From the Publisher

Process Improvement: Plan for Success

Reuel S. Alder
Publisher

Discipline is no fun—I consider day planners self-inflicted torture. My idea of a good day is to wake up with no plan and accomplish more than humanly thought possible. The work would be intuitively discovered as the day progressed. Creativity and spontaneity would be enhanced, and routine, repetitive activities would be minimized. Each day would be a fresh and exhilarating experience filled with learning, personal growth, and development. The variations would be unlimited, and the success would be phenomenal.

But if you believe the last 40 years of development data, this dream is not achievable for most software projects. Yet we are still largely living in a dream world where we think software can be built by pure "artists" who arrive at river's edge with no plans, and through sheer talent can turn a pile of scrap iron into a decent bridge. However, I have learned from unfortunate personal experience that almost all significant human achievements require more than just talent and creativity. Decades of data prove it: Even the best software artists do better work when they start with a foundation of planning, preparation, and discipline.

This issue of *CrossTalk* addresses software process improvement. Development models like the Capability Maturity Model may not be as easy to apply as the random search for truth, but they help apply the discipline necessary to create complex software systems.

Real process improvement is not easy, and anyone who believes otherwise has either never tried it or has never helped make an improvement of lasting significance. Learning better techniques and technologies is only the beginning—there are many human aspects to work through as you try to fund the improvements, sell all the players on them, and then follow up until the changes are institutionalized.

The expertise to navigate through these challenges is available, and over the years the Software Technology Support Center has acquired much experience in helping others adopt proven processes and technologies. We know that to adopt a new process, you must first create a strong process improvement infrastructure.

First, you need an initial assessment to know your strengths and weaknesses so your senior managers can scope the improvement effort. Only with this knowledge can you customize an infrastructure for process improvement. We have had success with plans tailored according to the Software Engineering Institute's IDEAL model. This includes formation of a Management Steering Group, which helps you develop a charter and vision and to establish a clear match to organizational goals and objectives.

You will then need to establish functioning change agents (such as Software Engineering Process Groups) and implementation teams (such as Process Action Teams) who know their roles, responsibilities, charters, and action plans. They will be much more effective if they receive the right formal and informal training.

Without a strong process improvement infrastructure, it may be impossible to institutionalize superior processes and technologies. If you do not have such an infrastructure in place, do not hesitate to get the help you need to build one. Process improvement is paying dividends for those with the discipline to do it right. ◆

Letter to the Editor

Acquisition Managers Get What They Pay For

Your July issue contained a letter to the editor from Joe Saur of Fort Monroe, Va. It is interesting that in acquisition management the "developer" (usually the contractor) is still referred to as the enemy. He also refers to the IV&V (independent validation and verification) group as "experienced developers" (are we to consider developers here as the enemy, also?) hired to search through the "horse dung (documents)."

As a contractor, I run across this mindset every time I go to a military installation. It is a sad state of working affairs to continue to make references to developers or even internal IV&V team members in this way. And if the developers are searching through the horse dung to find the intent, the fault probably rests with how the contract was awarded. You get what you pay for. If you pay for horse dung, expect to receive horse dung. It sounds like Saur is and has been constantly on the receiving end of work not produced by a company assessed at least CMM (Capability Maturity Model) Level 3, and they are probably not ISO (International Organization for Standardization) registered. Most of the companies the government does business with do not meet either criterion. And usually, these companies are able to bid their services at a much lower rate than one that makes the investment in its people and processes to do the job right. Therefore, I recommend to Mr. Saur that he reassess his idea of who is the enemy and pro-actively eliminate the problem before a contract is awarded by basing award on technical capability and desired output and not the lowest cost.

As a final note, most of the experienced developers (IV&V team members) I've dealt with have had less than five years experience, with most being in the one- to two-year range. Remember that they are a part of the equation, too. In this analogy then, the IV&V members are not the Apache but instead are the Apache wannabes.

Alan L. Reagan
Warner Robins, Ga.

Hybrid Multi-Model Assessment
When the CMM Meets ISO 9001

Robert C. Bamford and William J. Deibler II
Software Systems Quality Consulting

This article outlines a strategy and methods to employ formal appraisals to determine which model—or which elements from either model—offer the most value for a particular software engineering organization. The article is illustrated with examples from our experiences in guiding software engineering organizations to examine and select the most appropriate model for software development.

Faced with governments transitioning to commercial standards, and responding to business pressures to expand into new lines of business, many software engineering organizations are faced with adapting their CMM-based systems for ISO 9001 compliance. At the same time, many small- and medium-sized software engineering organizations are exploring methods to exploit these models for process definition and improvement. In the United States, CMM (Capability Maturity Model) to ISO (International Organization for Standardization) is emerging as the prevalent transition for government organizations and their suppliers. In Europe, because of the early adoption of ISO 9001 and ISO 9000-3, there is greater interest by commercial software development organizations in the transition from ISO 9001 to the CMM. European interest is evidenced by steadily increasing attendance numbers for the annual European Software Engineering Process Group (ESEPG) conferences sponsored by the SEI.

In response to this industry need, a number of articles and conference presentations published since 1992 [1,2,3,12] have laid a foundation for comparing the requirements of the two models. These articles and presentations provide background information useful to people who are preparing to

- Plan a transition between models.
- Identify the most appropriate model.
- Implement the most effective and efficient combination of elements from both models.

CMM is registered in the U.S. Patent and Trademark Office. Capability Maturity Model is a registered service mark of Carnegie Mellon University.

The balance of this article presents steps such people can take to translate this generic background information into a detailed, specific action plan for their organization.

Where to Start

At the beginning of the selection or transition process, there are typically champions for both models and a small group responsible for selecting a model. Whether the champions participate in the group is determined by the company culture and the willingness of the champions to participate in a process that has an uncertain outcome.

The first step in the selection or transition process is to explore and assimilate enough of the large body of knowledge to overcome initial resistance to the language and structure of ISO 9001 and ISO 9000-3 [7] or to the sheer size of the CMM—even when the focus is only on Levels 2 and 3. For organizations that persevere, the outcome of this phase is typically an understanding that both models

- Can be the basis for effective process improvement in any software engineering organization.
- Are flexible in principle, and in practice, support any software development lifecycle.[1]
- Are extremely susceptible to problems introduced in the implementation (excessive bureaucracy, inflexible lifecycle definitions, over-documentation, lack of management or engineer buy-in, etc.).
- Require executive management commitment.
- Require continuing organization-wide support.
- Include regular appraisals (audits or assessments) to ensure the effective implementation and continuing relevance of the defined processes.
- Are not equivalent to each other, although ISO 9001 compliance and CMM Level 3 assessment are similar [3, 12].
- Have requirements that can be selectively implemented to satisfy the requirements of the other model. For example:
  - ISO 9001 clause 4.18, Training, maps to the requirements outlined in the Level 3, Training Program.
  - The key practices described in the Level 3 key process area (KPA), Peer Reviews, can satisfy the requirements of ISO 9001, clause 4.4.6, Design Reviews.[2]

Formal Appraisal

The second step in the selection or transition process is the appraisal, which creates a bridge between the available information and the impact of a model on the practices in a particular software engineering organization. The formal appraisal is a continuing opportunity to educate the organization about the content of the models, to allay concerns about bureaucracy and change, and to reinforce the credibility of the models and the appraisal process. Experience with the appraisal process also serves as a valuable input to the model selection. An appraisal is also a necessary first step in planning the introduction and implementation of any new system or process.

There are a number of well-defined, formal appraisal methods associated with ISO 9001 and the CMM: the ISO 9001 registration audit or pre-assessment, the Software Process Assessment (SPA), and the family of CMM-based appraisals (CBAs), which includes the Software Capability Evaluation (SCE), the CBA for Internal Process Improvement (CBA/IPI), and Interim Profiles.

ISO Registration Audits

Although the ISO 9001 audit process originally had little associated documentation, it has been well understood and consistently practiced by registrars:

- A scope is selected.
- The quality manual and supporting policy and procedural documents are reviewed.
- A plan and schedule are prepared.
- People from all levels of the organization within the selected scope are interviewed, typically at their workstations.
- The results of the audit, including a recommendation for or against registration, are consolidated into a presentation and report that is given to the managers of the audited organization.
- The report, including a recommendation for or against registration, is reviewed and approved by registrar personnel who were not involved in the audit.

The registrars' initial and surveillance audit methods have shaped the internal audits (ISO 9001, 4.17), which are periodic, mandatory self-assessments to ensure on-going compliance "with planned arrangements and to determine the effectiveness of the quality system." [6, clause 4.17]

The internal audits are critical to the management review process and to the management representative, who is responsible to "ensure that a quality system is established, implemented, and maintained in accordance with this International Standard." [6, clauses 4.1.2.3 and 4.1.3]

With the publication of ISO/IEC Guide 61 [8] and ISO/IEC Guide 62 [9] in 1996, the accreditation and the registration processes have been completely defined in standards virtually equivalent to ISO 9002. With the formalization of the role of the International Accreditation Forum (IAF), an audit structure has been implemented to monitor the on-going compliance of the accreditation bodies and registrars[3] and to ensure a baseline consistency across registrations.

Level 2 KPA	ISO 9001 Clause
Requirements management	4.3, 4.4
Software Project Planning	4.1, 4.3, 4.4, 4.9
Software Project Tracking and Oversight	4.4, 4.9
Software Subcontract Management	4.6
Software Quality Assurance	4.4, 4.17
Software Configuration Management	4.4, 4.5, 4.7, 4.8, 4.9, 4.12, 4.13, 4.14, 4.15

Level 3 KPA	ISO 9001 Clause
Organization Process Focus	4.9, 4.14
Organization Process Definition	4.2
Training Program	4.18
Integrated Software Management	4.4, 4.9
Software Product Engineering	4.4, 4.10
Intergroup Coordination	4.4
Peer Reviews	4.4

Figure 1. *Relationship between the Level 2 and Level 3 KPAs and the ISO 9001 clauses.*

Because of the experience of the assigned assessors, advance preparation tends to be minimized. Typically, only the lead assessor reviews the quality manual, the top-level document, and selected policies and procedures. In the course of the on-site interviews, auditors examine process and product documentation in detail to ensure compliance with planned arrangements. ISO 9001's unique requirement that the quality policy be "understood, implemented, and maintained at all levels of the organization" [6, clause 4.1.1] has led to the practice of interviewing a sample of some 15 percent to 20 percent of the organization. Although national programs, like TickIT, include guidance (suggestions) regarding contact hours [4, section 11, p. 18], there are no international standards that define sample size and contact hours for ISO registration and surveillance audits. This has become a problem as competition forces registrars to reduce costs.

For implementation guidance, a pre-assessment by a registrar falls short. Although the registrar will be as thorough as the implementor requires, the results can only define observed nonconformity. Corrective recommendations would be a form of consulting, which is forbidden by accreditation bodies and ISO/IEC Guide 62 as a conflict of interest that would contravene the impartiality of the assessors [9, clause 2.2.2 (o)], who would have a vested interest in their recommended solutions.[4] As a result, a large, independent consulting community exists to provide detailed, collaborative audits, similar to the CBA/IPI (pre-assessments, gap analyses, diagnostic audits, etc.) to support implementation and action planning.

CMM Appraisal Methods

It is only since 1993 that methods like the SCE have been published. Before that, detailed information about the SCE Method was available only through SCE team training, which was available only to government teams [15, p. 28]. The Software Engineering Institute (SEI) has also published a standard, the CMM Appraisal Framework (CAF), against which any CMM-based appraisal method can be evaluated [16] and with which it has been stated the SEI-provided methods will comply.

From comments in [16, p. 25] and [17, section LA.B, p. 13] and from a wealth of anecdotal data provided by numerous people, it appears the CMM appraisal process has changed dramatically. In its initial form, the appraisal process did not rely on objective evidence or on systematic techniques to obtain evidence; the process incorporated interviews with project managers and free-form discussions with functional area representatives. The CMM appraisal process has evolved to its present form, engaging individuals from all levels of a development organization, including middle management, and incorporating more traditional auditing methods and systematic techniques for corroboration.

This change in method, in terms of the CBA/IPI, is presented in the SEI lead assessor training [17, section LA.B, p. 13]. Key improvements are identified as

- *Documentation review*. The CBA/IPI incorporates more extensive documentation review.
- *Interviews*. The CBA/IPI includes individual interviews of project leaders, structured group interviews of middle managers, and structured group interviews of functional area representatives and individual contributors. Previously,
  - Middle managers were not necessarily interviewed.
  - Functional area representative interviews were free-form discussions.
- *Data consolidation*. The CBA/IPI incorporates more systematic analysis and consolidation of the data from the interviews and documentation review.

The CBA/IPI requires an extensive commitment from the software engineering organization undergoing the assessment. An integral part of the CBA/IPI, as a CAF-compliant method, is training for a team of members from the sponsoring organization, who participate in the assessment and who are positioned to participate in the follow-up process improvement activities. Although this commitment of resources and money has a high potential rate of return, it is typically more than can be justified by an organization investigating whether the CMM is applicable.

Key Differences Between the Appraisal Methods

Both the ISO audit and the CBA/IPI produce objective results to support process improvement. Although the two models differ in content and scope,[5] both appraisal methods require that the organization have completed the implementation activities necessary to define and institutionalize—implement across the organization—effective processes. The appraisal confirms the success (or continuing success) of the implementation by gathering and analyzing objective evidence. Both methods require that management invest whatever resources are required to address issues identified in the assessment or audit. There are, however, a number of key differences between the methods.

Difference 1: Level of Involvement

The most prominent difference between the ISO audit and the CBA/IPI is the level of involvement required of the organization. To position the organization to understand and take action on the findings, the CBA/IPI requires that one or more members of the organization serve on the assessment team. The comparable ISO audit relies on a representative of the audited organization (the guide), who accompanies the ISO auditor, and depends on detailed, written findings reviewed with the audited organization's management as the last step in the on-site portion of the audit.[6]

Difference 2: Interview Methods

A second difference is in how input is gathered. In an ISO audit, interviews are typically conducted in or near the interviewee's workplace and are attended by the auditor, interviewee, and the guide. In the CBA/IPI, although project managers are interviewed individually, small groups of middle managers and small groups of functional-area representatives meet with panels of assessors, chaired by the lead assessor or another member of the assessment team. The group approach is based on the principle that individual contributors will be more willing to speak frankly when they are part of a small group.

Difference 3: Reporting Results

The third difference is in how findings are reported at the conclusion of the assessment or audit. As described above, in conjunction with Difference 1, detailed, written findings are provided to and reviewed with the audited organization's management as the last step in the on-site portion of the ISO audit, typically on the day following the last interviews. Findings are expressed in terms of a specific clause of ISO 9001 and include detailed information about the nonconformity and are classified as major or minor. The only global finding is binary. Based on the identified nonconformities, the audit team states whether it recommends that the audited organization be registered or retain its registration.

The CMM assessment concludes with presentations of draft and final findings, defining at a minimum the organization's strengths and weaknesses at the KPA level. Detailed information is transferred through the members of the organization who participate on the assessment team. In addition, a written report may be purchased as an option, but typically it is not available for an additional four to six weeks, and it does not necessarily include any significant information beyond that which was published in the final findings presentation.

Difference 4: Role of the Lead Assessor and Assessment Team Make-Up

In the CBA/IPI, preparation spans at least two weeks. Team members may lack prior assessment experience or direct experience with the CMM. In the first week, the lead assessor trains the team. In the second week, documentation is reviewed, and the assessment checklists and schedules are prepared. The lead assessor conducts key interviews, leads team interviews, and facilitates team discussions that reach consensus on findings. The lead assessor also is the CMM expert, guiding the team in its interpretation of whether observed practices satisfy the requirements of the CMM. This latter function becomes increasingly critical as the CMM is applied to commercial organizations, providing products to customers outside the Department of Defense (DoD) community. The problem of interpretation is exacerbated as commercial organizations go beyond the experience provided by many DoD software providers who have been able to rely on CMM-compatible Military Standards, e.g., MIL-STD-2167A and MIL-STD-498, to completely define the implementation.

The CMM recognizes this problem and states that "Organizations using the key practices should be aware of these conventions (in expressing the key practices) and map them appropriately to their own organization, project, and business environment."[7]

To mitigate the problem of auditor preconception and to enhance the maintainability of the implemented quality system, ISO 9001 requires that the organization create a quality manual, which documents how the requirements of the standard are addressed.

Difference 5: Who Appraises the Appraiser—Ensuring the Quality of the Assessments

The ISO registration process includes four levels of quality assurance:

- ISO lead auditors must complete an approved course.
- Performance of ISO lead auditors is monitored by the registrars.
- Performance of registrars is monitored by accreditation bodies.
- Performance of accreditation bodies is monitored by the IAF.

The CBA/IPI infrastructure is less extensive; the SEI is just beginning to define standards, guidelines, and a mechanism for pro-active monitoring of the quality of assessments. People with extensive experience in software development, including participation in two CBA/IPIs, can become registered lead assessors by completing the SEI curriculum and completing an assessment observed by a lead assessor. Only assessments conducted by SEI-authorized lead assessors can be recorded at the SEI as "SEI-recognized" assessments. The registry of CBA/IPI lead assessors is maintained by the SEI.

Selecting an Appraisal Method

For *CMM Level 3-compliant* or *ISO 9001-compliant* organizations, one or two people experienced in both models and familiar with the organization, and who ideally have been or have access to internal auditors or members of the CBA/IPI team, can convert two available sources of information into an accurate benchmark of the position of the organization with respect to the other model. The first source of information is the organization's library of presentations and reports from recent appraisals (ISO 9001: internal and registrar's audits; CMM: CBA/IPI, SCE, and SQA audits and reviews). The results of these assessments record the degree of "institutionalization" (CMM) or "effective implementation" (ISO 9001) of the required practices. The second source of information is the organization's set of documented policies, procedures, and standards, which describe in detail how the organization should operate.

Omissions identified in this conversion are addressed by updating the existing set of process documents. A by-product of the conversion is a mapping of the organization's policies, procedures, and standards to the requirements of the other model.

For *CMM Level 2-compliant organizations,* the most effective strategy depends on whether the organization is committed to the CMM and how well it is positioned for Level 3, i.e., how much groundwork has been done, exceeding the requirements of Level 2. If the CMM is well-established and the organization is well-positioned for Level 3, the recommended strategy is to continue to Level 3 and to follow the conversion strategy outlined above for a CMM Level 3-compliant organization. An alternative for a CMM Level 2-compliant organization is to commission an ISO pre-assessment and proceed with an exclusive ISO focus. To the extent that the Level 2 organization has adopted standard processes across projects (a Level 3 requirement), work products and processes will carry forward to ISO.

Organizations that are not committed to either model face the greatest challenge—and opportunity—in designing an appraisal strategy. The most straightforward approach, to undertake separate ISO and CMM appraisals, is typically too expensive, too time-consuming, and too confusing. The *hybrid model* and its associated assessment method offer a viable alternative for the currently uncommitted organization.

Hybrid Models

A number of hybrid models exist, including Bootstrap [5, 11] and Trillium. Although these models incorporate ISO 9001, ISO 9000-3, the Malcolm Baldrige National Quality Award criteria, the CMM, and various other standards, they do not answer the needs of the organization investigating ISO 9001 and the CMM. The outputs of the appraisals associated with these hybrid models are unique to the model and do not facilitate translation among the various source models with which these proprietary hybrid models are intended to compete.

Hybrid Multi-Model Assessment

The emergence of a general agreement on the relationship between the requirements of the CMM and ISO 9001 implicitly defines another hybrid model, the union of the two models, and forms the basis for a set of assessment tools and report templates that address both the goals of the CMM KPAs and the requirements of ISO 9001.

The tools associated with the hybrid assessment address the requirements the two models share, like documented procedures for planning, and those requirements that lie outside the intersection of the two models. For example, ISO requirements for record retention, technical support, packaging and distribution, and software maintenance are not addressed in the CMM, and the CMM's detailed requirements for planning (size and cost) are not addressed in ISO 9001.

The method we adopted, a hybrid multi-model assessment, is conducted by a small team of experienced, independent assessors following standard audit practices reflected in both ISO 9001 registration audits and the SEI CAF:

- Scope selection.
- Off-site document review.
- On-site interviews.
- Report preparation and delivery of findings.

By building on the well-defined relationship between the clauses of ISO

9001 and the Level 2 and Level 3 CMM KPAs, as described in Figure 1, the results of a single set of comprehensive interviews can be presented from both an ISO 9001 and a CMM perspective. To achieve the maximum impact from the report, the findings are presented twice in separate sections. Each section of the report is organized around one of the models. Each time the finding is presented, it includes detailed recommendations for action planning and points to the sections of the other model that address similar or identical requirements. To support overall action planning, the report concludes with a single set of priorities for implementation that define the assessors' view of a logical path through the most important of the detailed findings.

Considerations for Planning Hybrid Multi-Model Assessments

A hybrid multi-model assessment can be completed by a small, independent team in approximately 140 percent of the time required for an ISO 9001 registration assessment. Based on anecdotal information and on our experience in a number of CMM-based process assessments,[8] a hybrid multi-model assessment can be completed with 20 percent to 30 percent of the time and resources of an initial CBA/IPI.[9]

Although the proposed hybrid multi-model assessment does not include the team training that prepares the organization to act on the results of the assessment, there is implicit training in the interview process, especially when a value-added approach is used [14]. Employing collaborative assessment techniques, the hybrid multi-model assessment method is sponsored by the assessed organization and allows time in the interviews, in the entry meetings, and in general communications to solicit and deal with specific issues and concerns that might otherwise influence responses. The communications and interviews can be structured to reinforce three key principles:

- The purpose of the appraisal is to gather information to measure the completeness of the organization's practices (what you do, not what you are supposed to do) with respect to the models.
- The purpose of measuring is to improve the organization's ability to develop and deliver software to its customers.
- Any changes or new processes introduced as a result of the appraisal must support the way the organization does or wants to do business. This principle leads to at least three corollaries:
  - Members of the organization will be required to adopt the defined processes—and alert the appropriate people if there are problems with the documented procedures.
  - The ISO 9001 and the CMM models will be used to determine only what has to be done—not how it will be done.
  - Achieving compliance with the chosen model will be a byproduct of implementing effective and efficient processes that support the organization's business goals and objectives and that meet the needs of its employees.

The report derived from the assessment reinforces the similarity between the requirements both models place on a software engineering organization. The consolidated report adopts the value-added audit technique of including recommended actions, allows the champions of each particular model, who are frequently well-respected and influential engineers, to consider the other model and to leverage their experience. The recommended actions also facilitate post-assessment action planning.

After the Assessment

The assessments described in this article provide the information required to select a model—or combination of models—and to prepare a plan for effective process definition, implementation, and improvement. Achieving compliance with the chosen model is a byproduct of acting on the assessment findings.

Whether the organization's management chooses to go beyond compliance to become ISO registered or to complete a formal CBA/IPI or an SCE, it is critical that management acknowledge and respond systematically to the assessment findings. In the context of ISO 9001 and CMM Levels 2 and 3, the assessment findings define problems that are adversely affecting the organization's ability to perform—that are costing the organization time and money and creating unnecessary stress. If management fails to respond, the members of the organization will inevitably draw the obvious conclusions about management's commitment to its customers and to its employees. In fact, it would have been preferable not to have conducted the appraisal.

If management responds positively, there is no guarantee of success, but that positive response may launch the organization on a path to software process improvement that will lead to increased efficiency, capability, and competitive strength. ◆

About the Authors

Robert C. Bamford, a principal of Software Systems Quality Consulting, has a master's degree in mathematics and has managed training development, technical publications, professional services, and third-party software development. His over 30 years experience include the implementation of a Crosby-based Total Quality Management System, facilitating quality courses, managing education teams, and serving on a corporate quality council. He is an active member of the U.S. Technical Advisory Group for the ISO/IEC JTC1 SC7 – Software Engineering Standards subcommittee, which is responsible for the development and maintenance of ISO 12207 and ISO 15504 (SPICE). He and William Deibler jointly developed and published numerous courses, auditing and assessment tools, research papers, and articles on interpreting and applying the ISO 9000 standards and guidelines and the SEI CMM for Software.

William J. Deibler II, a principal of Software Systems Quality Consulting, has a master's degree in computer science and 20 years experience in the computer industry, primarily in software and systems development, quality assurance, and testing. He has extensive experience in managing and implementing CMM- and ISO 9001-based process improvement in software and hardware engineering environments. He is an active member of the U.S. Technical Advisory Group for the ISO/IEC JTC1 SC7 – Software Engineering Standards subcommittee, which is responsible for the development and maintenance of ISO 12207 and ISO 15504 (SPICE). He and Robert Bamford jointly developed and published numerous courses, auditing and assessment tools, research papers, and articles on interpreting and applying the ISO 9000 standards and guidelines and the SEI CMM for Software.

Software Systems Quality Consulting
2269 Sunny Vista Drive
San Jose, CA 95128
Voice: 408-985-4476
Fax: 408-248-7772
E-mail: ssqc@concentric.net
Internet: http://www.ssqc.com

References

1. Bamford, R.C. and W.J. Deibler, "A Detailed Comparison of the SEI Software Maturity Levels and Technology Stages to the Requirements for ISO 9001 Registration," Software Systems Quality Consulting, San Jose, Calif., 1992.
2. Bamford, R.C. and W.J. Deibler, "Exploring the Relationship Between ISO 9001 and the SEI Capability Maturity Model for Software Engineering Organizations," *Proceedings of the 1993 International Conference on Software Quality*, Lake Tahoe, Oct. 4-6, 1993, The Software Division of the American Society for Quality Control, p. 199.
3. Bamford, R.C. and W.J. Deibler, "Comparing, Contrasting ISO 9001 and the SEI Capability Maturity Model," *COMPUTER*, IEEE Computer Society, October 1993, p. 68.
4. Gilbert Associates (Europe) Limited, "TickIT Auditor Training Course," Issue 1, April 1992.
5. Hasse, Volkmar, Richard Messnarz, and Robert M. Cachia, "Software Process Improvement by Measurement," BOOTSTRAP/ESPRIT Project 5441.
6. "ISO 9001, Quality Systems – Model for Quality Assurance in Design/Development, Production, Installation, and Servicing, ISO 9001, International Organization for Standardization (ISO), Geneva, Switzerland, 1987, revised 1994.
7. "ISO 9000-3, Guidelines for the Application of ISO 9001 to the Development, Supply and Maintenance of Software," ISO, Geneva, Switzerland, 1991.
8. "ISO/IEC Guide 61, General Requirements for Assessment and Accreditation of Certification/Registration Bodies," 1st ed., Geneva, Switzerland, 1996.
9. "ISO/IEC Guide 62, General Requirements for Bodies Operating Assessment and Certification/Registration of Quality Systems," 1st ed., Geneva, Switzerland, 1996.
10. Kasse, Tim, and Wihalm Josef, "The Long Way to CMM Level 4," *Proceedings of the First World Congress for Software Quality*, American Society for Quality Control, June 1995.
11. Kuvaja, Pasi, et al., *Software Process Assessment and Improvement – The Bootstrap Approach*, Blackwell Publishers, Oxford, England, 1994.
12. Paulk, Mark C., "A Detailed Comparison of ISO 9001 and the Capability Maturity Model for Software," *Software*, January 1994.
13. Paulk, Mark C., et al., *Key Practices of the Capability Maturity Model*, Version 1.1, CMU/SEI-93-TR-25, SEI, CMU, Pittsburgh, Pa., March 1993.
14. Sayle, Allan J., *Management Audits – The Assessment of Quality Management Systems*, (ISBN 0-9511739-1-X), ASQC Press, 1989.
15. CMM-Based Appraisal Project, *Software Capability Evaluation, Version 2.0, Method Description*, CMU/SEI-94-TR-6, SEI, CMU, Pittsburgh, Pa., 1994.
16. Masters, Steven and Carol Bothwell, *CMM Appraisal Framework, Version 1.0*, CMU/SEI-95-TR-001, SEI, CMU, Pittsburgh, Pa., 1995.
17. *CBA Lead Assessor Training, Participant's Guide, Version 1.1a*, SEI, CMU, Pittsburgh, Pa., April 1997.

Notes

1. Documents associated with both models contain explicit statements of principle regarding lifecycle independence. For ISO [7], paragraph 5.0 states that the standard is "intended for application irrespective of the lifecycle model used." For the CMM [13], paragraph 4.3.5 states that "the key practices are not meant to limit the choice of a software lifecycle. ... There is no intent to encourage or preclude the use of any particular software lifecycle."
2. The requirements for verification assigned to the software quality assurance function by the CMM can be satisfied by ISO 9001 internal audits.
3. "Current and Potential Role of QSAR," *Quality Systems Update*, June 1995, p. 5 and "ISO Decides Fate of QSAR," *Compliance Engineering Newswatch*, July/August 1997 (http://www.ce-mag.com/isojul.html). At the time this article is being written, the IAF is seeking incorporation.
4. It is interesting to note that although ISO 9000 accreditation bodies do not consider training to be a consulting activity, QS9000 expressly forbids both training and consulting.
5. [1], [2], and [3] contain the background for our conclusion that the two models produce comparable results when adopted by a software engineering organization. One of the key differences is in the ability to extend ISO 9001 to all parts of an organization, e.g., marketing, sales, program management, systems engineering, and system test.
6. The results presented by the audit team are submitted to the registrar's "home office" for review before they become official.
7. [13], paragraph 4.1. It is of interest to note that there is an analogous statement in ISO 9001. In the Introduction, [6] states, "It is not the purpose of these [standards] to enforce uniformity of quality systems. ... The design and implementation of a quality system will be influenced by the varying needs of an organization, its particular objectives, the products and services supplied, and the processes and specific practices employed."
8. We have participated in CBA/IPIs and conducted Hybrid Maturity Model assessments in organizations ranging in size from 100 to more than 800 employees. This size estimate is also consistent with [10], page 15.
9. Note that the initial CBA/IPI considers all the Level 2 KPAs and a number of Level 3 KPAs.

Software Process Improvement: Eight Traps to Avoid

Karl E. Wiegers
Process Impact

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

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

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

Trap No. 1: Lack of Management Commitment

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

If you are leading the SPI effort, you might obtain senior management commitment, but get resistance from middle managers. In this case, you will be forced to spend time and energy debating the importance of SPI with people who should only have to be educated, not sold.

Such mixed signals from management make it hard for team leaders and software developers to take the effort seriously. Watch out for lip service and buzz words that masquerade as commitments. Lower-level managers who assign their least capable people (or no one) to the program send a clear signal that the lack of management commitment is about to foil your effort.

Solutions: Managers at all levels need to send consistent signals about SPI to their constituencies. Executives must be educated about the costs, benefits, and risks so they will have a common vision and understanding of this complex aspect of software engineering. Commitments need to be aligned along the organizational hierarchy, so that managers are not working at cross purposes, and a reluctant manager cannot sabotage the effort through inaction.

"Commitment" means more than hearing a manager say, "I'm fully behind this process improvement thing you're doing." Look for this tangible evidence of management commitment:

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

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

Trap No. 2: Unrealistic Management Expectations

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

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

software experience, may not appreciate the effort and time involved in a large-scale SPI effort, such as one based on the Software Engineering Institute's five-level Capability Maturity Model (CMM) [1]. These managers may be confused about how process improvement frameworks like the CMM relate to other software engineering approaches, such as a specific object-oriented development method. They may focus on issues of pressing importance to them that are not realistic outcomes of the process improvement effort. For example, a manager may hope to solve current staff shortages by driving the organization to reach CMM Level 2, which can lead to higher software productivity. However, since it can take two years or more to reach Level 2, this is not an effective solution to a near-term staffing problem.

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

Solutions: Educate your managers to help them understand the realities of what a serious process improvement initiative will cost and what benefits they might expect. Collect data from the software literature on results that have been achieved by other companies with effective improvement programs and the investments those companies made over a specified period [4-6]. Every organization is different, so it is risky to promise an eightfold return from each dollar invested just because you read that some company achieved that level of success. Use data from the literature or from other areas of your company to help your managers develop realistic expectations and set reasonable, even ambitious, goals. SPI is no more of a magic silver bullet than any other single software tool or technology.

Trap No. 3: Time-Stingy Project Leaders

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

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

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

Senior management must make it clear that project leaders will be evaluated on the effectiveness of their process improvement activities as well as on the success of the software projects. Software project planning needs to account for the resources being devoted to design and implement the new software processes. The first-level manager is the most critical factor in the success of any process improvement effort. If this person does not make SPI a visible priority, it is not going to happen.

One way to keep a program viable is to treat all process improvement activities as miniprojects, to give them the visibility and legitimacy they need for success. Write a short action plan for each miniproject. This plan identifies resources, states time lines, itemizes deliverables, clarifies accountability, and defines techniques to assess the effectiveness of new processes implemented as a result of each miniproject. Track the effort devoted to SPI to see if the investment level matches your planned commitment. Do not try to solve every process problem in your group at once. Instead, concentrate on the two or three top-priority items, as determined through some process assessment mechanism, then tackle the next few, and so on down the line.

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

Trap No. 4: Stalling on Action Plan Implementation

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

Solutions: As with Trap No. 3, a good way to turn action plans into actions is to treat improvement activities as miniprojects. You need to measure progress against the plans and to measure the impact of each action plan on the business results achieved. For example, a plan to improve the effectiveness of unit testing performed by the programmers might include an interim goal to acquire test automation tools and train developers in their use. These interim goals can be easily tracked. The desired business outcome of such an action plan should be a specific quantitative reduction, over some period, in the number of defects that slip through the unit testing quality filter.

If your project leaders never seem to make much progress against their action plans, you may need to implement a management oversight function to encourage them to take SPI more seriously. In one organization I know of, all project leaders must report the status of their action plans every three months to a management steering committee. When this occurs, the project leaders do not want to be embarrassed by reporting little or no progress on their plans.

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

Trap No. 5: Achieving a CMM Level Becomes the Primary Goal

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

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

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

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

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

Trap No. 6: Inadequate Training Is Provided

Symptoms: A process improvement initiative is at risk if the developers, managers, and process leaders do not have adequate skills and training. Each person involved must understand the general principles of SPI, the CMM, and other pertinent SPI methods, change leadership, software measurement, and related areas.

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

Solutions: Training to support established process improvement frameworks can be obtained from various commercial sources (such as process improvement consultants or training vendors), or you can develop such training. Different participants in the SPI activities will need different kinds of training. If you are using a CMM-based approach, the process improvement group members should receive several days of training on the CMM. However, four hours of training about SPI using the CMM will be enough for most participants. At Eastman Kodak Co., we developed a series of four-hour overview courses on various software engineering practice areas (requirements engineering and management, peer reviews, project planning and tracking, metrics, and configuration management) for project teams engaged in SPI.

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

Trap No. 7: Expecting Defined Procedures to Make People Interchangeable

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

Solutions: Individual programmers have been shown to have a ratio of 10-to-1, 20-to-1, or even higher range of performance (quality and productivity) on software projects [2, 3]. Process improvements alone can never equalize such a large range of individual capability. You can close the gap quite a bit by expecting people to follow effective defined processes rather than using whatever methods they are used to. This will enable people at the lower end of the capability scale to achieve consistently better results than they might get otherwise. However, never underestimate the importance of attracting, nurturing, and rewarding the best software engineers and managers you can find. Aim for software success by creating an environment in which all teammates share a commitment to quality and are enabled—through superior processes, appropriate tools, and effective team interactions—to reach their peak performance.

Trap No. 8: Failing to Scale Formal Processes to Project Size

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

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

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

Conclusion

As you chart a course to improve your software process capability, be aware of the many minefields lurking below your organization's surface. Your chances of success increase dramatically if you watch for the symptoms that identify these traps as a threat to your SPI program, and when you make plans to deal with them right away. Process improvement is succeeding at many companies. Make yours one of them by controlling these risks—and others—as well as you can. ◆

About the Author

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

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

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

References

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

Ten Things Your Mother Never Told You About the Capability Maturity Model

Margaret Kulpa
Abacus Technology

This article discusses the 10 most common misconceptions the author has had to overcome concerning software process improvement and the Software Capability Maturity Model. Topics include management vs. developer changes required, having standards in place, consensus vs. steamroller approaches, keeping it simple, and why you cannot expect software process improvement to work unless you give your employees time to do it.

Most organizations that start out on the road toward software process improvement (SPI) using the Capability Maturity Model (CMM) for Software have no clue what this endeavor means. Most managers get sold on the idea based on competitive practices within the industry—"keeping up with the Joneses." This article discusses some common misconceptions about the torturous path to achieving a maturity level.

"What, me change? You've got to be kidding!"

Managers think the CMM focuses on changing the way the developers work. What happens is that the CMM forces management to change the way it manages projects. By requiring the development team to collect project data and report it to management, management becomes more aware of the project management process. In some organizations, managers do not want to know in detail what is really happening on their projects. The idea that someone would report to them actual schedule slippages and try to determine a standard deviation becomes incomprehensible. It is not uncommon to shoot the messenger.

What the CMM really provides is the ability to shape your own destiny. By generating procedures to do work, you control your work environment. If management understood that, they probably would not start CMM activities.

"We can't do this. We have to support our users."

It is amazing how often supporting the users is used as an excuse to not do CMM work. The CMM absolutely advocates supporting your users. That is why we are in this business. In case we have forgotten—no users, no work. Ultimately, by following CMM guidelines, supporting the user becomes easier because the ground rules have been established.

Change involves not only the developers but also management and the user community. No matter your position, your attitude plays an important role in SPI. For example, the way you do work in your twenties should be different from the way you do work in your forties, or at least it should be based on learning. If you are still doing things the way you always have, you need to reexamine your work and probably your life—and you are probably not the best person to be put in charge of the improvement effort. CMM work is all about change, something such people apparently know nothing about.

Users also need to change. Your users do not have the right to kill you, but that is what they are doing to our aging work force by creating unnecessary stress that contributes to heart attacks, cancers, and other ills. Control is the real issue. People who believe they have some control over their lives tend to be happier and live longer (so say the psychologists). So, to gain control of your project, you must control your users. Why should you accept an "emergency" request at 4 p.m. Friday that will keep you at work all night? Especially when it turns out that that particular user *always* turns in an "emergency" request at 4 p.m. on Friday and does not need the information until later the following week? Those users need to be trained in becoming pro-active and basically getting their act together. If everything is an emergency, nothing is an emergency. This sounds like an area in need of improvement.

"Standards? We don't need standards!"

Nowhere in the CMM does it say that standards are required. The CMM does not absolutely *require* anything. The model is not a step-by-step how-to model—it is a framework, a guideline. It tells you *what* you need to do but not *how* to do it. However, the CMM presupposes that you have standards and are trying to follow them. The standards they presuppose you already have are for products like coding standards, templates for a requirements specification, or test case scenarios.

Following standards institutes a basic structure within an organization. So, if you do not have any standards, get some. One place to search is Department of Defense (DoD) military standards, even if you are not a DoD organization. Start searching the Web for military standards as well as for the methods used to implement SPI. They are available, and they are free.

Just do not be anal when you interpret this information (see item 10, "Keep It Simple"). And all standards should be tailored for use in your organization. Do not think that you can use the same standards you used from the place you used to work in your new workplace. They do not fit. They cannot be used. They can be used as a target, but you will need to tailor them.

"Everybody knows what the process is. What's the big deal?"

Everybody knows what a process is until they try to define it in detail and write procedures that describe how to follow the process. Then, they shift back to documenting *who* needs to do something rather than on *how* that something is done. They also fall back on product standards (a form for documenting defects found during peer reviews) instead of process standards (*how* to perform the peer review, *how* to detect defects, and *how* to complete the form). Telling me that "it is the project manager's responsibility to determine schedule estimates" does not tell me *how* that manager is supposed to derive those estimates.

"Collaborative and achieving consensus …"

CMM teams usually try to work collaboratively and make decisions by consensus. This concept is great and fosters buy-in and ownership but is extremely time-consuming and expensive. Consensus is *not* majority rules. Consensus means that everyone can live with the decision—they may not love it, but they can live with it. This way of working takes time. If you are on a tight schedule, (CMM work always is) you may need to stop the philosophizing and touchy-feely stuff and steamroll some folks. You will never get 100 percent buy-in from everyone. Take what you can get, and get those procedures written down.

"The CMM requires that a good process be in place."

No. It requires that a process be in place that is documented and followed. At first, your process could be awful. That is where the "continuous process improvement" concept comes in. After you hammer out a process, it is piloted, and projects start to use it, refinements will be made until (it is hoped) the process becomes "good." But to start, get something down on paper and use it. Clean it up as you go.

"We need to model our *as-is* process in order to create our *to-be* process."

Yes, but I find that organizations take up to a year to do this, only to find that their processes are too ad hoc to be used as a baseline of good practices and lessons learned. I suggest doing a software capability evaluation (which is now done for internal software process improvement) or a CMM-based Appraisal for Internal Process Improvement. These assessment methods can quickly determine consistent practices across the organization as well as strengths and weaknesses. Measurable action plans can be generated based on the results. Tracking progress can also be measured. The thing to remember before starting CMM activities is to determine ahead of time how to measure success. Modeling current processes is great—but will you ever see a return on that investment?

"Tie CMM activities to your business objectives."

Of course. There are some things in the CMM that may not make sense for you. For example, having a separate group to do software quality assurance (SQA) may not work if you only have 10 people in your company. The challenge is to figure out a way to perform quality assurance reviews and oversight in an objective, independent manner. And do not confuse "organization" with "company" or enterprise. An organization achieves a maturity level rating—not one project, not an entire company. Without going into detail, an organization *generally* consists of three to eight projects reporting to the same person, like a director or a division head—not an entire company (like IBM).

Do not get stupid about "business objectives." Ultimately, most organizations' business objectives are to achieve Level X by a certain date. If you are not currently doing SQA and do not want to do SQA (because of the cost and because it is overhead) yet you must achieve the level, do not try to be clever and tailor SQA out of the CMM process. Any certified evaluation team will catch you.

"Better, cheaper, faster."

This really irks me. When the CMM was written, most organizations had not yet begun the downsizing frenzy. Nowadays, however, organizations have cut their staff to the bare minimum. Management loves the maxim "better, cheaper, faster" and eventually, you will be able to turn out software of better quality, more quickly, and less expensively—*but not at first!* The average time to obtain your return on investment is three to five years.

SPI is expensive. Most organizations either hire outside consultants to start the journey or build it from the inside. Even if you are not hiring consultants, taking people away from coding, i.e., "real work," and having them do SPI costs you time, money, and schedule slippage. So management instead assigns SPI work in addition to existing work to an organization with extreme resource constraints, and it fails. You cannot squeeze additional effort from people who are already overworked. And having these people "work weekends, holidays, I don't care what it takes" violates the CMM principle of establishing and following *reasonable* plans.

"Keep it simple."

I like this one. Most organizations start off believing that they can keep their procedures simple—until they try to do it. Writing procedures that are simple and easy to follow, yet are thorough and complete, is extremely difficult. That is why the people on your teams need to be able to write and like to write as well as have a technical background and knowledge of the organization.

Managers in organizations today seem to feel that one person can wear many hats, i.e., a Powerbuilder programmer can also write procedures for how to write a requirements specification. Do you know what happens when you ask that unfortunate "techie" to do that? He breaks out in a cold sweat. Although some people are adaptable and can do many jobs, not everyone can do everything well. Different skillsets are required for different jobs.

Another problem is that teams often catch the improvement fever. They want to improve everything. The challenge is to stay focused and use the CMM for software as your guide, but do not attack more than you can handle at one time. Remember: SPI is continuous improvement. It is iterative. Do what you can do in the time allotted, then go back and pick out more things once you have been allocated more time to do them.

Conclusion

Although there are other points to ponder when attempting this journey down the CMM path, these are the most frequently found errors made that I have documented. Good luck on your journey. ◆

About the Author

Margaret Kulpa is a consultant with Abacus Technology Corp. in Chevy Chase, Md. She is a certified lead evaluator and is authorized to teach the SEI's Introduction to CMM and the Software Capability Evaluation class. She has performed SPI duties for over 15 corporations and has evaluated over 30 organizations. She has also written and taught Key Process Area classes for Levels 2 and 3.

Abacus Technology
5454 Wisconsin Ave., Suite 1100
Chevy Chase, MD 20815
Voice: 301-951-1712
Fax: 301-907-8508
E-mail: kulpamk@songs.sce.com

METRICS, from page 26

be performed in an action plan allows the Measurement Team and manager to track progress with respect to the implementation of the measurement activities. An outline for an action plan follows:

1.0 Objective.
2.0 Description.
2.1 Background.
2.2 Goals.
- Business Goals.
- Measurement Goals.
- The Goals of This Plan.

2.3 Scope.
2.4 Relationship to Other Software Process Improvement Efforts.
2.5 Relationship to Other Functional Activities.
3.0 Implementation.
3.1 Activities, Products, and Tasks.
3.2 Schedule.
3.3 Resources.
3.4 Responsibilities.
3.5 Measurement and Monitoring.
3.6 Assumptions.
3.7 Risk Management.
4.0 Sustained Operation.

As the measurement activities are being planned, be sure to consider how the quality and success of the measurement activities will be measured. Building the need to measure the quality and success of the measurement activities into the measurement processes will help keep the activities aligned with the needs of the organization and mitigate some of the more common reasons why measurement fails. These reasons include lack of use of the data, personnel not understanding why the data need to be collected, and measurement viewed as an expendable, overhead activity. Following the goal-driven process outlined above provides a means to involve stakeholders, create understanding, and make measurement a part of the way the organization conducts business. Maintaining alignment between the measurement activities and the information needs of the organization helps the organization leverage information, which may otherwise not be captured, to enhance its performance. In summary, the goal-driven software measurement process directs attention toward measures of importance rather than measures that are merely convenient. ◆

About the Author

Dave Zubrow is team leader for software engineering measurement and analysis for the SEI and is assistant director of analytic studies for Carnegie Mellon University. He is an Assocation for Software Quality Certified Software Quality Engineer and a member of the Software Division Council for the American Society for Quality Control. He has a bachelor's degree from Penn State University and a master's degree and a doctorate from Carnegie Mellon University.

Voice: 412-268-5243
Fax : 412-268-5758
E-mail: dz@sei.cmu.edu

References

1. Schiemann, William and John Lingle, "Seven Greatest Myths of Measurement," *IEEE Engineering Management Review,* Spring 1998, pp. 114-116.
2. Park, R., W. Goethert, and W. Florac, *Goal-Driven Software Measurement*, (CMU/SEI 96-HB-002) Software Engineering Institute, Carnegie Mellon University, 1996.
3. PSM96, *Practical Software Measurement: A Guide to Objective Program Insight*, Washington, D.C., Joint Logistics Commanders, Joint Group on Systems Engineering, March 1996.
4. Basili, V. and D. Weiss, "A Methodology for Collecting Valid Software Engineering Data," IEEE Transactions on Software Engineering, Vol. 10, No. 6, 1984, pp. 728-738.
5. Briand, L., C.M. Differding, and H.D. Rombach, "Practical Guidelines for Measurement-Based Process Improvement," Software Process Improvement and Practices, Vol. 2, 1996, pp. 253-287.
6. Park, Robert E., et al., *Software Size Measurement: A Framework for Counting Source Statements* (CMU/SEI-92-TR-20), Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., September 1992.
7. Florac, William A., et al., *Software Quality Measurement: A Framework for Counting Problems and Defects* (CMU/SEI-92-TR-22), Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., September 1992.
8. Goethert, W., et al., *Software Effort Measurement: A Framework for Counting Staff-Hours* (CMU/SEI-92-TR-21), Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., September 1992.

Why Coaches Are Needed in Software Process Improvement

Lewis Gray
Abelia Corporation

This article defines an important role in software process improvement—the coach. The coach is different from the familiar roles of sponsor, champion, and change agent. The coach combines expert knowledge of new practices and tools with expert leadership skills in change management. Whether formal or informal, skilled or not, coaches are everywhere in software process improvement. Despite this, little has been written about what they do. Organizations often confuse the coach with other roles and fail to use coaches well. Software process improvement teams will likely perform better when their coaches do, just as sports teams with good coaches usually perform better than teams with poor coaches. Organizations can benefit from a better understanding of what process improvement coaches can and should do for them.

By now, everyone in software process improvement (SPI) knows about sponsors, champions, and change agents [1,8,9]. Watts Humphrey states that these roles are needed to lead stakeholders through their resistance to process change [1]. Humphrey has also discussed the need for coaches [2], and this topic has been mentioned here and there in articles and meetings, for example, in Gray and Stephenson [3] and Gray [4]. Still, leaders of SPI initiatives often overlook the coach's role and end up working too hard. Where they are used, coaches are the spark plugs of the process improvement engine. It is time to clarify what they do so process improvement will happen faster and easier.

In a famous psychological study of death and dying, Elisabeth Kübler-Ross [5] described five predictable stages of grieving over tragic news: (1) denial and isolation, (2) anger, (3) bargaining, (4) depression, and (5) acceptance. Implementation Management Associates, Inc. [6] and the Software Engineering Institute both argue in their training that people subjected to process change go through the same emotional stages. These emotional reactions are the source of the resistance to change that sponsors, champions, and change agents must overcome.

Figure 1 summarizes the Kübler-Ross grieving cycle graphically. In the typical Kübler-Ross case, shown by the darker curve in the figure, the cycle ends at a lower level of happiness and productivity than where it began. She studied people who were terminally ill; for these people, a happier, more productive final state was not a possibility.

During software process improvement, an important rationale for doing change management has to be that if process change is properly managed, people's resistance will be shorter in duration and less intense, and that following their period of resistance they will end at a higher level of happiness and productivity than where they began. A sample grieving cycle for well-managed process change is shown by the lighter curve in Figure 1.

There are some effective change management techniques that can be taught, such as the use of the Software Engineering Institute's Techology Transition Model [7] and the roles of sponsors, champions, and change agents. However, organizations that plan software process improvement should also seek help from a successful SPI coach, because change management is a complex, sensitive task that must be tailored to each different process improvement effort.

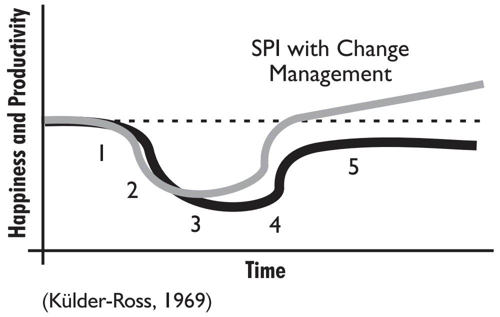
This article discusses activities that usually are necessary for successful software process improvement, distributed among four roles. Sponsors, champions and change agents are often ill-suited for the coach's activities. Following are all the roles in greater detail [1,8,9].

Sponsor

The sponsor's role is the easiest to describe. The sponsor acts like a banker who owns and donates resources to the process improvement effort for whatever reason the sponsor finds compelling, ranging from a bookkeeper-like attention to return on investment (ROI) to nothing more than a personal commitment to support the champion. The sponsor guarantees that the champion receives the resources needed for process improvement on time throughout the planned process improvement period.

A sponsor must have total control over the resources needed for the process improvement initiative. Only then can the sponsor guarantee that the champion will get them on time

Figure 1. *Grieving cycle with change management.*



(Külder-Ross, 1969)

as planned. Literally, the buck stops at the sponsor's desk. However, the sponsor typically does not need to know all the details of process change.

Champion

The SPI champion has three jobs: to initiate, promote, and protect SPI.

First, the champion initiates process improvement. This involves finding enough resources to do it properly and then generating so much enthusiasm for process improvement that the organization budgets for it. The champion usually recruits the SPI sponsor and coach and collaborates with the coach on an SPI action plan. The champion selects the change agents with the help of the coach.

Second, the champion promotes process improvement while it is under way. The champion is the public relations person for the effort. The champion justifies process improvement to colleagues and staff, and when the organization wants a presentation on SPI, perhaps for a proposal, the champion ensures it is done properly.

Third, the champion protects process improvement from detractors and poachers. Derogatory comments about process improvement stop at the champion, who defends the Process Improvement Team against detractors. Poachers are managers outside the process improvement effort who want to pull one or more of the SPI team members off the SPI initiative onto other projects that need help. The champion protects the integrity of the team against such poaching and defends the process improvement schedule.

Often, the role of champion is an informal one without a corresponding job description or funding. It is done in addition to the champion's regular duties. Typically, the champion does not need a detailed, implementer's understanding of the new practices and tools.

Change Agent

Change agents live and breathe the details of SPI. Their role is to make the changes that improve the organization's software process. Then, typically, they test the improved process on one or more example software development projects. These people are the players on the process improvement team. The sponsor and the champion could be compared to the team's owner and manager.

This is a tough, exciting job. The job is both a prize in itself and a burden. I recommend that it be assigned with care. Each Process Improvement Team member must learn to work with and trust every other member. Resistance to process change will appear in unexpected places in your organization, and the champion can never completely protect the team from it; so, change agents must cope with it. It helps if they are skilled and are respected by their peers. They should not be risk aversive, and they must not be heroic "prima donnas."

Change agents follow the action plan supplied by the coach. When they are criticized for what they are doing, they are protected by the champion. When they need planned resources, they get them from the sponsor by way of the champion. When they make mistakes, the coach corrects them. Change agents learn and implement all of the details of the new practices.

Coach

To date, unlike the other change management roles, the coach's role has not been defined in descriptions of SPI. To understand what the coach does, consider two key questions. First, who should prepare the process improvement action plan? Not the sponsor, champion, or the change agents, because one cannot assume they know enough details of SPI to properly plan it. Second, who corrects change agents when they make mistakes? Not the change agents, for obvious reasons, nor could the sponsor or the champion who, once again, may not know enough about SPI and change management to even spot a mistake.

In both cases, there is a clean fit to a fourth role in the process improvement effort—the SPI coach. An SPI coach is like a sports coach who tells the change agents what moves to make, but the change agents are the players. The coach plans the SPI effort, often in collaboration with the champion, corrects change agents when they make mistakes, and praises them when they perform well. In summary, the coach knows the improved practices and is skilled at introducing them.

A good coach is important to success. I recommend that an SPI coach finalize the action plan for process improvement and supervise implementation of the plan. Unlike the sponsor and the champion, the coach should be deeply involved in the implementation details of process change with the change agents. The coach will have the most influence on the change agents and should also serve as the champion's confidant and adviser on SPI. In some cases, the coach may play a similar role for the SPI sponsor.

The coach heads the change management effort and is most responsible for leading the other team members through the Kübler-Ross grieving cycle to a positive conclusion. It is easy to underestimate how difficult this is. Even SPI team members have occasional "allergic reactions" to what they are asked to do. The coach helps prevent and treat these types of reactions among stakeholders who are outside the team.

If you do not have a Software Engineering Process Group (SEPG) with an experienced coach, do not hesitate to bring one in from the outside. Keep in mind that for better or worse, someone will plan the software process improvement effort and guide the change agents. It is better if these roles are filled by people who are already knowledgable in the new practices and have the skills to successfully introduce new technologies.

Organizations that merely let coaching happen are likely to suffer from poor coaching. Unless someone knowledgeable is specifically assigned that role, there will be a series of SPI failures, as one would expect from a self-coached sports team.

Roles, Personnel, and Positions

Any of the above roles could be carried out by any number of personnel in the organization who may hold any of various positions. To understand how the roles interact, one must understand the differences between the terms "role," "personnel," and "position."

- *Role* – A collection of activities with one name, such as sponsor, champion, coach, or change agent.
- *Personnel* – Individual people in a project or its parent organization.
- *Position* – A box on an organizational chart for a project or parent organization. A position has a title assigned such as project manager, quality manager, SEPG director, Process Action Team (PAT) member, software engineer, or software requirements analyst.

These distinctions are helpful because, for example, personnel who hold two different positions in the same organization, such as a corporate vice-president and a midlevel manager, might both need to share the role of sponsor for an SPI effort to be successful. For example, the corporate sponsor might provide funds to the midlevel managers for the effort, but the managers might have authority to divert the funds to other more "important" projects if they can justify it. In such a case, a champion must also recruit the midlevel managers as sponsors if the funding from the corporate level is ever to reach the change agents on the Process Improvement Team.

Qualities of a Good Coach

What makes people good SPI coaches? These people share many of the same qualities as good basketball or football coaches. In a 1996 article in *The Washington Post*, Richard Justice [10] writes, "Coach inspires players with respect, honesty, and unrelenting drive." This coach was Jim Lynam of the Washington Bullets basketball team. Indiana Pacers Coach Larry Brown told Justice, "What that team has done is what everyone strives for in this business." According to Brown, "Jimmy Lynam has taken a group of players and gotten the absolute most out of them." When Justice tried to explain how Lynam did it, he was also describing qualities that good SPI coaches should have.

- Knowledge of the game and its strategies. Most good sports coaches also were once good players. In SPI, a good coach is likely to be someone who was a good software developer in an organization with a mature software process.
- A sense of humor. They can put work in the proper perspective. Excellence is not based on drudgery; it is based on fun.
- Honesty and straightforwardness.
- Inspire trust. This is critical to process improvement. Change agents cannot feel they must constantly second-guess the coach. Trust is built on honesty and success.
- Good communication. SPI coaches must be able to explain to change agents how to carry out the activities of a mature software process. Coaches must know when their explanations are getting through and when they are not. When team members do not understand them, coaches have to find another way to package the message so that it is understood.
- Respect for the team. A good SPI coach listens to team members when they raise an objection.
- No grudges. Richard said of Lynam: "If he chews out a player during a game—and he does it frequently—he has a one-on-one chat the next day to explain his actions." An SPI coach may have little control over who is chosen as SPI team members. The coach discovers a way, if there is one, to make the given team successful in improving the software process. Personal antagonism between the SPI coach and a team member, or between two team members, usually blocks long-term process improvement. The coach must find a way to avoid or defuse antagonistic situations.
- Negative reactions seen in the proper perspective. Coaches must understand that even their team members will have negative reactions to the pressures of change from time to time. The reactions are usually not directed personally at the coach, and the coach must not act as though they are unless they are.
- Focus on what *can* be done. I emphasize the word "can." Do not dwell on what cannot be done or waste time grieving about it. For example, there are many ways to accomplish each of the key practices in the CMM. Some practices, such as requirements management, software project planning, and software configuration management, can be much easier to perform with decent software tools than with pencil and paper. Nevertheless, pencil and paper are much underrated as tools. Organizations that cannot afford the proper software tools must learn a method that does not involve the desired tools—but they should not give up on the key practice. The coach must look for what the SPI team can do and show the team how to construct a mature software process that builds on that. An SPI coach never lets the SPI team give up as long as there is a reasonable probability of success.

Resources on Coaching

Readers who want to understand SPI coaching better can get a good start by reading Humphrey: "We have not yet developed a coaching ethic in software development. It could certainly help if we did. Sports and the performing arts have learned the value of coaching. … It seems unlikely that truly superior software development performance will be achieved without the help of skilled coaches." [2]

For further exploration of coaching as an organizational activity, see Curtis [11] and the SEI People Capability Maturity Model (P-CMM). Recognizing the value of coaches, SEI has placed coach development in the P-CMM as a Level 5 activity. Coach development is hard, but hiring a good coach is much easier. For most organizations, I recommend that you consider the key coaching activities in the P-CMM to be a checklist of what to look for in a prospective SPI coach.

Conclusion

In conclusion, the following is a quiz. Pick any ongoing SPI initiative (or software technology introduction effort) within your project or parent organiza-

TIS Achieves CMM Level 5

The Ogden Air Logistics Center, Software Engineering Division (OO-ALC/TIS) at Hill Air Force Base, Utah was assessed July 13-23, 1998 and found to be a Level 5 maturity organization according to the Software Engineering Institute Capability Maturity Model (CMM).

The Software Engineering Division, which comprises over 500 employees, develops and maintains software for operational flight programs and automatic test equipment. TIS is the first government agency known to be rated at this maturity level. Only three other companies involved in software development are known to share this rating.

The development of numerous tools, such as time and accounting systems, defect tracking databases, and a technology change management database helped TIS automate many of the activities relating to the goals in the Level 4 and Level 5 key process areas.

As a final self-check, TIS prepared cross matrices between their documentation and the goals, commitments, abilities, and activities associated with each key process area. These matrices provided a road map through the hierarchy of documentation. The projects within TIS also organized examples by each key process area. The seminar "Surviving a Software Capability Evaluation," presented at the April 1998 Software Technology Conference in Salt Lake City, Utah, reinforced TIS's belief in the need for this detail of preparation. This final check was also a benefit to the assessment team; it helped shorten the long days experienced by the assessment team members.

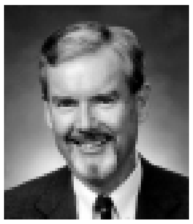
The assessment team consisted of nine members, six of which were either lead assessors or candidate lead assessors. The team consisted of Mark Paulk, Brian Larman, and Donna Dunaway from the Software Engineering Institute, Bonnie Bollinger and Millee Sapp from Robins Air Force Base, Ga., Mike Ballard from the Software Technology Support Center, and David Putman, Pat Cosgriff, and David Haakenson from the Software Engineering Division.

tion. Using the descriptions above of the four roles of sponsor, champion, coach, and change agent, and restricting your answers just to this initiative, can you answer:

- What personnel are the sponsors for that initiative? What positions do they have within your project or parent organization?
- What personnel are the champions for the initiative? What positions do they have within your project or parent organization?
- What personnel are the coaches for your initiative? What positions do they have within your project or parent organization?
- What personnel are the change agents for your initiative? What positions do they have within your project or parent organization?

Now ask yourself, if no one is filling one or more of these four roles, how will the activities associated with those roles be accomplished? Do you have one or more of the risks here in your initiative? Should you be tracking them in your risks matrix? ◆

About the Author



Lewis Gray, president of Abelia Corporation, has 30 years experience introducing new technology. He specializes in coaching and teaching CMM-based software process improvement. He was a leader in the development of IEEE/EIA 12207, MIL-STD-498, and J-STD-016 and is the only instructor outside the SEI authorized to present the Technology Transition Model for leading the adoption of new technology.

Prior to founding Abelia, Gray had key project management and technical positions at TRW, GTE, and INTELLIMAC. He received a bachelor's degree in mathematics and a doctorate in the philosophy of science (specializing in technology assessment) from Indiana University, where he also taught mathematics, technology assessment, and philosophy of science.

Abelia Corporation
12224 Grassy Hill Court
Fairfax, VA 22033-2819
Voice: 703-591-5247
Fax: 703-591-5005
E-mail: lewis@abelia.com
Internet: http://www.abelia.com

References

1. Humphrey, Watts S., *Managing the Software Process*, Addison-Wesley, Reading, Mass., 1989.
2. Humphrey, Watts S., *A Discipline for Software Engineering*, Addison-Wesley, Reading, Mass., 1995.
3. Gray, Lewis, and Dennis G. Stephenson, "Achieving Intergroup Coordination Through a Product Architecture Organization," *Proceedings of the Seventh Annual Software Technology Conference*, Salt Lake City, April 1995.
4. Gray, Lewis, "Defining the Coach's Role in Software Process Improvement," *Proceedings of the Tenth Annual Software Technology Conference*, Salt Lake City, April 1998.
5. Kübler-Ross, Elisabeth, *On Death and Dying*, Collier, New York, 1969.
6. "Implementing Change," Implementation Management Associates, Inc., Denver, Colo., 1989.
7. The Technology Transition Model (TXM), SEI workshop on "Introducing New Software Technology." For more information, see http://www.abelia.com/instcrse.htm.
8. Fowler, Priscilla and Stan Rifkin, *Software Engineering Process Group Guide*, Technical Report CMU/SEI-90-TR-24, Pittsburgh, Pa. Software Engineering Institute, 1990.
9. Rogers, Everett M., *Diffusion of Innovations*, New York, The Free Press, 4th ed., 1995.
10. Justice, Richard, "Bullets Take Heart from Lynam," *The Washington Post*, April 10, 1996.
11. Curtis, Bill, William E. Hefley, and Sally Miller, *People Capability Maturity Model*, SEI, Pittsburgh, Pa. 1995.

Real Process Improvement

Steve Neuendorf
Independent Management Consultant

Process improvement: The name belies its nature. Many people see it as the way to build products better, faster, and cheaper. In truth, there are many ways to be better, faster, and cheaper at whatever you are doing, and process improvement is only one of them. However, process improvement is the most reliable because first it works on people and only then does it work on the process. First you understand how good, fast, and cheap a process is and why it is that way. You also develop an understanding of why alterations of a process or adoption, as well as altogether different processes may be better, faster, or cheaper. With this understanding, you can then implement the changes that will result in improved performance.

Everyone agrees on what process improvement is, right? It is the most discussed topic in software engineering. It has been around for such a long time; there is no need to define anything before launching into a familiar discussion, right? Wrong.

Process Improvement Defined

To paraphrase W. Edward Deming, author of *Out of the Crisis*, process improvement is not something you talk about; it is something you do. For process improvement in software engineering, when all is said and done, a lot more is said than done. What is worse, far too much of what is done is done wrong—the process is not better, and process improvement is besmirched. Let us look at process improvement and see if there is a way to effectively and consistently improve software engineering processes.

Two things need to be understood: process and improvement. It may sound silly, but the number of different definitions and their imprecision lead to a great deal of confusion and misunderstanding about process improvement.

A prerequisite for process improvement is to have a comprehensive process definition. The operating definition of process is "the definition of the way in which something is intended to be done." Any process has an accomplishment objective. A process must describe the steps to achieve its objective and the means to do the steps. Since this definition could apply to many things, for software engineering, our process is assumed to have a formal definition so that it can be repeated.

Important Terms and Concepts

A variety of concepts must be understood before you can make the distinctions needed to initiate true process improvement.

Processes vs. Projects

Distinguish processes from projects: A process is the intended way to complete a project, whereas a project is the application of resources to that process. Because projects are tangible, the only way you can measure a process is to measure projects and use the information to make inferences about the process. When measuring, it is important to know what can affect processes and projects:

- Processes are affected by *common causes of variation*, e.g., teams, tools, environment. When you work on common causes, it is called "process improvement." Common causes should be addressed by your process.
- Projects can be affected by either common or *special causes of variation*, e.g., system failures, attrition, or improper schedule or budget. Working on special causes is (or should be) problem identification and correction, commonly called "fire fighting." Special causes cannot necessarily be prevented or addressed through process changes.

You must always distinguish which type of cause is at work, because if you treat a common cause with special cause techniques and tools (or vice versa), you are "meddling" rather than solving problems. Meddling invariably causes more "fires," and fire fighting does not improve processes.

Efficiency, Cycle Time, Quality

Three characteristics describe any process:

- Efficiency (E) – the relationship between resource use and accomplished results.
- Cycle Time (T) – the "design speed" of the process, i.e., the speed of a particular development process relative to other processes (assuming certain factors are equal).
- Quality (Q) – the quality of the process.

You may laugh when someone says "good, fast, cheap—pick two" but this is more insight than humor. For any process,

function $f(E,T,Q)$ = constant K.

Therefore, if you want better *and* faster *and* cheaper, you will need a different process.

Sequence and Means

Sequence and means are the two elements of a process. Sequence is simply the order in which things are accomplished. There are two types of sequences:

- Required – an order that *must* be followed, as demonstrated through precedence, i.e., putting on a second coat of paint requires that the first coat already be applied and cured. In software development, precedence has long shown it is best to start coding after the requirements are gathered.
- Discretionary – the order in which it is *decided* that something will be done. Past precedence and the availability of related means influence the sequence in which discretionary processes are executed.

After sequence comes the means by which the steps or tasks will be carried out. It is understood that a project is "the application of resources to a process to produce a result." People usually think of resources as labor, time, and money, but for understanding the process you must consider other resources such as tools, techniques, technology, and factors particular to "resources" such as the skill and experience of the team members. Therefore, the definition of means includes all aspects you should consider so that you can understand and improve processes. It is helpful to consider the means separately in categories such as management, technology, teams (or people), tools, techniques, and environment.

Capability and Capacity

For effective process improvement, an organization must consider all of its processes collectively. As defined so far, a process is a sequence and a set of means. By this definition, each organization would have a virtually infinite number of processes (possible combinations of E, T, and Q). Therefore, the additional dimensions of *capability* and *capacity* must be understood. Each possible combination of E, Q, and T defines a capability. The current ability of the organization to execute a capability defines capacity.

This is not as complicated as it may sound. For example, one of the means defined as "Red Team" is comprised of members with certain experience and knowledge. Red Team projects are done faster, better, and cheaper than "Blue" or "Green" Team projects. Red Team performance levels are a "capability." Because the organization may have only a limited number of employees qualified to form Red Teams, there is a limited capacity associated with the Red Team capability.

Innovation and Continuous Improvement

There are two categories of improvement: *continuous improvement* and *innovation*. Continuous improvement of processes is the systematic upgrading of lower capability means to give higher capacity at a higher capability. To use the prior example, continuous improvement of team capability would be to provide training to Blue and Green Team members so that more Red Teams can be formed.

Innovation is the introduction of new capability. Again, using the prior example, innovation would be training everyone in a new technique. Everyone, even the Red Team, would have a greater capability to execute the new technique. It is important to note that innovation usually introduces learning curve dynamics and a risk of failure to a greater extent than continuous improvement.

Following is another distinction between strategy and process improvement. Again, using our example, if you were to merely adopt a strategy to replace Green and Blue Team members with Red Team-qualified candidates, better values of performance (E, Q, and T) would be expected. However, Red Team-caliber candidates are expensive and much harder to find. Green and Blue Team members could be trained (their processes improved); however, process improvement requires understanding the process—you need to know what will improve a process and how much improvement is needed. For example, the cost of making your Blue and Green Team members perform like the Red Teams must first be determined, then the benefit of improving team performance can be discovered.

As in anything complex, what is "obvious" is not always true, and what is true is not always obvious. Only by improving your understanding of the process can you manage the risk of making changes that do not result in the desired improvement.

Tools for Process Improvement

For all of the above categorization, there are still two categories of process improvement that need to be considered when you apply the available process improvement tools. There are repetitive processes, such as most manufacturing, and there are nonrepetitive processes, such as software engineering.

Repetitive Processes

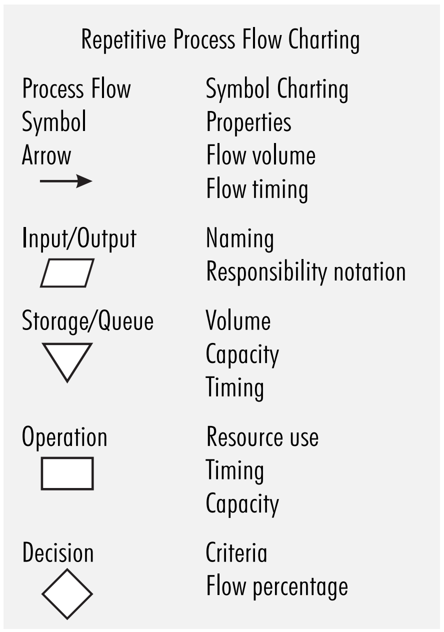
Virtually all of the common process and statistical process control (SPC) literature focuses on repetitive process principles and examples. Characteristics of these processes are a mostly fixed precedence, sequence, and means. Generally, the process flow-chart tool is used to understand and improve these processes, with care to use the decision element of the flow-chart tool to divide flow into segments that are distributed in such a manner that SPC tools can be used to analyze data (see Figure 1).

Nonrepetitive Processes

The goal in process improvement for software engineering is to improve nonrepetitive processes. Your understanding of these processes comes from analysis of the effects of variation in the means of production on performance. For nonrepetitive processes, attempts to understand and improve processes by using repetitive process tools and techniques will yield the same results as trying to teach a pig to sing—you get no singing, and it upsets the pig.

The key to understanding nonrepetitive processes is knowing that while the means may be similar from project to project, the steps and sequence can never be exactly the same. That is, even

Figure 1. *Flow charting without documenting the properties makes a nice picture but a poor tool for process understanding and improvement.*



Repetitive Process Flow Charting

Process Flow Symbol	Symbol Charting Properties
Arrow →	Flow volume Flow timing
Input/Output ▱	Naming Responsibility notation
Storage/Queue ▽	Volume Capacity Timing
Operation ▭	Resource use Timing Capacity
Decision ◇	Criteria Flow percentage

if you start a new project with the last project's teams (people and skills), management (people, styles, and leadership), and tools and techniques in the same environment, you are nevertheless dealing with a different project. However, there are myriad aspects of the means (management, teams, tools, techniques, environment, etc.) that can be measured in a way so the variation in these measures can be related to variations in performance (E, T, and Q) for any project.

However, at this point you do not necessarily yet understand the nature of your nonrepetitive processes. Notice in the process flow-chart tool description from Figure 1 that the volume property of the arrows, along with the "normalization" use of decisions, allows an analyst to "normalize out" variation due to throughput and size. For nonrepetitive processes, project cost, defects, and duration will vary due to differences in the means of completing the project *and* the (deliberately) heretofore not mentioned size factor—changing the size of a project can change everything.

I have not mentioned the size measurement because it seems to work like a light switch—flip the switch and about half the people turn on and the rest turn off. The goal of a size measure is to remove variation due to size from the analysis without introducing another source of variation and without removing a process-related source of performance variation. If your size measure does this (some straightforward statistical analysis will tell), you are on the right track.

An Example

To illustrate how this translates into reality, I use an example with which I assume most readers are familiar: the Software Engineering Institute (SEI) Capability Maturity Model (CMM). In a simplistic view, the CMM measures capability maturity on a scale of Level 1 to Level 5 with Level 5 being most mature. Several practitioners and analysts have also developed relationships between the CMM maturity level and performance in each dimension of process measurement (E, T, and Q), notably, the higher the maturity level, the better the performance. From the analyses I have seen, the differences in each E, T, and Q are substantial, and I assume they are statistically significant.

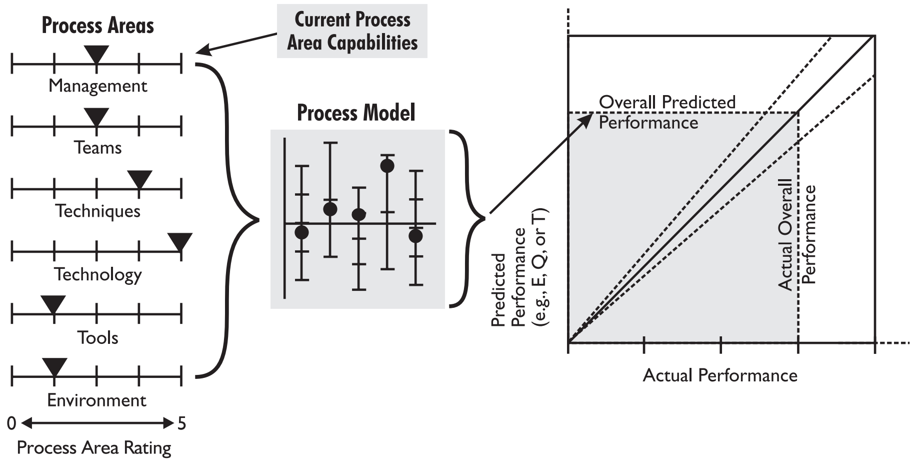
The nonrepetitive process improvement model is analogous to this view of the SEI CMM. That is, for each of the means (management, teams, tools, techniques, environment, etc.), a scale of possible conditions is developed, and criteria to evaluate that means according to that scale are prepared. For example, imagine project management is the means being evaluated. The low end of the scale would be no project management or maybe a project lead and no formal tracking. The top of the scale would be a full-time project manager who reports to the project management office and uses a full set of project management tools.

With all means identified and a scale created for each of the means, the values of the means for any given project constitute a process. Improvement of the process is the improvement of the means. Figure 2 shows how a process is managed to obtain a performance result. For the organization measured, the scale in each means, e.g., management, represents the range of influence of the measured capability on performance. For any project, the actual value of the mean (the triangle symbol) predicts the contribution of that aspect to the overall performance of the project. The process model shows the collective effect of the measured means values on project performance and gives the overall predicted project performance. The goal of performance measurement then becomes measuring actual performance against predicted performance to detect problems. Process improvement is working on the means, both for capability (higher levels on the scale) and for capacity (how much is available for use).

If you know what to do, calibrating the process improvement model can be straightforward, i.e., for any given set of means, values for E, Q, and T (exclusive of special causes) can be predicted. It follows that if the model is calibrated, the effect of changes in the means (benefits) can also be quantified. Finally, understanding the means makes improvement actions and investments evident and makes improvement cost (time and money) easily determined. You now have the action, cost, and benefit of process improvement objectively in front of you. Now more can be done than said.

It also is important to note that even in a calibrated model, it is likely that the variation in performance effect for

Figure 2. *Managing a process to obtain a performance result.*


Process Areas
Current Process Area Capabilities
Management
Teams
Techniques
Technology
Tools
Environment
0
5
Process Area Rating
Process Model
Overall Predicted Performance
Predicted Performance (e.g., E, Q, or T)
Actual Overall Performance
Actual Performance

Project Management is (1) identifying the process used (and predicted performance) and (2) assuring that the actual performance equals predicted performance.
Process Improvement is (1) improving your capacity at each higher level of capability (more projects at the higher expectation levels) and (2) increasing your capability in each of the process areas (dotted lines).

the lower levels of a means is much greater than for higher levels. That translates into the risk that the prediction at low levels will be off. In analysis theory, these are prediction anomalies called outliers, which are ignored. In process improvement, these must be accommodated. There are some stellar software producers at CMM Level 1; therefore, at Level 1, the model is not fully predictive. However, the overwhelming odds are that Level 1 performance will be worse than the performance of organizations at higher levels.

The real key to process improvement is hidden in the first phrase of a sentence two paragraphs back: "If you know what to do, ..." Process improvement is not intuitive ("*perfect* practice makes perfect"). At risk of confusing my explanation of process improvement, the process that most needs implementing is the one you use for process improvement. Once this process is good, the result will be an improved software engineering process. In other words, for process improvement to succeed, the skill level of the process improvement project team must be high, not the skill level in software engineering groups.

Most organizations do not have the requisite process improvement skills among their management or staff. No matter how motivated, facilitated, or well led, a team without the right skills is likely to fail to implement process improvement. If you are in a typical organization, process improvement has failed at least once. If you blamed the team, you were wrong—your expectations were unfounded. If you blamed process improvement merely because it is an art that is difficult to master, you were wrong, too—process improvement is alive and well and works great in the right hands. If you brought together the right resources in the right place at the right time with the right management and right leadership, you probably did not read this far. Good Luck. ◆

About the Author

Steve Neuendorf is an independent management consultant. He has over 25 years measurement and process improvement experience, with over 15 years in software engineering process improvement. He has a bachelor's, a master's, and a doctorate from the University of Puget Sound.

Voice: 425-557-8747
E-mail: steve@serv.net
Internet: www.serv.net/~steve

Need Assistance with Software Process Improvement?

Call the SPI Hotline at 801-775-5555 ext. 3055 DSN 775-775-5555 ext. 3055 or E-mail us at spi@stsc1.hill.af.mil.

We can answer questions about various software process improvement (SPI) issues, including

- How to get started on SPI.
- Available SPI training.
- Assessments.
- CMM key process areas.
- SPI best practices.
- SPI return on investment.

Software Technology Support Center (STSC) SPI veterans are on call to answer questions and research your problems for up to one hour without charge. We can provide you with policy, process, and procedure templates from our STSC library. If you need in-depth assistance, we will refer you to the appropriate experts using our database of prequalified consultants from the STSC and external sources.

Several SEPG members responsible for leading OO-ALC/TIS to its Level 5 status now work in the STSC. Call the SPI Hotline for their consulting services.

S T S C

Passing the CBA/IPI Torch

The Air Force Communications Agency (AFCA) at Scott Air Force Base, Ill. has transferred its responsibility for conducting Capability Maturity Model (CMM)-Based Appraisals for Internal Process Improvement (CBA/IPI) for Air Force organizations to the Software Technology Support Center (STSC) at Hill Air Force Base, Utah. The CBA/IPI is a method licensed by the Software Engineering Institute to assess an organization's capability to develop and maintain software. STSC consultants have experience in CMM-Based Appraisals that range from maturity Levels 2-5.

Organizations that need CBA/IPI assessments should call the STSC staff at 801-775-5555 ext. 3065 DSN 775-5555 ext. 3065 or E-mail spi@stsc1.hill.af.mil.

S T S C

Measurement with a Focus
Goal-Driven Software Measurement

Dave Zubrow
Software Engineering Institute

The collection of accurate metrics is a pointless exercise until the data is analyzed and used to predict and influence future events. The article discusses how to set up a metrics collection and analysis game plan that will advance specific business interests.

In a recent article by William Schiemann and John Lingle, they describe "Seven Greatest Myths of Measurement." [1] Among the points made in this article is the need to use measurement to anticipate the future rather than to merely record the past. This is the same perspective promoted by the Software Engineering Institute's (SEI) Goal-Driven Software Measurement process [2] and the Department of Defense initiative for Practical Software Measurement [3]. The benefit and value of software measurement come from the decisions and actions taken in response to analysis of the data, not from the collection of the data. I liken software measurement activities to potential and kinetic energy: Gathering the data creates a potential, but it takes analysis and action to make it kinetic. The Goal-Driven Software Measurement approach identifies 10 steps to establish a measurement program that is aligned with the organization's business processes. In this way, the risk of having data gathered, but not used, is minimized.

The steps of the approach are organized into three sets of activities: identifying goals, defining indicators and the data needed to produce them, and creating an action plan to guide the implementation. Business goals are translated into measurement goals [4, 5] by identifying high-level business goals and refining them into concrete, operational statements with a measurement focus. This refinement process involves probing and expanding each high-level goal to derive questions, the answers to which would help manage the organization. The questions provide concrete examples that can lead to statements that identify what type of information is needed.

However, a sense of what information is needed is not specific enough. The goal-driven approach requires that indicators, e.g., charts, tables, or other types of displays and reports, be sketched out and approved by the intended user. These indicators serve as a requirements specification for the data that must be gathered, the processing and analysis that must take place, and the schedule by which these activities should occur. The final set of activities takes the output of the preceding two sets of activities and uses them to develop an action plan. First, the existing data collection and measurement activities within the organization are analyzed to avoid duplication and identify gaps. Priorities, in terms of data to gather to produce the indicators, are assigned. Then, tasks are defined to take advantage of existing activities and to address the gaps. Part of the plan also addresses the need for the measurement activities to evolve with respect to staying synchronized with the organization's goals and to become more efficient and effective in its own operation. The following sections summarize each of the 10 steps. In summary, the Goal-Driven Software Measurement process consists of the following:

Identifying Goals

1. Identify your business goals.
2. Identify what you want to know or learn.
3. Identify your subgoals.
4. Identify entities and attributes related to your subgoals.
5. Formalize your measurement goals.

Defining Indicators

6. Identify quantifiable questions and the related indicators that you will use to help you achieve your measurement goals.
7. Identify the data elements that you will collect to construct the indicators that help answer your questions.
8. Define the measures to be used and make these definitions operational.

Creating an Action Plan

9. Identify the actions that you will take to implement the measures.
10. Prepare a plan to implement the measures.

Identifying Goals

Step 1: Identify Business Goals

The first step in identifying and defining software measures is to identify the business goals that drive your organization's efforts. If a strategic plan exists and is currently being followed, it can be used as a starting point. It is often worthwhile, however, to check the current commitment to the strategic goals. Without a clear sense of the organization's strategic goals and the objectives and responsibilities for each work unit or position, there is a risk that measures will not be aligned with important issues within the organization or used. To elicit goal statements, it is sometimes useful to ask a question such as, "What do we want to achieve?" Once the goals have been identified, they need to be prioritized. This is best done in a team setting with the relevant stakeholders participating.

Step 2: Identify What You Want to Know or Learn

If measurement activities are to be aligned with business goals, the goals must be translated into operational statements. The strategic actions planned and taken with the hope of meeting the goals provide proper targets for measurement. In this step, the goals are linked with knowledge of the organization's business strategies and processes. As illustrated in Figure 1, the questions related to the goals are framed in terms of the entities (work products or activities) and attributes (the size, effort to produce, or quality of an entity) associated with the organization's work processes. Oftentimes, the description of the work processes is in the form of a mental model rather than an explicit definition. If this is the case, it is worthwhile to identify the work products, activities, and other entities that offer opportunities for measurement. The key is taking the time to think through and document what you want to know about those entities with respect to the goals previously identified.

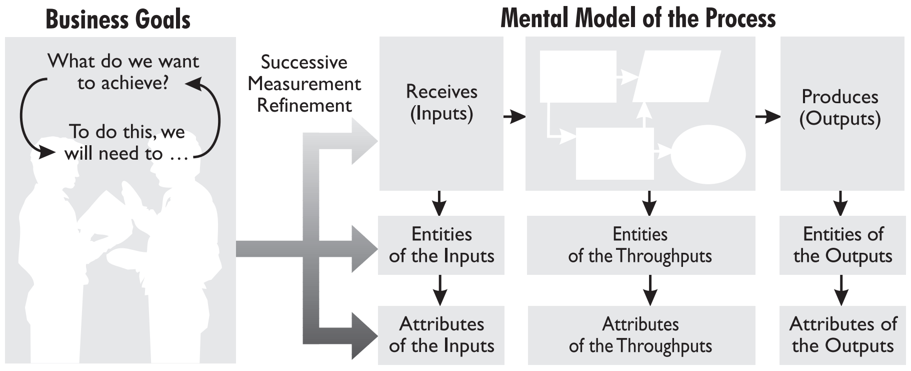

Business Goals
Mental Model of the Process
What do we want to achieve?
To do this, we will need to …
Successive Measurement Refinement
Receives (Inputs)
Produces (Outputs)
Entities of the Inputs
Entities of the Throughputs
Entities of the Outputs
Attributes of the Inputs
Attributes of the Throughputs
Attributes of the Outputs

Figure 1. Creating process according to business goals.

Step 3: Identify Your Subgoals

The preceding step usually generates many questions. Although they are stimulated by the top-level goal statement, the questions must be focused. By analyzing the questions and seeking commonality among them, subgoals can be derived. The subgoals provide a refinement of the goal and serve as a summary for the questions to which we would like answers. Subgoals are not directly derived from the goals to allow managers and other stakeholders the opportunity to brainstorm about the kinds of information they need with respect to their goals. This helps avoid the tendency to prematurely close the discussion on goals and the information that is needed. Similarly, the grouping and summarization of questions in this step provides a check that the question asked is related to an important dimension or subgoal of the original goal.

Step 4: Identify Entities and Attributes

In this step, attention is once again turned to the work processes of the organization. The questions from Step 2 are useful in this step as well. As noted, the opportunities to gather data and measure reside in the organization's work processes. The subgoals and related questions define the focus for the measures. Careful analysis of the questions will usually help identify what needs to be measured; for example, "How large is our backlog of customer change requests?" The entity in this question is the backlog of change requests and the attribute of interest is its size. Note that at this point, we can further ask in what way should size be measured. This point is addressed in the next step.

Step 5: Formalize Your Measurement Goals

In this step, a measurement goal is crafted that merges the purpose and perspective derived from the business goal with the possibilities for measurement as they exist within the organization's work processes. In addition, the goal statements express environmental or contextual factors that are important to understand for those who will design and do the measurement and analysis activities.

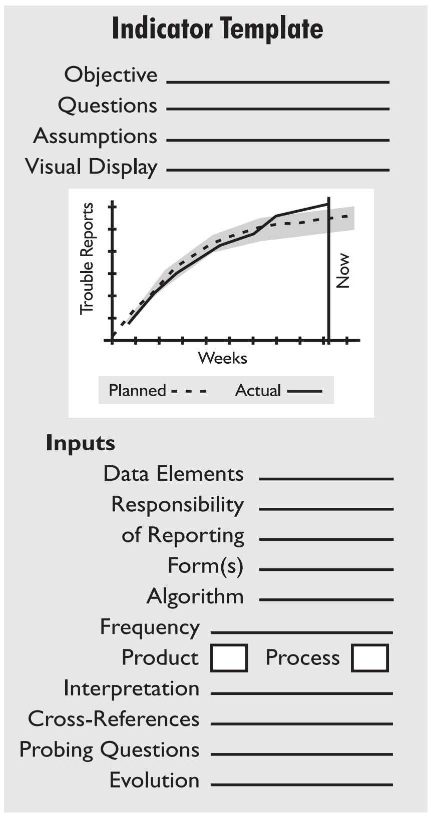
Well-structured measurement goals have four components:

- An object of interest (an entity).
- A purpose.
- A perspective.
- A description of the environment and constraints.

Note that in the four components, the perspective of who will use the information is explicitly documented. One means to ensure the information gathered will be used is to identify and document the user ("audience" is too passive) for the information.

As an example, the first five steps for defining goals might yield the following:

- Increasing customer satisfaction was identified as a business goal (Step 1).
- Questions asked about increasing customer satisfaction (Step 2) include
  - Do our products satisfy customer requirements?
  - Do we respond to customers in a timely manner?
  - Does our process ensure quality?
- From a set of the questions, a subgoal associated with requirements might be derived. The derived subgoal (Step 3) might be to increase the traceability between requirements and subsequent work products in the development process.
- The entities associated with the derived subgoal (Step 4) include the work products that define and validate the delivered product. The attribute of interest of the work products is the degree to which they address the requirements and perhaps the degree to which they only address the requirements. The latter captures the extent of "unrequired" features.
- Finally, we would address the derived subgoal with a formal measurement goal (Step 5), such as
  - Object of interest: The development process.
  - Purpose: Assess the degree of traceability of work products to requirements in order to control the scope of development efforts.
  - Perspective: Measure traceability of subsequent work products to requirements from the perspective of project managers.
  - Environment: New development project for military avionics. Process maturity at the site has been rated at the Repeatable Level of the CMM. Work products follow MIL-STD 2167A.


Indicator Template

Objective ____________
Questions ____________
Assumptions ____________
Visual Display ____________

Trouble Reports
Weeks
Now
Planned - - -
Actual ——

Inputs

Data Elements ____________
Responsibility of Reporting ____________
Form(s) ____________
Algorithm ____________
Frequency ____________
Product ☐ Process ☐
Interpretation ____________
Cross-References ____________
Probing Questions ____________
Evolution ____________

Figure 2. *Indicator Template.*

Defining Indicators

Step 6: Identify Quantifiable Questions and the Related Indicators

Armed with the measurement goal statement, indicators or displays to address the goal can be sketched out. Sketching or drafting the table, chart, or report that needs to be produced helps ensure the requirements for measurement are complete. In the course of designing the indicator, issues regarding the frequency of data gathering, the timing for generating the indicator, the need to use current and historical data, etc., surface. Similarly, the indicator also elicits whether the points on the chart, for instance, represent “raw” values, percentages, or some other derived scale. To a large extent, the indicator represents the product of the measurement activities. It is the consumable for the managers and practitioners who are looking for information to support their decisions and actions. Figure 2 shows an example of a template that can be used to document the definition, inputs, and use of an indicator. Continuing with the previous customer satisfaction example, an indicator such as the following might be created to answer the question, “What percentage of projects are producing traceability matrices between requirements and other work products?”

Step 7: Identify the Data Elements

The indicators reflect what data elements are needed. For instance, to produce the preceding indicator, the total number of projects per quarter and the number of projects having traceability matrices per quarter are required. Identifying the data elements, however, is not the same as defining them.

Step 8: Define the Measures

To continue the customer satisfaction example, definitions are needed for

- Projects.
- Criteria to determine whether they have traceability matrices, i.e., must they be reviewed prior to accepting them for this measure.
- How to assign projects to the periods for reporting, e.g., the quarter in which the project completes its design review.

These definitions are critical to achieve proper interpretations of the data. Note, however, that the definitions need to be created with the purpose of the indicator in mind; that is, they should be consistent with providing an answer to the question that the indicator addresses.

Developing a complete and unambiguous as possible definition can be arduous. To aid this task, the SEI developed a series of measurement framework checklists for common software measures such as size, effort, milestones, and defects [6-8].

Creating an Action Plan

Step 9: Identify the Actions for Implementation

Knowing the data needed and having defined them, the existing situation within the organization can be analyzed with respect to your measurement needs. Existing sources of the needed data should be identified. The data elements needed may be found in a variety of sources including project plans, defect tracking systems, the configuration management systems, and effort reporting systems. Likewise, data that is needed but is not available should be analyzed with respect to the amount of effort required to obtain the data. Considerations at this step include whether new forms, tools, or training would be required to obtain the data. Additionally, you must prioritize the currently unavailable data in terms of the indicators that depend upon the data. For each data element, you should determine its status with respect to the following:

- Does an explicit definition of the measure exist?
- Have the frequency of collection and the points in the process where measurements will be made been determined?
- Has the time line required to move measurement results from the points of collection to databases or users been established?
- Are there forms and procedures to collect and record the data?
- Have storage and access mechanisms and procedures been determined?
- Who is responsible to design and operate the database?
- Who will collect and who can access the data?
- How will the data be analyzed and reported? Who is responsible for the data, and who will receive the reports?
- Have the supporting tools been developed or acquired?
- Has a process guide to collect the data been developed?

In our example, reporting on the existence of traceability matrices may not exist. This gap would then be addressed in the action plan. For instance, to capture this data, the organization may need to add this to a project review or audit checklist for the Software Quality Assurance Group.

Step 10: Prepare an Action Plan

Once a gap analysis has been completed between the data needed and the existing measurement activities, prepare an action plan. Documenting the tasks to

See METRICS, Page 15

Using Test Cycles for Testing Year 2000 Projects

Randall W. Rice
Rice Consulting Services

A major challenge in enterprise-wide system testing is to devise a test process that simulates the operation of the organization over a period of time and that covers the processing of many systems. Enterprise-wide testing is extremely complex, and the need for such testing is particularly evident in most Year 2000 testing efforts. However, traditional testing processes often test software and systems at snapshots of time as opposed to testing transactions through specific checkpoints or cycles. The test-cycle approach in this article describes how to construct a set of tests that are controllable, repeatable, and measurable across any given span of time.

A variety of testing techniques are available that can easily be adapted and applied to Year 2000 (Y2K) projects. One technique that greatly helps plan, coordinate, document, and track testing is the test-cycle technique. In this technique, testing is organized and performed in cycles that can be defined to simulate specific dates. The great value in this approach is its application to large-scale enterprise systems. This kind of "end-to-end" test is especially critical in Y2K projects because of the need for testing interfaces between many different systems, both internal and external to the enterprise. This article presents an overview of the test-cycle concept, the benefits of using test cycles, and an example of how test cycles can facilitate Y2K testing.

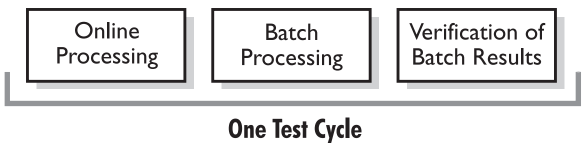
The examples presented in this article may appear simple because they are presented for the sake of illustration. In the real-world application of the test-cycle approach, the scope can be much larger but these techniques may still be used to manage the complexity of large-scale systems or user acceptance testing.

What Is the Test-Cycle Concept?

First, a test cycle is any defined period of testing. A test cycle could simulate a day, a week, a month, or no period. The ability to simulate a given period, however, is what makes test cycles an ideal technique for date-sensitive testing.

Exactly what happens during a test cycle depends on the technology involved. For example, in a traditional legacy mainframe environment, a test cycle usually consists of three parts: on-line data entry, batch processing, and the verification of batch results (Figure 1). In an environment that does not contain batch processing, the test cycle consists of interactive processing only. In a batch-only environment, a test cycle would consist of batch processing followed by the verification of batch results.

Figure 1. *Traditional test cycle.*



ID	Description	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6	Cycle 7
		12/15/1999	12/31/1999	1/1/2000	1/3/2000	1/15/2000	2/28/2000	2/29/2000

Figure 2. *Matrix headings with test-cycle dates.*

For each test cycle, a simulated processing period can be defined. That is why test cycles are an ideal way to plan and organize Y2K testing. One test cycle can be set for 12/15/1999, another cycle defined as 12/31/1999, another at 1/1/2000 and so on. The test environment date for each test cycle will need to be set using a date simulation tool.

The number of test cycles required for a test will depend on the amount of simulated time to be spanned during the test. For example, if you are merely testing the date rollover, you will only need a few cycles—probably 12/31/1999, 1/1/2000, and 1/3/2000 (the first Monday in 2000). However, if you are going to perform a more complete Y2K compliance test, you will need to define test cycles that allow a longer span of testing. For example, if you are testing a 30-day cancellation period across the century rollover, you might have one cycle defined as 12/15/1999 and another at 1/15/2000. You would also want other test cycles defined at 2/28/2000 and 2/29/2000 to test leap year processing. With the test-cycle approach and a date simulation tool and a data aging tool, you can define cycles as far in the future as you like. So, for testing leap year processing, you could also have cycles for 2/28/2004 and 2/29/2004.

Within each test cycle, one or more tests are defined to be performed. In some test cycles, it may be desirable to define no tests, depending on the cases being tested. The tests may be defined using test scripts or test cases.

The Process of Defining and Using Test Cycles

Now that test-cycle concepts have been discussed, let us look at the details of planning a Y2K test using test cycles.

Step 1 – Make Sure You Have the Right Tools

You will need a date simulation tool to easily change your test environment dates. You will also need a data aging tool to

ID	Description	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6	Cycle 7
		12/15/1999	12/31/1999	1/1/2000	1/3/2000	1/15/2000	2/28/2000	2/29/2000
1	DED = $500							
2	DED = $1.000							

Figure 3. *Test cycle matrix with cases.*

advance the dates in the test data and keep the relationships synchronized.

Step 2 – Define the Dates You Will Need to Simulate

These simulated system dates will depend upon the extent of your testing—namely, the levels of Y2K compliance you need to validate. There are four basic categories of Y2K compliance to consider:

- No value for the current date will cause interruption in operation. No matter what the system date is, the system will work correctly.
- Date-based functionality must behave consistently for dates prior to, during, and after 2000. All functions using dates as a basis should be correct. This includes calculations in the 19th, 20th, and 21st centuries, and calculations that span those centuries.
- In all interfaces and data storage, the century in any date must be specified either explicitly or by unambiguous algorithms. Either the century must be explicitly shown in the date, e.g., as a four-position field or by using a century indicator, or by using a logic routine to interpret the date based on a window of time or some other method.
- The year 2000 must be recognized as a leap year. If your system processes data from early in the 20th century, you need to be able to distinguish 1900 from 2000 for leap year purposes.

The dates that many people are using as system test dates at a minimum are

- 1/1/1999
- 9/9/1999
- 12/31/1999
- 1/1/2000
- 1/3/2000
- 1/4/2000
- 2/28/2000
- 2/29/2000

Your specific system dates will depend on your applications, business, and technology needs.

Figure 4. *Test cycle with test ID numbers.*

ID	Description	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6	Cycle 7
		12/15/1999	12/31/1999	1/1/2000	1/3/2000	1/15/2000	2/28/2000	2/29/2000
1	DED = $500	A001	A002	A003		A004	A005	A005
2	DED = $1.000	A001	A002	A003		A004	A005	A005

Step 3 – Build a Test-Cycle Matrix

Spreadsheets are great tools for this. You need to leave at least the first two columns blank for the test case identification (ID) and description, then define the test cycles along the top of the spreadsheet (Figure 2).

Step 4 – Define the Test Cases or Business Cases to Be Placed on the Matrix

Test cases and business cases are those entities you intend to test. These cases will go through one or more cycles of testing and will execute several test scripts or test scenarios. This approach to testing is what gives the test cycle concept so much power. You get to simulate not only the effect of the century rollover but also how people and things are processed through your systems from beginning to end. This is in contrast to merely testing one program at a time in a stand-alone fashion.

Figure 5. *Sample test script.*

Step	Program ID	Action	Expected Result	Observed Result	Pass/Fail
1	ACB001	Enter policyholder number and press <ENTER>.	Policyholder information displayed correctly.		
2	ACB001	With policyholder information displayed, press <f5>.	Control is transferred to billing screen (ACB002). Billing information correct for policyholder.		
3	ACB002	Press <f10>.	Exit to main menu.		

Some examples of test and business cases would be a policyholder, a customer, a patient, or a taxpayer. Each of these entities would then have attributes that would make it unique. For example, if you are testing policyholders, you might have one policyholder with a deductible of $500 and another with a $1,000 deductible (Figure 3). The number of test and business cases you include will depend on the level of test coverage you need relative to the risk involved.

Step 5 – Define the Test Order for Each Test or Business Case and Place in Correct Spreadsheet Cell

Each cell can contain a reference to a test or tests that are to be performed for a particular test and business case in a particular test cycle (Figure 4). You might decide to skip a cycle or two for some cases and double up or have several tests in other cycles. Once again, this is an example of how test cycles help you simulate the real world. Just like your live production databases were not instantly created in your business, the test data entered into the system cycle by cycle will continuously build. Keep in mind, however, that every test and business case added to the test will be one more item to maintain throughout the test.

Step 6 – Define the Tests in Detail

For every test indicated on the test-cycle matrix, a detailed description of the test will be needed for documentation both before and after the test. The details should include controls (such as when the test will start and stop), input, expected output, and the test procedure to be followed. An ideal way to

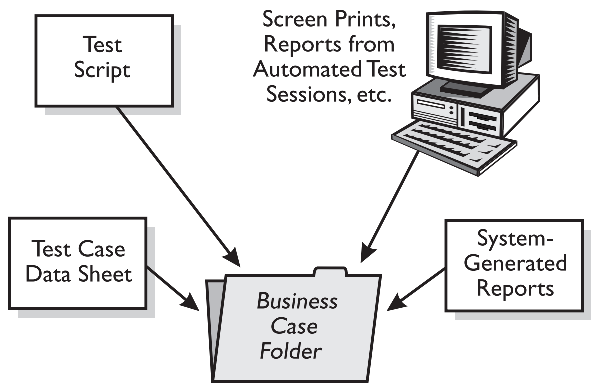


Figure 6. *Business case folder.*

document these aspects of a test for interactive software is to use a test script (Figure 5). You must determine how much detail is reasonable, given the amount of time you have left for testing and the relative business and technical risk.

Step 7 – Put It All Together

After using this method for many years, I have developed what seems to be a fairly smooth procedure to organize a major test based on the test-cycle concept. Although you can certainly automate testing using test cycles, many organizations do not have test automation tools in place and do not have the time to integrate a tool before starting Y2K testing. In addition, my surveys show that although 80 percent of the organizations I surveyed own an automated test tool, only about 25 percent of those organizations use the tool. For these reasons, the manual application of the test execution process will be shown.

First, you will need a manila folder for each test and business case you have defined. There will be a folder for each row on the matrix (spreadsheet). Label each folder with a business case ID number. This should also correspond to the ID on the matrix. Next, place everything you will need for the business case in the folder. This will include test data and test scripts or test procedures (Figure 6).

To simplify things and to find the right test information quickly, place a cover sheet (Figure 7) on the outside of the folder. The cover sheet shows the test cycles, the test scripts and the procedures performed in each test cycle, and a sign-off column to be initialed by the person who tests the business case.

The final piece is to get as many cardboard bankers' boxes as you have test cycles. If you have only a few folders per cycle, you can get by with using dividers in one or two boxes. You will need a way to start out the test with each set of folders separated by test cycle. Place the folders in the boxes by test cycle and in business case ID order. Each test cycle results in a new collection of these types of folders.

Figure 7. *Folder cover sheet.*

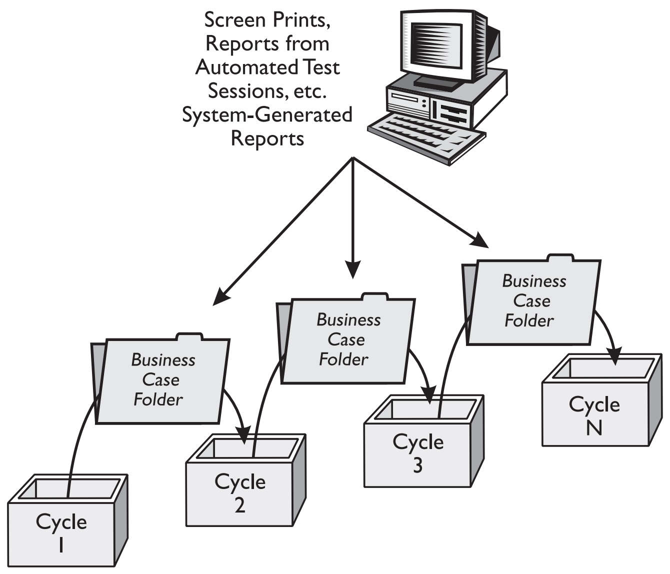
Cycle	Test Scripts	Tested	Verified
1	A001		
2	A002		
3	A003		
4			

Step 8 – Execute the Test

Start the test by setting the system date with the date simulator to the first test-cycle date. If a bed of test data will be used from the start, make sure the dates in the test data are correct.

Starting with the folders in the first cycle box, perform the tests in each folder for Cycle 1 only. During the test, you might create documentation you would like to save, such as screen prints or reports. These can be placed in the folder, unless the volume is large. In this regard, the test is self-documenting. When the test is complete, initial the folder in the "tested" and "verified" columns on the cover sheet, and place it in the next cycle in which it will be used. If batch processing is part of the test cycle or test procedure, the folder will go back into the same test-cycle division from which it was retrieved. After batch processing is complete, the folder can be pulled, evaluated, and moved on to the next cycle division in which it will be used (Figure 8).

Figure 8. *Test execution process using cycles.*



This process continues until the folder is finished and placed in a "done" box. Eventually, all of the business case folders will be filed in the done box in business case ID order. A year or two from now, if anyone needs to know what was tested, it is a simple matter to locate and retrieve the test documentation.

Step 9 – Evaluate and Track the Test

As the test is performed, you will evaluate the results and determine if the test passed or failed in that particular cycle. There are two effective and easy ways to keep track of test progress

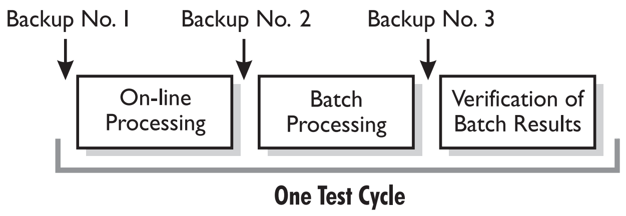


Figure 9. *Backups performed in test cycle.*

manually. One way is to use the outside cover sheet of the folder to indicate pass or fail. The other is to highlight each cell in the matrix as the test is completed and passed. It is good to use both methods.

The Key Benefits of Using Test Cycles

Although designing test cycles and business cases is extra work, there are some excellent benefits you achieve with no other test method that are especially important for Y2K testing.

- The ability to simulate a business case from point A to point Z in your processing. Most other test methods focus on one process or software module at a time, but never have a way to effectively string them together for end-to-end testing of a system or systems.
- The ability to plan and coordinate the march of time for a test. For Y2K testing, the tester knows that time must be advanced, but the problem is how to maintain synchronization among the test data, test environment, and test cases. The test-cycle concept allows you to do this with ease.
- A safety net in case the test environment gets corrupted. It is common in testing for the test to destroy data or update data files with incorrect information. It also is not uncommon for other people to delete or to restore over test files. The common response to this situation is to restore from the last backup, but how do you know what was tested since the last backup? In most test processes you do not know exactly what was done, but with test cycles, you *do* know. The backup process is fairly straightforward. You take image backups of the test environment before and after on-line input. If batch processing is part of your test, the backup taken after on-line processing will also suffice for the batch backup (Figure 9). These backups should be taken during each test cycle.

Conclusion

In testing, the confidence level of the test depends on the rigor and coverage of the test. The rigor and coverage of the test depends on the relative risk, both business and technical. While some might look at the work involved in planning a test using test cycles as being excessive, others will testify that this kind of effort is required on some projects and systems to validate their operation through multiple simulated dates. The extent of test planning and execution always depends on the scope of coverage and risk. The question is, are you willing to bet your business or systems operation on anything less than the right test method for the job? ◆

About the Author

Randall W. Rice is president of Rice Consulting Services, Inc. and has over 20 years experience building and testing large-scale information systems. He is a certified quality analyst and certified software test engineer specializing in systems testing and the testing of Y2K projects. He is the author of *The Year 2000 Testing Handbook*, creator of the "Testing the Year 2000" workshop, and co-author of *The Top Ten Challenges of Software Testing.* He also is chairman of the Quality Assurance Institute's annual International Software Testing Conference. He has worked with corporations and government agencies worldwide on Y2K testing issues.

Rice Consulting Services
P.O. Box 891284
Oklahoma City, OK 73189
Voice: 405-692-7331
Fax: 405-692-7570
E-mail: rcs@telepath.com
Internet: http://www.riceconsulting.com

Coming Events

Call for Papers: Software Engineering Laboratory Software Engineering Workshop

Dates: Dec. 2-3, 1998
Location: Goddard Space Flight Center, Md.
Topics: Software benchmarks, technologies, environments, standards, requirements capture and validation approaches, methods for safety-critical systems, reuse, COTS-based development (emphasis on process experiences, not products), automatic code generation, process improvement, and measures.
Abstracts (3-5 pages) should be directed to
SEB Abstracts Coordinator
Code 581
NASA/Goddard Space Flight Center
Greenbelt, MD 20771
E-mail (ASCII text only): Jackie Boger, jboger@cscmail.csc.com
Deadline for receipt of abstracts: Sept. 14, 1998
Internet: http://fdd.gsfc.gov/seltext.html.

Camden Technology Conference: The Transformation of Learning

Dates: Oct. 23-25, 1998
Location: Camden, Maine
Hosts: Bob Metcalfe, Tom DeMarco, and John Sculley.
Subject: The event will gather a faculty of experts from business, technology, government, and academia who will play a major role in shaping the learning methods and technologies of the coming century. Speakers include Alan Kay, Brenda Laurel, Seymour Papert, and Roger Schank.
Contact: 877-223-9752
Internet: http://www.camcon.org

CrossTALK

Sponsor	***Lt. Col. Joe Jarzombek*** 801-777-2435 DSN 777-2435 jarzombj@software.hill.af.mil
Publisher	***Reuel S. Alder*** 801-777-2550 DSN 777-2550 publisher@stsc1.hill.af.mil
Managing Editor	***Forrest Brown*** 801-777-9239 DSN 777-9239 managing_editor@stsc1.hill.af.mil
Senior Editor	***Sandi Gaskin*** 801-777-9722 DSN 777-9722 senior_editor@stsc1.hill.af.mil
Graphics and Design	***Kent Hepworth*** 801-775-5555 ext. 3027 graphics@stsc1.hill.af.mil
Associate Editor	***Lorin J. May*** 801-775-5555 ext. 3026 backtalk@stsc1.hill.af.mil
Editorial Assistant	***Bonnie May*** 801-775-5555 ext. 3022 customer_service@stsc1.hill.af.mil
Features Coordinator	features@stsc1.hill.af.mil
Customer Service	801-777-8045 custserv@software.hill.af.mil
Fax	801-777-8069 DSN 777-8069
STSC On-Line	http://www.stsc.hill.af.mil
CrossTalk On-Line	http://www.stsc.hill.af.mil/Crosstalk/crostalk.html
ESIP On-Line	http://www.esip.hill.af.mil

Subscriptions: Send correspondence concerning subscriptions and changes of address to the following address:

Ogden ALC/TISE
7278 Fourth Street
Hill AFB, UT 84056-5205

E-mail:	custserv@software.hill.af.mil
Voice:	801-777-8045 DSN 777-8045
Fax:	801-777-8069 DSN 777-8069

Editorial Matters: Correspondence concerning *Letters to the Editor* or other editorial matters should be sent to the same address listed above to the attention of *CrossTalk Editor* or send directly to the senior editor via the E-mail address also listed above.

Article Submissions: We welcome articles of interest to the defense software community. Articles must be approved by the CrossTalk editorial board prior to publication. Please follow the *Guidelines for CrossTalk Authors*, available upon request. We do not pay for submissions. Articles published in CrossTalk remain the property of the authors and may be submitted to other publications.

Reprints and Permissions: Requests for reprints must be requested from the author or the copyright holder. Please coordinate your request with CrossTalk.

Trademarks and Endorsements: All product names referenced in this issue are trademarks of their companies. The mention of a product or business in CrossTalk does not constitute an endorsement by the Software Technology Support Center (STSC), the Department of Defense, or any other government agency. The opinions expressed represent the viewpoints of the authors and are not necessarily those of the Department of Defense.

Coming Events: We often list conferences, seminars, symposiums, etc., that are of interest to our readers. There is no fee for this service, but we must receive the information at least 90 days before registration. Send an announcement to the CrossTalk Editorial Department.

STSC On-Line Services: STSC On-Line Services can be reached on the Internet. World Wide Web access is at http://www.stsc.hill.af.mil. The STSC maintains a Gopher server at gopher://gopher.stsc.hill.af.mil. Its ftp site may be reached at ftp://ftp.stsc.hill.af.mil. The Lynx browser or gopher server can also be reached using telnet at bbs.stsc.hill.af.mil or by modem at 801-774-6509 or DSN 775-3602. Call 801-777-7989 or DSN 777-7989 for assistance, or E-mail to schreifr@software.hill.af.mil.

Publications Available: The STSC provides various publications at no charge to the defense software community. Fill out the Request for STSC Services card in the center of this issue and mail or fax it to us. If the card is missing, call Customer Service at the numbers shown above, and we will send you a form or take your request by phone. The STSC sometimes has extra paper copies of back issues of CrossTalk free of charge. If you would like a copy of the printed edition of this or another issue of CrossTalk, or would like to subscribe, please contact the customer service address listed above.

The ***Software Technology Support Center*** was established at Ogden Air Logistics Center (AFMC) by Headquarters U.S. Air Force to help Air Force software organizations identify, evaluate, and adopt technologies that will improve the quality of their software products, their efficiency in producing them, and their ability to accurately predict the cost and schedule of their delivery. CrossTalk is assembled, printed, and distributed by the Defense Automated Printing Service, Hill AFB, UT 84056. CrossTalk is distributed without charge to individuals actively involved in the defense software development process.

I'm Level 5! (Technically)

A formal CMM appraisal isn't the freewheeling laughfest you might expect it to be. For the process team, it's like having Internal Revenue Service auditors camp in your kitchen for a couple of weeks to analyze six-year-old gas receipts and discarded Q-Tips, and to interview everyone including the neighbor's cat to see if every detail of your life is really what you say it is. Yet in the waning days of our recent appraisal, our software engineering process group leader wore a perma-grin that made me wonder if the strain had made him blow a gasket.

Then we heard the appraisal results: Level 5! Management is ecstatic, but they're also quick to emphasize that they've pushed CMM-based improvements all these years to achieve business objectives—not for the chance to gloat. With this in mind, let me emphasize that just because the nine-member appraisal team was led by the author of the CMM himself (Mark Paulk), and included reputably the toughest CMM appraiser around (Donna Dunaway), and just because they all agreed that all the teams and product lines from our entire 500+ software producers utilize Level 5 processes, that doesn't mean your processes aren't just as good or better than ours, although there's a 99 percent chance you'd be wrong. Neener neener neener.

So in this spirit of humility, I'll share my firsthand insight of what it takes to be a CMM Level 5 organization. First, there's preparation. To be fair, my preparation for this last appraisal was limited, but if I'd been interviewed, I would have been prepared: "That whaduyacallit—CMM thing—is for software, and I don't help produce any," I'd have said.

So if you want to split hairs, I suppose you could say that the Level 5 appraisal applies slightly more to everyone else in our division than to CrossTalk. But we're right there on the same organizational chart, clear as day. And that means I have insight from working in the same organizational *culture* as the developers. Not that I'm saying I know any of the *specific* processes our developers use—give me a break; most of our developers work in another building, and the closest ones in this building are at the end of a particle accelerator-length hallway that probably ends in another time zone—but I sometimes use the same vending machines they use, and I'm on a few of the same E-mail lists, most of which have to do with somebody-I-don't-know's retirement luncheon or whatever. And shouldn't anybody who knew these people well enough that they'd pay eight bucks and blow three hours in a restaurant waiting for one waitress to process everybody's credit cards already know about so-and-so's luncheon?

But that is not the point. Actually, I forgot what my point was. But this I know: I no longer put process improvement in the same category as "recreational bug tasting" or "accompanying my wife to the fabric store." As much as it amazes me to say it, I'm a process convert.

First, some background. My old attitudes came from work in the publishing field, which has a lot in common with software development. Both fields require extensive coordination and tracking, plus both involve mostly desks and computers and meetings—I mean, they're practically identical.

Anyway, my previous employers all relied on heroics to get the job done, and I liked being Mr. Heroic Stud Hombre. So when they sat us down here to get our processes in order, I thought they had no business documenting what (in my experience) was 100 percent unrepeatable. But we pushed ahead and laid out our roles and responsibilities, documented a process we could all live with, created metrics that actually told us things, and we've been tweaking the process ever since. And now that I've had a taste of both approaches, I'll take a good process over being Zorro the Firefighter and Bomb Defuser. Why waste my creative energies solving problems that a good process would take care of automatically?

So I guess the local process improvement culture did waft down the particle accelerator tunnel into this office. And in case you were wondering, working for a Level 5 organization is still a lot like working for any other organization, if you don't count the luminescent glow emanating from the building and the engineers' faces—or at least the faces of us converts. – Lorin May

Got an idea for BackTALK? Send an E-mail to backtalk@stsc1.hill.af.mil.