11	2	3
Metrics	13	2	1

* Eight months old or less.

† More than eight months old.

Figure G-14: Activities of Newer Versus Older SEPGs

Index