Technical Report
CMU/SEI-92-TR-007
ESC-TR-92-007

Introduction to Software Process Improvement

Watts S. Humphrey

June 1992 (Revised June 1993)

Technical Report
CMU/SEI-92-TR-007
ESC-TR-92-007
June 1992 (Revised June 1993)

Introduction to Software Process Improvement

Watts S. Humphrey

Process Research Project

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This report was prepared for the

SEI Joint Program Office
HQ ESC/AXS
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER

(signature on file)

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

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

This work was created in the performance of Federal Government Contract Number F19628-95-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 52.227-7013.

This document is available through Research Access, Inc., 800 Vinial Street, Pittsburgh, PA 15212.
Phone: 1-800-685-6510. FAX: (412) 321-2994. RAI also maintains a World Wide Web home page. The URL is http://www.rai.com

Copies of this document are available through the National Technical Information Service (NTIS). For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161. Phone: (703) 487-4600.

This document is also available through the Defense Technical Information Center (DTIC). DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center, Attn: FDRA, Cameron Station, Alexandria, VA 22304-6145. Phone: (703) 274-7633.

Table of Contents

List of Figures

Author's Note

This version of CMU/SEI-92-TR-7 has been produced to reflect the current SEI assessment practice of holding a five day on-site period. The reference to the earlier four day period had resulted in some reader confusion. While making this update, I also took the opportunity to bring the assessment and SCE data up to date and make a few minor corrections.

Introduction to Software Process Improvement

Abstract: While software now pervades most facets of modern life, its historical problems have not been solved. This report explains why some of these problems have been so difficult for organizations to address and the actions required to address them. It describes the Software Engineering Institute's (SEI) software process maturity model, how this model can be used to guide software organizations in process improvement, and the various assessment and evaluation methods that use this model. The report concludes with a discussion of improvement experience and some comments on future directions for this work.

1 Introduction

The Software Process Capability Maturity Model (CMM) deals with the capability of software organizations to consistently and predictably produce high quality products. It is closely related to such topics as software process, quality management, and process improvement. The drive for improved software quality is motivated by technology, customer need, regulation, and competition. Although industry's historical quality improvement focus has been on manufacturing, software quality efforts must concentrate on product development and improvement.

Process capability is the inherent ability of a process to produce planned results. A capable software process is characterized as mature. The principle objective of a mature software process is to produce quality products to meet customers' needs. For such human-intensive activities as software development, the capability of an overall process is determined by examining the performance of its defined subprocesses. As the capability of each subprocess is improved, the most significant causes of poor quality and productivity are thus controlled or eliminated. Overall process capability steadily improves and the organization is said to mature.

It should be noted that capability is not the same as performance. The performance of an organization at any specific time depends on many factors. While some of these factors can be controlled by the process, others cannot. Changes in user needs or technology surprises cannot be eliminated by process means. Their effects, however, can often be mitigated or even anticipated. Within organizations, process capability may also vary across projects and even within projects. It is theoretically possible to find a well controlled and managed project in a chaotic and undisciplined organization, but it is not likely. The reason is that it takes time and resources to develop a mature process capability. As a consequence, few projects can build their process while they simultaneously build their product.

The term maturity implies that software process capability must be grown. Maturity improvement requires strong management support and a consistent long-term focus. It involves fundamental changes in the way managers and software practitioners do their jobs. Standards

are established and process data is systematically collected, analyzed, and used. The most difficult change, however, is cultural. High maturity requires the recognition that managers do not know everything that needs to be done to improve the process.

The software professionals often have more detailed knowledge of the limitations of the processes they use. The key challenge of process management is to continuously capture and apply this knowledge.

Process maturity improvement is a long-term incremental activity. In manufacturing organizations, the development and adoption of effective process methods typically has taken 10 to 20 years. Software organizations could easily take comparable periods to progress from low to high process maturity. U.S. management is generally impatient for quick results and not accustomed to thinking in terms of long-term continuous process improvement. Once the cultural, organizational, and managerial obstacles are removed, the needed technical and procedural actions can often be implemented quite quickly. While history demonstrates that such changes can take a long time, it also demonstrates that, given a great enough need and broad consensus on the solution, surprising results can be produced quite quickly. The challenge is thus to define the need and achieve broad consensus on the solution. This is the basic objective of the Software Process Maturity Model and the Software Process Assessment method.

The focus on the software process has resulted from a growing recognition that the traditional product focus of organizational improvement efforts has not generally had the desired results. Many management and support activities are required to produce effective software organizations. Inadequate project management, for example, is often the principle cause of cost and schedule problems. Similarly, weaknesses in configuration management, quality assurance, inspection practices, or testing generally result in unsatisfactory quality. Typically, projects do not have the time nor resources to address such issues and thus a broader process improvement focus is required.

It is now recognized that traditional engineering management methods work for software just as they do for other technical fields. There is an increasing volume of published material on software project management and a growing body of experience with such topics as cost and size estimation, configuration management, and quality improvement. While hardware management methods can provide useful background, as shown in Table 1-1, there are key differences between hardware and software. Hardware managers thus need to master many new perspectives to be successful in directing software organizations.

Software is generally more complex.

Software changes appear relatively easy to make.

Many late-discovered hardware problems are addressed through software changes.

Because of its low reproduction cost, software does not have the natural discipline of release to manufacturing.

Software discipline is not grounded in natural science and it lacks ready techniques for feasibility testing and design modeling.

Software is often the element that integrates an entire system, thus adding to its complexity and creating exposures to late change.

Software is generally most visible, thus most exposed to requirements changes and most subject to user complaint.

Because software is a relatively new discipline, few managers and executives have sufficient experience to appreciate the principles or benefits of an effective software process.

Figure 1-1: Hardware and Software Differences

The Shewhart cycle provides the foundation for process improvement work. As shown in Table 1-2, it defines four steps for a general improvement process [Deming 82].

1. Plan
   Define the problem
   Establish improvement objectives
2. Do
   Identify possible problem causes
   Establish baselines
   Test change
3. Check
   Collect data
   Evaluate data
4. Act
   Implement system change
   Determine effectiveness.

Figure 1-2: The Shewhart Improvement Cycle

The cycle begins with a plan for improving an activity. Once the improvement plan is completed, the plan is implemented, results are checked, and actions taken to correct deviations. The cycle is then repeated. If the implementation produced the desired results, actions are taken to make the change permanent. In the Software Engineering Institute's (SEI) process strategy, this improvement plan is the principle objective of a Software Process Assessment.

The Shewhart approach, as espoused by W. E. Deming, was broadly adopted by Japanese industry in the 1950s and 1960s. The key element of the remarkable success of Japanese industry has been the sustained focus on small incremental process improvements. To enroll the employees in the improvement effort, quality control circles were formed and given considerable authority and responsibility for instituting change. Japanese management's basic strategy, followed to this day, is to focus on quality improvement in the belief that the desired productivity and profit improvements will naturally follow.

Based on these principles, the Software Process Maturity Model was designed to provide a graduated improvement framework of software capabilities. Each level progressively adds further enhancements that software organizations typically master as they improve. Since some capabilities depend on others, it is important to maintain an orderly improvement progression. Because of its progressive nature, this framework can be used to evaluate software organizations to determine the most important areas for immediate improvement. With the growing volume of software process maturity data, organizations can also determine their relative standing with respect to other groups.

This technical report describes the background of the Software Process Maturity Model: what it is, where it came from, how it was developed, and how it is being improved. Its major applications are also described, including the users, application methods, and the general state of software practice. Future developments are then described, including improvement trends, likely application issues, and current research thrusts. Finally, the conclusion outlines the critical issues to consider in applying these methods.

2 Background

With the enormous improvements in the cost-performance of computers and microprocessors, software now pervades most facets of modern life. It controls automobiles, flies airplanes, and drives such everyday devices as wrist watches, microwave ovens, and VCRs. Software is now often the gating element in most branches of engineering and science. Our businesses, our wars, and even our leisure time have been irretrievably changed. As was demonstrated in the War in the Middle East, “smart” weapons led to an early and overwhelming victory. The “smart” in modern weapons is supplied by software.

While society increasingly depends on software, software development's historical problems have not been addressed effectively. Software schedules, for example, are uniformly missed. An unpublished review of 17 major DoD software contracts found that the average 28 month schedule was missed by 20 months. One four year project was not delivered for 7 years; no project was on time. Deployment of the B1 bomber was delayed by a software problem and the $58 billion A12 aircraft program was cancelled partly for the same reason. While the military has its own unique problems, industry does as well. In all spheres, however, one important lesson is clear: large software content means trouble.

There are many reasons for this slow rate of improvement. Until recently, software project management methods have not been defined well enough to permit their inclusion in university curricula. They have thus generally been learned by observation, experience, or word of mouth. Second, few managers have worked in organizations that effectively manage software. This lack of role models means these managers must each learn from their own experiences. Unfortunately, these experiences are often painful and the managers who have learned the most are often associated with failed projects. By searching for an unblemished hero who can “clean up the mess,” management generally picks someone who has not been tested by a challenging software project. Unfortunately, this generally starts another disastrous learning cycle.

The most serious problems in software organizations are not generally caused by an individual manager or software team. They typically concern organizational procedures and cultural behavior. These are not things that individual managers can generally fix. They require a comprehensive and longer term focus on the organization's software process.

3 Process Maturity Model Development

The U.S. Department of Defense recognized the urgency of these software issues and in 1982 formed a joint service task force to review software problems in the U.S. Department of Defense. This resulted in several initiatives, including the establishment of the Software Engineering Institute (SEI) at Carnegie Mellon University, the Software Technology for Adaptable Reliable Systems (STARS) Program, and the Ada Program. Examples of U.S. industrial efforts to improve software practices are the Software Productivity Consortium and the early software work at the Micro-Electronics and Computer Consortium. Similar initiatives have been established in Europe and Japan, although they are largely under government sponsorship.

The Software Engineering Institute was established at Carnegie Mellon University in December of 1984 to address the need for improved software in U.S. Department of Defense operations. As part of its work, SEI developed the Software Process Maturity Model for use both by the Department of Defense and by industrial software organizations. The Software Capability Evaluation Project was initiated by the SEI in 1986 at the request of the U.S. Air Force.

The Air Force sought a technically sound and consistent method for the acquisition community to use to identify the most capable software contractors. The Air Force asked the MITRE Corporation to participate in this work and a joint team was formed. They drew on the extensive software and acquisition experience of the SEI, MITRE, and the Air Force Electronic Systems Division of Hanscom Air Force Base, Ma. This SEI-MITRE team produced a technical report that included a questionnaire and a framework for evaluating organizations according to the maturity of their software processes [Humphrey 87]. This maturity questionnaire is a structured set of yes-no questions that facilitates objective and consistent assessments of software organizations. The questions cover three principal areas.

1. Organization and resource management. This deals with functional responsibilities, personnel, and other resources and facilities.
2. Software engineering process and its management. This concerns the scope, depth, and completeness of the software engineering process and the way in which it is measured, managed, and improved.
3. Tools and technology. This deals with the tools and technologies used in the software engineering process and the effectiveness with which they are applied. This section is not used in maturity evaluations or assessments.

Some sample questions from the maturity questionnaire are:

- Is there a software engineering process group or function?
- Is a formal procedure used to make estimates of software size?
- Are code and test errors projected and compared to actuals?

There have been many contributors to this work and many companies have participated in early questionnaire reviews. The basic ideas behind the maturity model and the SEI assessment process come from several sources. The work of Phil Babel and his associates at Aeronautical Systems Division, Wright Patterson Air Force Base, provided a useful model of an effective Air Force evaluation method for software-intensive acquisitions. In the early 1980s, IBM initiated a series of assessments of the technical capabilities of many of its development laboratories. An IBM software quality and process group then coupled this work with the maturity framework used by Phil Crosby in his Quality College [Crosby 79]. The result was an assessment process and a generalized maturity framework [Radice 85]. SEI then extended this work to incorporate the Deming principles and the Shewhart concepts of process management [Deming 82]. The addition of the specific questions developed by MITRE and SEI resulted in the Software Capability Maturity Model [Humphrey 87, Humphrey 89].

As part of its initial development, SEI and MITRE held many reviews with individuals and organizations experienced in software development and acquisition. During this process it became clear that software is a rapidly evolving technology and that no static criteria could be valid for very long. Since this framework must evolve with advances in software technology and methods, the SEI maintains a continuing improvement effort. This work has broad participation by experienced software professionals from all branches of U.S. industry and government. There is also growing interest in this work in Europe and Japan.

In 1991 SEI produced the Capability Maturity Model for Software (CMM) [Paulk 91]. This was developed to clarify the structure and content of the maturity framework. It identifies the key practice areas for each maturity level and provides an extensive summary of these practices. To insure that this work properly balances the needs and interests of those most effected, a CMM Advisory Board was established to review the SEI work and to advise on the suitability of any proposed changes. This board has members from U.S. industry and government.

4 The Process Maturity Model

The five-level improvement model for software is shown in Figure 4-1. The levels are designed so that the capabilities at the lower levels provide a progressively stronger foundation on which to build the upper levels.

These five developmental stages are referred to as maturity levels, and at each level, the organization has a distinct process capability. By moving up these levels, the organization's capability is consistently improved.

At the initial level (level 1), an organization can be characterized as having an ad hoc, or possibly chaotic, process. Typically, the organization operates without formalized procedures, cost estimates, and project plans. Even if formal project control procedures exist, there are no management mechanisms to ensure that they are followed. Tools are not well integrated with the process, nor are they uniformly applied. Change control is generally lax and senior management is not exposed to or does not understand the key software problems and issues. When projects do succeed, it is generally because of the heroic efforts of a dedicated team rather than the capability of the organization.

An organization at the repeatable level (level 2) has established basic project controls: project management, management oversight, product assurance, and change control. The strength of the organization stems from its experience at doing similar work, but it faces major risks when presented with new challenges. The organization has frequent quality problems and lacks an orderly framework for improvement.

At the defined level (level 3), the organization has laid the foundation for examining the process and deciding how to improve it. A Software Engineering Process Group (SEPG) has been established to focus and lead the process improvement efforts, to keep management informed on the status of these efforts, and to facilitate the introduction of a family of software engineering methods and technologies.

The managed level (level 4) builds on the foundation established at the defined level. When the process is defined, it can be examined and improved but there is little data to indicate effectiveness. Thus, to advance to the managed level, organizations must establish a set of quality and productivity measurements. A process database is also needed with analysis resources and consultative skills to advise and support project members in its use. At level 4, the Shewhart cycle is used to continually plan, implement, and track process improvements.

At the optimizing level (level 5) the organization has the means to identify its weakest process elements and strengthen them, data are available to justify applying technology to critical tasks, and evidence is available on process effectiveness. At this point, data gathering has at least been partially automated and management has redirected its focus from product repair to process analysis and improvement. The key additional activity at the optimizing level is rigorous defect cause analysis and defect prevention.

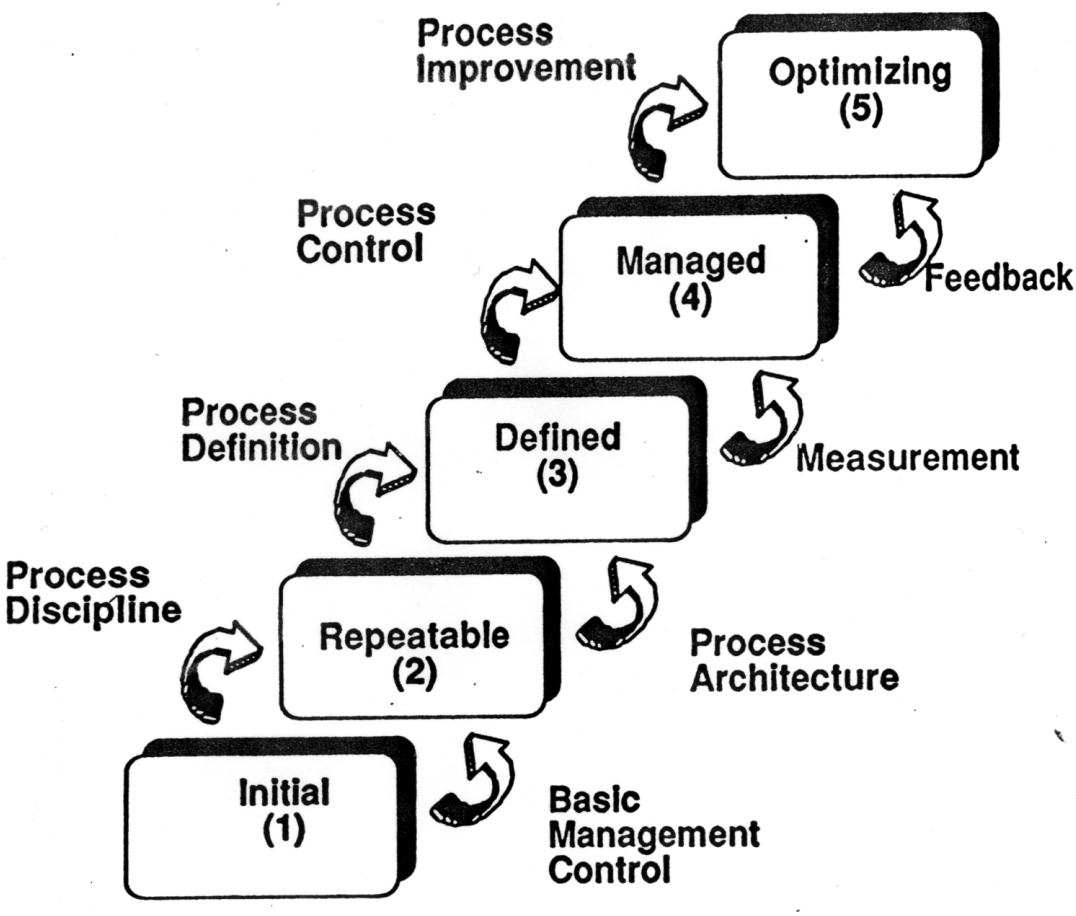

Figure 4-1: The Five Levels of Software Process Maturity

These maturity levels have been selected because:

- They reasonably represent the historical phases of evolutionary improvement experienced by software organizations.
- They represent a reasonable sequence of achievable improvement steps.
- They suggest interim improvement goals and progress measures.
- They make obvious a set of immediate improvement priorities, once an organization's status in this framework is known.

While there are many aspects to the advancement from one maturity level to another, the basic objective is to achieve a controlled and measured process as the foundation for continuous improvement. Some of the characteristics and key challenges of each of these levels are shown in Figure 4-2. A more detailed discussion is included in [Humphrey 89].

Level	Characteristic	Key Problem Areas	Result
5 **Optimizing**	Improvement fed back into process	Automation	**Productivity & Quality**
4 **Managed**	(quantitative) Measured process	Changing technology Problem analysis Problem prevention	
3 **Defined**	(qualitative) Process defined and institutionalized	Process measurement Process analysis Quantitative quality plans	
2 **Repeatable**	(Intuitive) Process dependent on individuals	Training Technical practices • reviews, testing Process focus • standards, process groups	
1 **Initial**	(ad hoc /chaotic)	Project management Project planning Configuration management Software quality assurance	**Risk**

Figure 4-2: The Key Process Challenges

Because of their impatience for results, organizations occasionally attempt to reach level 5 without progressing through levels 2, 3, or 4. This is counterproductive, however, because each level forms a necessary platform for the next. Consistent and sustained improvement also requires balanced attention to all key process areas. Inattention to one key area can largely negate advantages gained from work on the others. For example, unless effective means are established for developing realistic estimates at level 2, the organization is still exposed to serious overcommitments. This is true even when important improvements have been made in other areas. Thus, a crash effort to achieve some arbitrary maturity goal is likely to be unrealistic. This can cause discouragement and management frustration and lead to cancellation of the entire improvement effort. If the emphasis is on consistently making small improvements, their benefits will gradually accumulate to impressive overall capability gains.

Similarly, at level 3, the stability achieved through the repeatable level (level 2), permits the process to be examined and defined. This is essential because a defined engineering process cannot overcome the instability created by the absence of the sound management practices established at level 2 [Humphrey 89]. With a defined process, there is a common basis for measurements. The process phases are now more than mere numbers; they have recognized prerequisites, activities, and products. This defined foundation permits the data gathered at level 4 to have well-understood meaning. Similarly, level 4 provides the data with which to judge proposed process improvements and their subsequent effects. This is a necessary foundation for continuous process improvement at level 5.

5 Uses of the Maturity Model

The major uses of the maturity model are in process improvement and evaluation:

- In assessments, organizations use the maturity model to study their own operations and to identify the highest priority areas for improvement.
- In evaluations, acquisition agencies use the maturity model to identify qualified bidders and to monitor existing contracts.

The assessment and evaluation methods are based upon the maturity model and use the SEI questionnaire. It provides a structured basis for the investigation and permits the rapid and reasonably consistent development of findings that identify the organization's key strengths and weaknesses. The significant difference between assessments and evaluations comes from the way the results are used. For an assessment, the results form the basis for an action plan for organizational self-improvement. For an evaluation, they guide the development of a risk profile. In source selection, this risk profile augments the traditional criteria used to select the most responsive and capable vendors. In contract monitoring, the risk profile may also be used to motivate the contractor's process improvement efforts.

5.1 Software Process Assessment

An assessment is a diagnostic tool to aid organizational improvement. Its objectives are to provide a clear and factual understanding of the organization's state of software practice, to identify key areas for improvement, and to initiate actions to make these improvements. The assessment starts with the senior manager's commitment to support software process improvement. Since most executives are well aware of the need to improve the performance and productivity of their software development operations, such support is often readily available.

The next step is to select an assessment coordinator who works with local management and the assessing organization to make the necessary arrangements. The assessment team is typically composed of senior software development professionals. Six to eight professionals are generally selected from the organization being assessed together with one or two coaches who have been trained in the SEI assessment method. While the on-site assessment period takes only one week, the combined preparation and follow-on assessment activities generally take at least four to six months. The resulting improvement program should then continue indefinitely.

Software Process Assessments are conducted in accordance with a signed assessment agreement between the SEI-licensed assessment vendor and the organization being assessed. This agreement provides for senior management involvement, organizational representation on the assessment team, confidentiality of results, and follow-up actions. As described in Section 5.4, SEI has licensed a number of vendors to conduct assessments.

Software Process Assessments are typically conducted in six phases. These phases are:

1. Selection Phase: During the first phase, an organization is identified as a candidate for assessment and the assessing organization conducts an executive level briefing.
2. Commitment Phase: In the second phase, the organization commits to the full assessment process. An assessment agreement is signed by a senior executive of the organization to be assessed and the assessment vendor. This commitment includes the personal participation of the senior site manager, site representation on the assessment team, and agreement to take action on the assessment recommendations.
3. Preparation Phase: The third phase is devoted to preparing for the on-site assessment. An assessment team receives training and the on-site period of the assessment process is fully planned. This includes identifying all the assessment participants and briefing them on the process, including times, duration, and purpose of their participation. The maturity questionnaire is also filled out at this time.
4. Assessment Phase: In the fourth phase, the on-site assessment is conducted. The general structure and content of this phase is shown in Figure 5-1. On the first day, senior management and assessment participants are briefed as a group about the objectives and activities of the assessment. The team then holds discussions with the leader of each selected project to clarify information provided from the maturity questionnaire. On the second day, the team holds discussions with the functional area representatives (FARs). These are selected software practitioners who provide the team with insight into the actual conduct of the software process. At the end of the second day, the team generates a preliminary set of findings. Over the course of the third day, the assessment team seeks feedback from the project representatives to help ensure that they properly understand the issues. A final findings briefing is then produced by the team. On the fourth day, this briefing is further refined through dry run presentations for the team, the FARs, and the project leaders. The findings are revised and presented to the assessment participants and senior site management on the fifth day. The assessment ends with an assessment team meeting to formulate preliminary recommendations.
5. Report Phase: The fifth phase is for the preparation and presentation of the assessment report. This includes the findings and recommendations for actions to address these findings. The entire assessment team participates in preparing this report and presenting it to senior management and the assessment participants. A written assessment report provides an essential record of the assessment findings and recommendations. Experience has shown that this record has lasting value for the assessed organization.
6. Assessment Follow-Up Phase: In the final phase, a team from the assessed organization formulates an action plan. This should include members from the assessment team. There may be some support and guidance from the assessment vendor during this phase. Action plan preparation typically takes from three to nine months and requires several person-years' professional effort. After approximately 18 months, the organization should do a reassessment to assess progress and to sustain the software process improvement cycle.

To conduct assessments successfully, the assessment team must recognize that their objective is to learn from the organization. Consequently, 50 or more people are typically interviewed to learn what is done, what works, where there are problems, and what ideas the people have for process improvement. Most of the people interviewed are non-management software professionals.

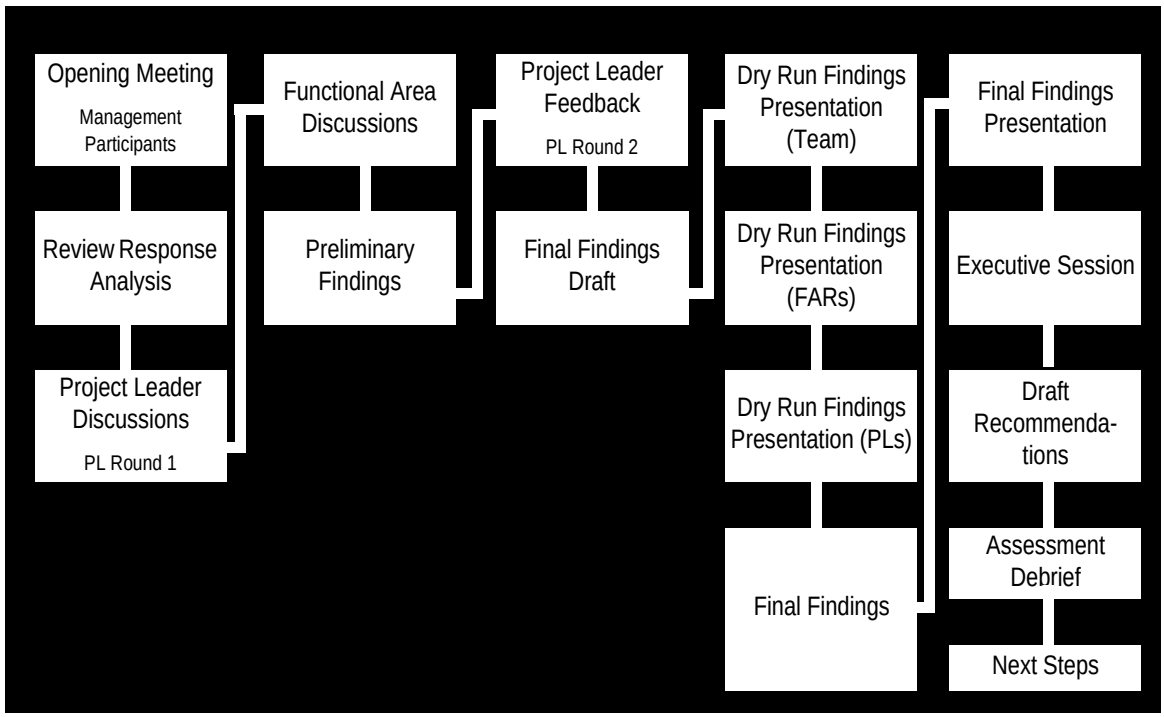

Figure 5-1: Assessment Process Flow

5.2 Software Capability Evaluation

Software capability evaluations (SCE) are typically conducted as part of the Department of Defense or other government or commercial software acquisition process. Many U.S. government groups have used SCE and several commercial organizations have also found them useful in evaluating their software contractors and subcontractors. A software capability evaluation is applied to the site where the proposed software work is to be performed. While an assessment is a confidential review of an organization's software capability largely by its own staff, an evaluation is more like an audit by an outside organization.

The software capability evaluation method helps acquisition agencies understand the software management and engineering processes used by a bidder. To be fully effective, the SCE evaluation approach should be included in the Source Selection Plan and the key provisions described in the request for proposal. After proposal submission and SCE evaluation team training, site visits are planned with each bidder. The acquisition agency then selects several representative projects from a set of submitted alternatives, and the project managers from the selected projects are asked to fill out a maturity questionnaire.

The on-site evaluation team visits each bidder and uses the maturity questionnaire to guide selection of the representative practices for detailed examination. Information is generally gained through interviews and documentation reviews. By investigating the process used on

the bidder's current projects, the team can identify specific records to review and quickly identify potential risk areas. The potential risk categories considered in these evaluations are of three types:

1. The likelihood that the proposed process will meet the acquisition needs.
2. The likelihood that the vendor will actually install the proposed process.
3. The likelihood that the vendor will effectively implement the proposed process.

Because of the judgmental nature of such risk evaluations, it is essential that each evaluation use consistent criteria and a consistent method. The SCE method provides this consistency.

Integrating the evaluation method into the acquisition process involves four steps:

1. Identifying the maturity of the contractor's current software process.
2. Assessing program risks and how the contractor's improvement plans alleviate these risks.
3. Making continuous process improvement a part of the contractual acquisition relationship .
4. Ongoing monitoring of software process performance.

Evaluations take time and resources. To be fully effective, the evaluation team must be composed of qualified and experienced software professionals who understand both the acquisition and the software processes. They typically need at least two weeks to prepare for and perform each site visit. Each bidder must support the site visit with the availability of qualified managers and professional staff members. Both the government and the bidders thus expend considerable resources in preparing for and performing evaluations for a single procurement. Software capability evaluations should thus be limited to large-scale and/or critical software systems, and they should only be performed after determining which bidders are in the competitive range.

A principle reason for this SCE evaluation approach is to assist during the source selection phase of a project. Since no development project has yet been established to do the work, other representative projects must be examined. The review thus examines the development practices used on several current projects because these practices are likely representative of those to be used on the new project. The SCE evaluation thus provides a basis for judging the nature of the development process that will be used on a future development.

5.3 Contract Monitoring

The SCE method can also be used to monitor existing software contracts. Here, the organization evaluates the maturity of the development work being done on the current contract. While the process is similar to that used in source selection, it is somewhat simpler. During the site visit, for example, the evaluation team typically only examines the project under contract. To facilitate development of a cooperative problem-solving attitude, some acquisition agencies

have found it helpful to use combined acquisition- contractor teams and to follow a process that combines features of assessments and evaluations. As more experience is gained, it is expected that contract monitoring techniques will evolve and improve as well.

5.4 Other Assessment Methods

Self-assessments are another form of SEI assessment, with the primary distinction being assessment team composition. Self-assessment teams are composed of software professionals from the organization being assessed. It is essential, however, to have one or two software professionals on the team who have been trained in the SEI method. The context, objective, and degree of validation are the same as for other SEI assessments.

Vendor-assisted assessments are SEI assessments that are conducted under the guidance of commercial vendors who have been trained and licensed to do so by the SEI. The assessment team is trained by the vendor and consists of software professionals from the organization being assessed plus at least one vendor professional who has been licensed by the SEI. By licensing commercial vendors, SEI has made software process assessments available to the general software community.

During the early development of the assessment method, SEI conducted a number of assessment tutorials. Here, professionals from various organizations learned about process management concepts, assessment techniques, and the SEI assessment methodology. They also supplied demographic data on themselves and their organizations as well as on a specific project. They did this by completing several questionnaires. This format was designed to inform people about the SEI assessment methodology to get feedback on the method, and to get some early data on the state of the software practice.

5.5 Assessment and Evaluation Considerations

The basic purposes of the Software Process Maturity Model, assessment method, and capability evaluation method are to foster the improvement of U.S. industrial software capability. The assessment method assists industry in its self-improvement efforts and the capability evaluation method provides acquisition groups with an objective and repeatable basis for determining the software capabilities of their vendors. This in turn helps to motivate software organizations to improve their capabilities. Generally, vendors cannot continue to invest in efforts that are not valued by their customers. Unless a software vendor's capability is recognized and valued by its customers, there can thus be little continuing motivation for process improvement. The SCE approach facilitates sustained process improvement by establishing process quality criteria, making these criteria public, and supporting software acquisition groups in their application.

One common concern with any evaluation system concerns the possibility that software vendors could customize their responses to achieve an artificially high result. This concern is a consequence of the common misconception that questionnaire scores alone are used in SCE

evaluations. Experience with acquisitions demonstrates that the SCE method makes this strategy impractical. When properly trained evaluators look for evidence of sound process practices, they have no difficulty in identifying organizations with poor software capability. Well-run software projects leave a clear documented trail that less competent organizations are incapable of emulating. For example, an organization that manages software changes has extensive change review procedures, approval documents, and control board meeting minutes. People who have not done such work are incapable of pretending that they do. In the few cases where such pretense has been attempted, it was quickly detected and caused the evaluators to doubt everything else they had been told. To date, the record indicates that the most effective contractor strategy is to actually implement a software process improvement program.

6 State of Software Practice

One of the most effective quality improvement methods is called benchmarking. An organization identifies one or more leading organizations in an area and then consciously strives to match or surpass them. Benchmarking has several advantages. First, it clearly demonstrates that the methods are not theoretical and that they are actually used by leading organizations. Second, when the benchmarked organization is willing to cooperate, it can be very helpful in providing guidance and suggestions. Third, a benchmarking strategy provides a real and tangible goal and the excitement of a competition. SEI has established state-of-the-practice data and has urged leading software organizations to identify themselves to help introduce the benchmarking method to the software development community.

While no statistically constructed survey of software engineering practice is available, SEI has published data drawn from the assessments SEI and its licensed vendors conducted from 1987 through 1991 [Kitson 92]. This includes data on 293 projects at 59 software locations together with interviews of several thousand software managers and practitioners. The bulk of these software projects were in industrial organizations working under contract to the U.S. Department of Defense (DoD). A smaller set of projects was drawn from U.S. commercial software organizations and U.S. government software groups. While there is insufficient data to draw definitive conclusions on the relative maturity of these three groups, SEI has generally found that industrial DoD software contractors have somewhat stronger software process maturity than either the commercial software industry or the government software groups.

Figure 6-2 shows the data obtained as of January 1992 on the maturity distribution of the U.S. software sites assessed. These results indicate that the majority of the respondents reported projects at the initial level of maturity with a few at levels 2 and 3. No sites were found at either the managed level (level 4) or the optimizing level (level 5) of software process maturity.

While these results generally indicate the state of the software engineering practice in the U.S., there are some important considerations relating to SEI's data gathering and analytical approach.

First, these samples were not statistically selected. Most respondents came from organizations affiliated with the SEI. Second, the respondents also varied in type and degree of involvement with the projects on which they reported.

These results are also a mix of SEI-assisted and self-assessments. While the assessment methods were the same in all cases, the selection process was not. SEI focused primarily on organizations whose software work was of particular importance to the DoD or organizations that were judged to be doing advanced work. Figure 6-2 shows these two distributions. Here, the SEI-assisted assessments covered 63 projects at 13 sites and the self-assessments covered 233 projects at 46 sites.

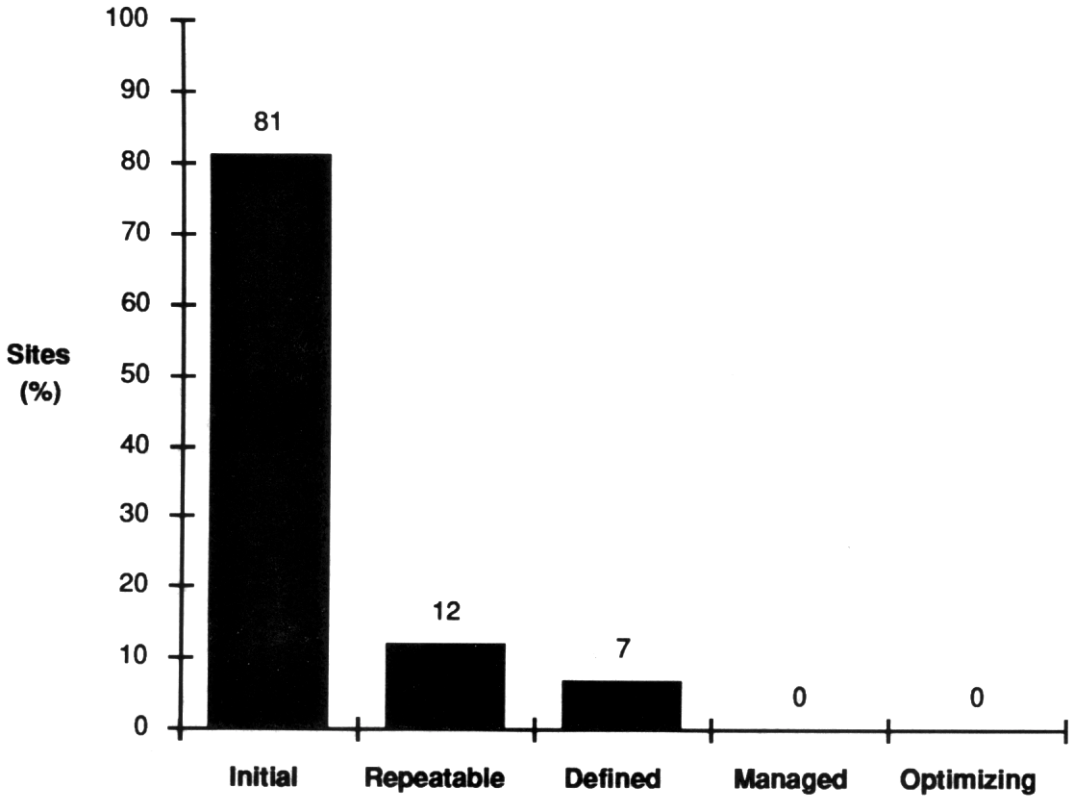

Figure 6-1: U.S. Assessment Maturity Level Distribution

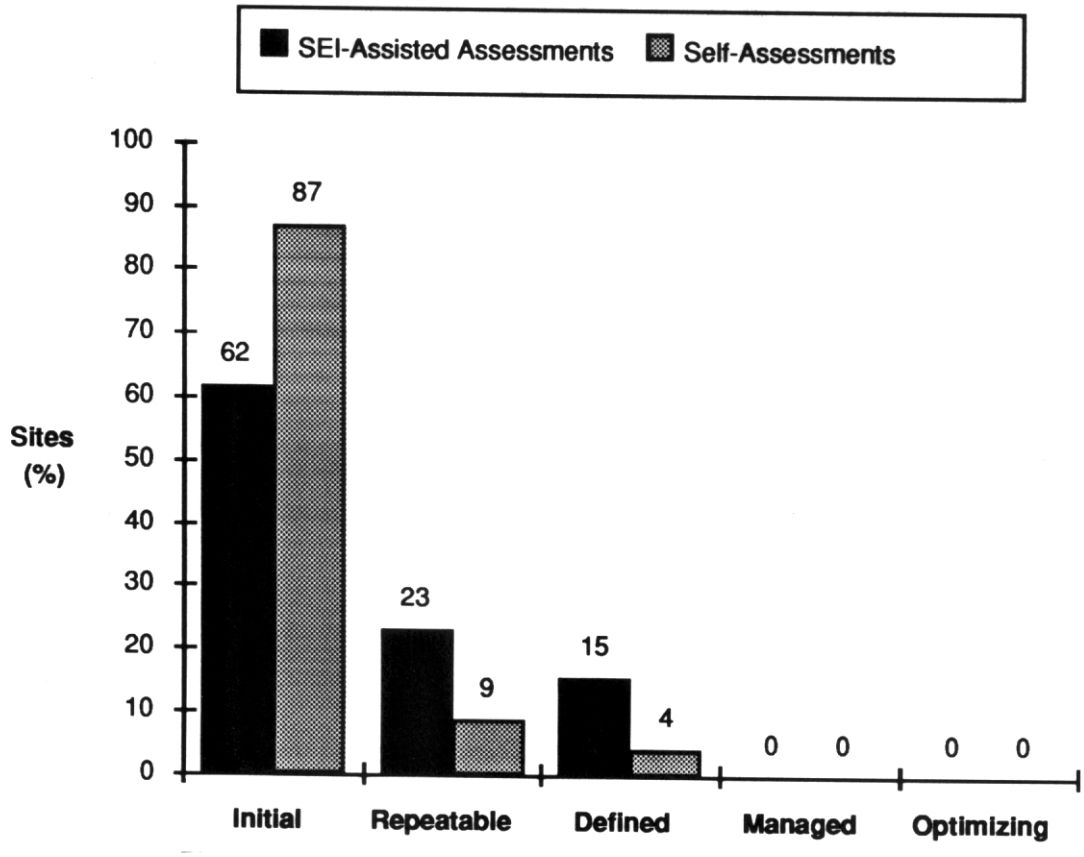

Figure 6-2: U.S. Assessment Maturity Level Distribution by Assessment Type[1]

1. Thirteen sites encompassing 63 projects conducted SEI-assisted assessments; 46 sites encompassing 233 projects conducted self-assessments.

7 Process Maturity and CASE

CASE systems are intended to help automate the software process. By automating the routine tasks, labor is potentially saved and sources of human error are eliminated. It has been found that a more effective way to improve software productivity is by eliminating mistakes rather than by performing tasks more efficiently [Humphrey 89B]. This requires an orderly focus on process definition and improvement.

For a complex process to be understood, it must be relatively stable. W. Edwards Deming, who inspired the post war Japanese industrial miracle, refers to this as statistical control [Deming 82]. When a process is under statistical control, repeating the work in roughly the same way will produce roughly the same result. To get consistently better results, statistical methods can be used to improve the process. The first essential step is to make the software process reasonably stable. After this has been achieved, automation can be considered to improve its efficiency.

Consistent and sustained software process improvement must ultimately utilize improved technology. Improved tools and methods have been helpful throughout the history of software; but once the software process reaches maturity level 5 (optimizing), organizations will be in a far better position to understand where and how technology can help.

In advancing from the level 1 (chaotic) process, major improvements are made by simply turning a group of programmers into a coordinated team of professionals. The challenge faced by immature software organizations is to learn how to plan and coordinate the creative work of their professionals so they support rather than interfere with each other. The first priority is for a level 1 organization to establish the project management practices required to reach level 2. It must get its software process under control before it attempts to install a sophisticated CASE system. To be fully effective, CASE systems must be based on and support a common set of policies, procedures, and methods. If these systems are used with an immature process, automation will generally result in more rapid execution of chaotic activities. This may or may not result in improved organizational performance. This conclusion does not apply to individual task-oriented tools such as compilers and editors, where the tasks are reasonably well understood.

8 Improvement Experience

Of the many organizations that have worked with SEI on improving their software capability, some have described the benefits they obtained. At Hughes Aircraft, for example, a process improvement investment of $400,000 produced an annual savings of $2,000,000. This was the culmination of several years work by some experienced and able professionals, with strong management support [Humphrey 91]. Similarly, Raytheon found that the cost benefits of software process improvement work reduced their testing and repair efforts by $19.1 million in about four years [Dion 92]. While the detailed results of their improvement efforts have not been published, it is understood that their costs were of the same general order of magnitude as at Hughes Aircraft.

A similar and longer term improvement program was undertaken at IBM Houston by the group that develops and supports the on-board software for the NASA space shuttle. As a result of their continuous process improvement work, and with the aid of IBM's earlier assessment efforts, they achieved the results shown in Table 8-1 [Kolkhorst 88].

	1982	1985
Early error detection (%)	48	80
Reconfiguration time (weeks)	11	5
Reconfiguration effort (person-years)	10.5	4.5
Product error rate (errors per 1000 lines of code)	2.0	0.11

Figure 8-1: On-Board Shuttle Software

9 Future Directions

Future developments with the SEI maturity model must be intimately related to the future developments in software technology and software engineering methods. As the SEI works to evolve and improve the software process maturity model, it will likely focus on organizations in the U.S. but both the Europeans and Japanese are showing considerable interest. With proper direction, this work should evolve into an internationally recognized framework for process improvement. This would likely require broader and more formal review and approval mechanisms to involve all interested parties.

Historically, almost all new developments in the software field have resulted in multiple competing efforts. This has been highly destructive in the cases of languages and operating systems, and appears likely to be a serious problem with support environments as well. While each competing idea may have some nuance to recommend it, the large number of choices and the resulting confusion has severely limited the acceptance of many products and has generally had painful economic consequences as well. The SEI has attempted to minimize such a risk by inviting broad participation in the development and review of the CMM and questionnaire improvement efforts. Several hundred software professionals from many branches of industry and government have participated in this work, and the SEI change control system now has several thousand recorded comments and suggestions. It is thus hoped that anyone with improvement suggestions will participate in this work rather than start their own. Such separate efforts would dilute the motivational and communication value of the current software process maturity model and make it less effective for process improvement. The need is to evolve this instrument and framework to meet the needs of all interested parties. This would assure maximum utility of the maturity model and retain its credibility in the eyes of managers and acquisition groups.

Several divisions of the U.S. Department of Defense acquisition community are beginning to adopt the SCE method, with the CMM as its basis, as a routine practice for evaluating its major software vendors. This work has been started with the support of the SEI and, as of 1992, more than 500 government personnel had been trained in SCE and more than 45 such acquisitions have been conducted. Roughly 88% of the personnel trained and 75% of the acquisitions conducted were during 1991-1992. This is indicative of the increasing user perception and acceptance of the SCE method as a viable acquisition tool. The anecdotal evidence from these acquisition experiences have been consistently positive for the users, and increased usage is likely.

The greatest risk with SCE is its accelerated use. In their eagerness to address the current severe problems with software acquisition, management could push its application faster than capability evaluation teams could be trained and qualified. If this results in an overzealous audit mentality, industry's improvement motivation will likely be damaged. The result could easily be another expensive and counterproductive step in an already cumbersome and often ineffective acquisition process.

More generally, several research efforts in the U.S. and Europe are addressing various aspects of process modeling and process definition. This work has recognized that processes are much like programs and that their development requires many of the same disciplines and technologies [Osterweil 87, Kellner 88]. This body of work is building improved understanding of the methods and techniques for defining and using processes. This in turn will likely facilitate work on improved automation support of the software process.

10 Conclusions

Software is pivotal to U.S. industrial competitiveness, and improved performance of software groups is required for the U.S. to maintain its current strong position. As has happened in many industries, an early U.S. technological lead can easily be lost through lack of timely attention to quality improvement. With proper management support, dedicated software professionals can make consistent and substantial improvements in the performance of their organizations. The SEI Capability Maturity Model has been helpful to managers and professionals in motivating and guiding this work.

References

[Crosby 79] Crosby, P. B., ***Quality is Free***, McGraw-Hill, New York, 1979.

[Deming 82] Deming, W. E., ***Out of the Crisis***, MIT Center for Advanced Engineering Study, Cambridge, MA, 1982.

[Dion 92] Dion, R.A., "Elements of a Process-Improvement Program," ***IEEE Software***, July 1992, pp. 83-85.

[Humphrey 87] Humphrey, W. S. and Sweet, W. L., ***A Method for Assessing the Software Engineering Capability of Contractors***, Software Engineering Institute Technical Report CMU/SEI-87-TR-23, ADA187230 Carnegie Mellon University, Pittsburgh, PA, 1987.

[Humphrey 89] Humphrey, W. S., ***Managing the Software Process***, Addison-Wesley, Reading, MA, 1989(B).

[Humphrey 91] Humphrey, W. S., Snyder, T. R., and Willis, R. R., "Software Process Improvement at Hughes Aircraft," ***IEEE Software***, July 1991.

[Kellner 88] Kellner, M. I., "Representation Formalism for Software Process Modeling," ***Proceedings of the 4th International Software Process Workshop***, ACM Press, 1988, pp. 93-96.

[Kitson 92] Kitson, D. H. and Masters, S., ***An Analysis of SEI Software Process Assessment Results 1987-1991***, Software Engineering Institute Technical Report CMU/SEI-92-TR-24, ADA253351, Carnegie Mellon University, Pittsburgh, PA, 1992.

[Kolkhorst 88] Kolkhorst, B. G. and Macina, A. J., "Developing Error-Free Software," Proceedings of Computer Assurance Congress ‘88, ***IEEE Washington Section on System Safety,*** June 27-July 1, 1988, pp. 99-107.

[Osterweil 87] Osterweil, L., "Software Processes are Software Too," ***Proceedings of the 9th International Conference on Software Engineering***, Monterey, CA, IEEE Computer Society Press, 1987, pp. 2-13.

[Paulk 91] Paulk, M. C., Curtis, B., and Chrissis, M. B., ***Capability Maturity Model for Software***, Software Engineering Institute Technical Report CMU/SEI-91-TR-24, Carnegie Mellon University, Pittsburgh, PA, August 1991.

[Radice 85] Radice, R. A., Harding, J. T., Munnis, P. E., and Phillips, R. W., "A Programming Process Study," ***IBM Systems Journal***, vol. 24, no. 2, 1985.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release Distribution Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CMU/SEI-92-TR-7	5. MONITORING ORGANIZATION REPORT NUMBER(S) ESC-TR-92-007
6a. NAME OF PERFORMING ORGANIZATION Software Engineering Institute 6b. OFFICE SYMBOL (if applicable) SEI	7a. NAME OF MONITORING ORGANIZATION SEI Joint Program Office
6c. ADDRESS (city, state, and zip code) Carnegie Mellon University Pittsburgh PA 15213	7b. ADDRESS (city, state, and zip code) HQ ESC/ENS 5 Eglin Street Hanscom AFB, MA 01731-2116
8a. NAME OFFUNDING/SPONSORING ORGANIZATION SEI Joint Program Office 8b. OFFICE SYMBOL (if applicable) ESC/ENS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F1962890C0003
8c. ADDRESS (city, state, and zip code)) Carnegie Mellon University Pittsburgh PA 15213	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO: 63756E PROJECT NO.: N/A TASK NO: N/A WORK UNIT NO.: N/A
11. TITLE (Include Security Classification) Introduction to Software Process Improvement (Revised: June 1993)	
12. PERSONAL AUTHOR(S) Watts S. Humphrey	
13a. TYPE OF REPORT Final 13b. TIME COVERED FROM TO	14. DATE OF REPORT (year, month, day) June 1992 (Revised June 1993) 15. PAGE COUNT 42
16. SUPPLEMENTARY NOTATION THIS IS A REVISION TO THE FIRST VERSION, WHICH WAS ORIGINALLY ISSUED IN JUNE 1992.	

17. COSATI CODES			18. SUBJECT TERMS (continue on reverse of necessary and identify by block number)
FIELD	GROUP	SUB. GR.	culture, goals, improvement, management, maturity model, measurement, process, quality, resources, software

19. ABSTRACT (continue on reverse if necessary and identify by block number)

While software now pervades most facets of modern life, its historical problems have not been solved. This report explains why some of these problems have been so difficult for organizations to address and the actions required to address them. It describes the Software Engineering Institute's (SEI) software process maturity model, how this model can be used to guide software organizations in process improvement, and the various assessment and evaluation methods that use this model. The report concludes with a discussion of improvement experience and some comments on future directions for this work.

(please turn over)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ■ SAME AS RPT □ DTIC USERS ■	21. ABSTRACT SECURITY CLASSIFICATION Unclassified, Unlimited Distribution
22a. NAME OF RESPONSIBLE INDIVIDUAL Thomas R. Miller, Lt Col, USAF	22b. TELEPHONE NUMBER (include area code) (412) 268-7631 22c. OFFICE SYMBOL ESC/ENS (SEI)

ABSTRACT — continued from page one, block 19