## SOFTWARE PROCESS MANAGEMENT

As pointed out by many experts in the software industry, measurement and management of the measures have been "the basis of all science and engineering progress except for software" (1). While this has been the case historically, we believe that software engineering has evolved into a well-disciplined process. There are many indicators or evidence that suggest this evolution. Many companies in the software industry, for example, have focused internal efforts on identifying best practices that lead to the production of higher quality software products with predictive results. Best practices may include specific project management techniques, inspection and reviews, root cause analysis techniques, and continuous process improvement. These practices are defined, documented and transitioned across organizations developing software products to improve software quality, decrease interval and reduce cost. Examples of companies that have focused efforts on best practices include Boeing, Citicorp, IBM, Hewlett-Packard Lucent Technologies, and Motorola to name a few. Microsoft has also been using many best practices (2). Another indicator of this evolution is the adoption of the Deming Plan/Do/Check/Act cycle by many software companies to help drive their process improvement efforts. Other indicators include the increased use of benchmarking, the focus on process definition, and the application of ISO 9000 to software organizations.

This movement has been driven largely by ever increasing customer expectations and demands for higher quality, shorter delivery cycles (interval) and lower costs. A major breakthrough in recent years has been the fact that software projects and organizations are beginning to evaluate their own performance in order to determine how to meet these increasingly challenging customer expectations. As a result, companies in the software industry now routinely evaluate their capabilities to develop software products. Questions like the following are often asked to understand this capability (or lack thereof). Do we have the ability to define and control product requirements? Can we adequately plan and manage a software project? Is there a way to know quantitatively if our product is ready for deployment? Software companies now often attempt to answer these types of questions through the use of various assessment techniques, which are used internally or provided by external consultants.

Watts Humphrey developed the concept of process maturity levels which is at the heart of the SEI Software Capability Maturity Model (CMM). The Software CMM has become largely an industry standard in terms of evaluating organizational capability. CMM-based assessments are used by many companies in the software industry to understand their current processes, and to determine how to make improvements to reduce software intervals, improve customer satisfaction and reduce costs. Quantitative data is now being shared across companies in the software industry that show a direct relationship between the organization capability or maturity and the ability to produce high quality software that achieves strong levels of customer satisfaction in an acceptable schedule. What is unique about the CMM is that it identifies the hierarchy of processes essential to effective management of software development. Organizations that are striving to reach higher levels of quality and productivity in software build their capabilities incrementally following the CMM road map.

In the rest of this paper, we will examine software development as a process following the Deming Cycle.

### PROCESS MANAGEMENT

A process is a set of defined steps organized to achieve some purpose. It usually has some inputs and associated outputs. For example, a design process may have customer requirements as input, and the output may be a design document that has been approved by the software project team. The high level steps in the process might look like the following:

- Review requirements
- Identify the set of requirements that will be satisfied by this design
- Identify the necessary software modules that will satisfy the design
- Document the proposed design with a structure chart and interfaces
- Review the proposed design document with team members and revise as needed
- Obtain design document approval from the software team

Process management means incorporating a specific and systematic program into an organization or project to:

- Establish product or service requirements, based on customer needs, that can be measured and verified.
- Define and structure a process for producing a product or service that aims to meet these requirements for a given cost (budget) and in a defined time.
- Measure, analyze, control and improve a product or service and the process by which it is realized, using the most effective tools and techniques available.

Figure 1 depicts the general model for a customer supplier relationship in process management. The structure of interrelated elements that combine to achieve these objectives is the *quality system*. The quality system helps create an environment in which capable people do their best work by providing measurements and controls that define what is expected and monitor what is received. Correctly used, these measurements and controls provide the basis for driving improvement. Managers assure their correct use by focusing the feedback from measurements and controls on the system, not the individuals. The tools and techniques of quality management are meant to fix the system, not fix the blame. The goal is improvement.

The quality system offers a systematic approach of planning and problem solving, of analyzing the deficiencies of existing systems and of acting to attain improvement. W. Edwards Deming approached this as a cycle of *planning*,

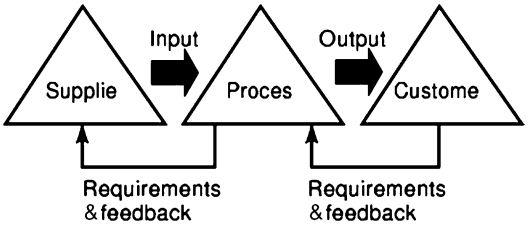


**Figure 1.** The Customer Supplier Model

*doing*, *checking*, and *acting* (3). This is very often referred to as the Plan-Do-Check-Act (PDCA) Model.

The application of the PDCA Model to a software process is conceptually straightforward: a **plan** defines how to produce the desired software product; the project team executes the plan to build the desired software product (**do**); as activities occur during the execution of the plan, they are monitored to ensure they are occurring as desired (**check**); and as the team finds problems during the execution of the plan, it takes action (**act**) to resolve the problems, and put in place preventive measures. The following subsections describe software process management in terms of the application of the PDCA Model to a software project.

### Plan

A key driver of the planning phase is to ensure there is an agreed upon understanding of the customer needs: what is driving the software organization to create a new or enhanced product, what problem(s) will the product solve for the customer, when is the product needed, and how will the product be used. This includes defining the requirements for the product, including cost, schedule and other factors.

Once there is a common understanding of the customer needs, a plan is necessary to answer all the important questions about how the product will be defined, developed and delivered to the customer. These questions usually revolve around addressing who, when, what, where and how. The plan will define the specific objectives for the project, as well as how the project will be launched, monitored and controlled through to completion.

In a typical software project, plans are generally defined with the following types of information:

- *Scope:* establishing the project goals and objectives.
- *Product description:* defining the major product functions and characteristics.
- *Resources:* identifying the team members who will work on the project, what customer resources the project is dependent upon, and what physical resources (e.g., computing hardware) or special resources (e.g., training) are necessary.
- *Cost:* what will the project cost? This includes the costs associated with the people working on the project, related hardware, training and physical environment, etc.
- *Schedule:* the project interval and milestones, including work item dependencies, as well as the critical path.
- *Organization:* who specifically will be working on the project, and what the individual roles and responsibilities are. What deliverables will be provided by whom? What external dependencies (if any) exist, both in terms of off-the-shelf as well as custom software that may be incorporated as part of the product? Are there hardware (or other) dependencies beyond the scope of the software team?
- *Methods, processes and procedures:* the process by which the product will be developed. How will requirements, design, coding and testing activities be performed? Are there certain methods that will be used for key life cycle phase activities, and/or tools that should support the product development?
- *Risk identification and management:* what risk factors exist for the project? These may include risks related to the software team (e.g., the risk that a new Java technology is being used for the first time, and team members have no experience with this) or beyond the team. External dependencies may include things like relying on a new hardware platform or component, subcontracting with a vendor for a custom software component or a critical portion of the software, or undertaking a product development effort significantly different from the team's previous experience base (taking compiler developers, for example, and having them work on the next generation of network switches; or assigning a team that has done development locally to a globally distributed development project.) For each risk, the software team needs to estimate the probability that the undesired event will occur, as well as identify the contingency plan(s) to mitigate the risk. This should include defining the specific actions that will be taken if the risk should materialize.
- *Supporting functions:* how will the software project be managed from concept to delivery? How will configuration management be performed? What is the approach that will be taken in terms of education and training of team members, both in terms of training needed across the team as well as among individual members.

### Do

The "*do*" portion of the PDCA Model applied to the software process primarily involves the execution or implementation of the project plan. That is, gather the resources necessary to do the work that was planned, and have the team members begin to do the work according to the schedule that was defined in the plan.

The project management function is a very key one during this portion of PDCA Model. This function is usually directing and managing project resources to undertake specific activities according to the plan, and will be dealing with problems that invariably occur when executing a project plan (e.g., computing resources are not available that were anticipated, or critical project team members will be starting work on the project two months later than anticipated). The project management function will control changes to the plan as the need arises, and this often occurs

as the progress is tracked and measured against the schedule. This control is typically characterized as elements of the check and act portions of the PDCA cycle.

### Check

As a project implementation proceeds, there is a need to "*check*" or validate the correctness and effectiveness of the implementation. There are basically three layers of "checks" in a software development process as depicted in Figure 2:

- Checks applied to the product (work-product or end-product), or *Product Assurance*. These include reviews and inspections, and test and verification.
- Checks applied to the project, or *Project Review*. These include project status reviews, project audit or project management audit, and retrospectives.
- Checks applied to the development process or processes in a software organization, or *Process Assessment*.

Each of these is briefly discussed.

**Product Assurance.** During the development process, the work-products or the output from various stages of the process is evaluated for compliance to applicable standards and requirements. *Inspection* is one of the key mechanisms used by software projects to validate and improve the work-in-progress as the project moves from concept to deployment. Formal inspection is a structured process, conducted by a team of subject matter experts. The work product is carefully examined by the inspection team to identify any problems, deficiencies and non-conformities and to recommend corrective action. Effective software development organizations conduct rigorous inspections of all project documentation, particularly requirements, architecture, design, and test plans, as well as the code (4–8). In software development, inspections are normally done by peers (also known as peer reviews) and are overseen or approved by the quality assurance organization. Although the primary reason for inspection is to ensure the quality of the product at hand, the data generated from inspections are a rich source of information for improving the processes to prevent such errors in the future. Leading edge software companies apply statistical techniques using inspection data to predict latent faults in the product and to predict product performance (9, 10). Product assurance for the end-product, the software, is done through test and verification. Some of the testing is done by the developers themselves, validating the individual modules, functions and subsystems. The integrated product goes through system verification and validation, often done by a separate and dedicated group. The product may also be further validated by the customer through customer acceptance testing. The quality assurance organization often has the primary responsibility to support and oversee the product assurance activities.

**Project Review.** In addition to product assurance, a number of checks are applied to the project to assure the project as a whole.

*Project Status Review* is a regular project activity conducted throughout the project life cycle to evaluate project's progress and identify issues that need to be resolved quickly to keep the project on track with respect to the project plan.

*Project Audit* or project management audit is a broader check on the project, often conducted by an independent team. A project audit is an evaluation of the project's plans, activities and resources for:

- Adequacy and effectiveness to achieve the project objectives.
- Adherence to the established guidelines, policies and practices.
- Identification of problem areas and recommended actions.

Project audits are conducted during the life cycle of the project to minimize project risks by addressing problems as early as possible.

*Retrospective* is also a project-level review. It is done at the completion of the project for lessons learned. In many companies project audits and retrospectives are routinely conducted on every project (11), and in some organizations these are conducted at the conclusion of major product life cycle phases.

**Process Assessment.** At yet a higher level, a software development process is evaluated for its effectiveness and ability to develop and deliver software products faster, cheaper and better. ISO/IEC 90003: 2004[1] (its earlier version ISO 9000-3) and TickIT[2] certification is a form of process assurance for software. Under ISO 9001: 2000 the processes are periodically assessed to ensure they are well-defined and documented and that projects do follow the defined processes. ISO 9000 provides for the basic process management to meet contractual requirements for adequacy of a quality system. To meet the challenges of a very competitive industry, software companies are striving to go beyond ISO 9000 to improve their software competency and process effectiveness. Among industry approaches, two quantitative models have emerged as effective process assessment tools to help software development organizations systematically evaluate their process capabilities against the benchmarks. These models are: Software Engineering Institute (SEI) Capability Maturity Model (CMM)® (12, 13) and Software Productivity Research (SPR) Model (1, 14).

**SEI-Capability Maturity Model.** The SEI CMM is a framework for application of process management and quality improvement to software development and maintenance. The CMM considers a natural progression for a software organization towards organizational excellence. The model was developed by the Software Engineering Institute; a Department of Defense funded organization associated with the Carnegie Mellon University. The model has five distinct levels as shown in Figure 3:

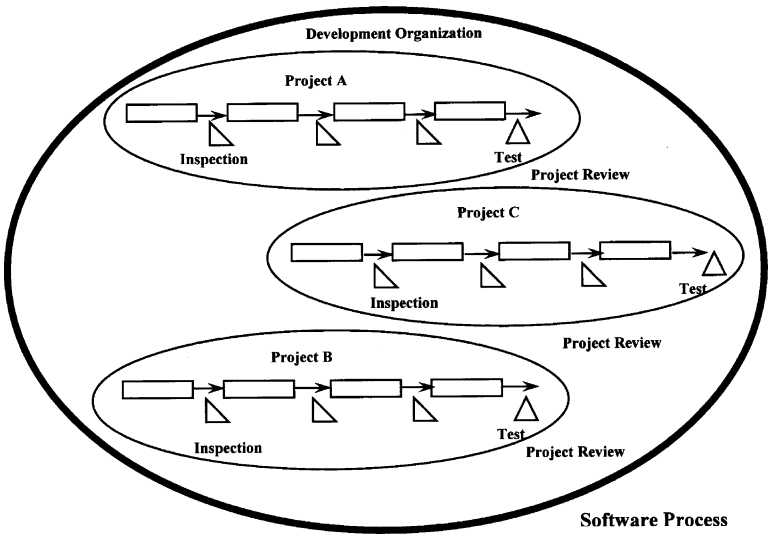


**Figure 2.** Hierarchy of Checks in Software Development

- Level 1 is characterized as *Initial*. The processes are *ad hoc*; there is very little formalism, and projects are completed by heroics.
- Level 2 is characterized as *Repeatable*. There exists a set of project management capabilities that allow consistent planning and tracking of projects.
- Level 3 is characterized as *Defined*. The organization has achieved a process focus; processes are well defined and documented.
- Level 4 is characterized as *Managed*. The organization is focusing on quality improvement; processes are consistently measured and managed.
- Level 5 is characterized as *Optimizing*. The organization is focusing on problem prevention, innovation, and management of process change.

An organization capability can be assessed against the CMM requirements to establish the current level and develop improvement plans for moving up on the maturity level. The assessment tool uses a questionnaire along with document review and interviews for the evaluation. The questionnaire samples the existence and effectiveness of the elements of the model. SEI has evolved the CMM assessment process. The Capability Maturity Model Integration (CMMI®) is a process improvement approach using CMM framework (15). It can be used to guide process improvement across a project, a division, or an entire organization. SEI provides a Standard CMMI Appraisal Method for Process Improvement (SCAMPISM) through licensed appraisers to support broad application of its best practices (16).

**Software Productivity Research (SPR).** The SPR model was developed by Capers Jones, a software industry expert. It is based on an extensive questionnaire covering various dimensions of software development process. The model scores process effectiveness in a number of functional areas so that the organization assessed will know if the effectiveness of their process is comparable to industry benchmarks. The scoring is built into the questions with five possible answers for most questions, where the choices reflect excellent (score of 1) to very poor practices (score of 5). The SPR assessment results can be summarized in a number of ways. One view is a Kiviat chart, which is a graphical depiction of the overall assessment results across eight functional categories:

- *Customer Focus*–Measures the level of customer involvement in the various stages of product development, such as requirements, customer documentation, and testing.
- *Project Management*–Measures the effectiveness of estimation and scheduling, alignment of managers and engineers on goals, methods, and other project management issues.
- *Project Team Variables*–Measures the experience and training of the project staff.
- *Tools*–Measures the effectiveness of the coding and non-coding tools used by the project.
- *Quality Focus*–Measures the extent and effectiveness of various quality assurance activities used by the project throughout the entire product development life cycle.
- *Methodologies*–Measures the effectiveness of methods used in development activities, such as requirements methods, coding standards and system test methods.
- *Physical Environment*–Measures the effectiveness of the development and target hardware, the office environment, and the adequacy of clerical support.
- *Metrics*–Measures the extent and effectiveness of product performance, defects, and project productivity data collected and used within the project.

The score for each category is the average of the score for the questions included in that category. An average score from 1.0 to 2.5 is considered leading edge with respect to the software industry, a score from 2.6 to 3.4 is considered at industry norm, and a score from 3.5 to 5.0 is consid-

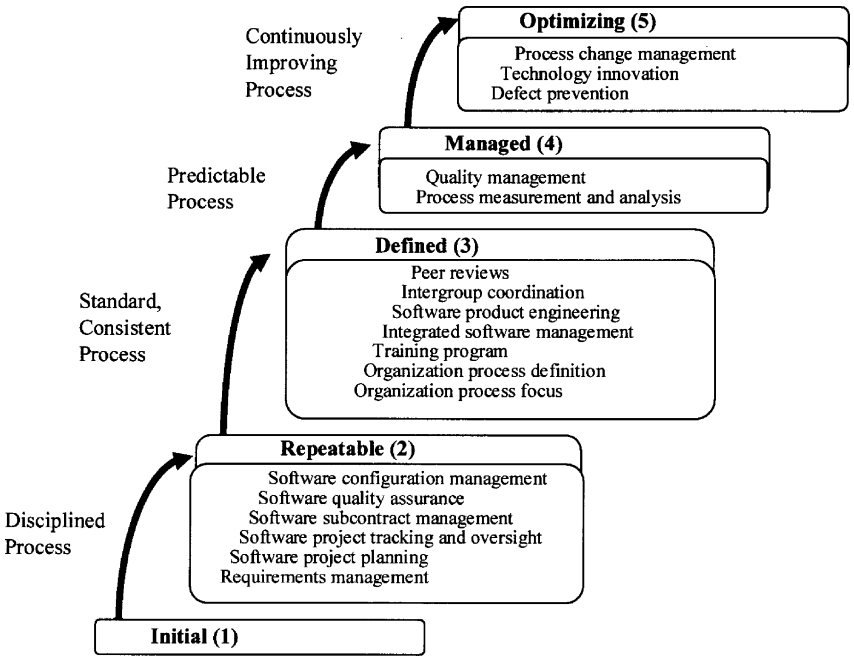


**Figure 3.** SEI Capability Maturity Model

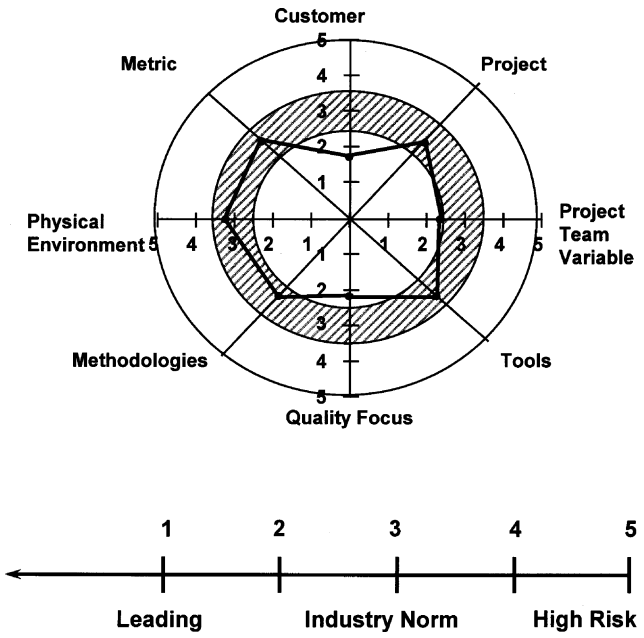


**Figure 4.** A Typical Kiviat Chart

ered high risk. Figure 4 shows a typical Kiviat chart for a hypothetical organization (or a project). In Figure 4, the organization can track its improvements over time as it goes through PDCA; the points on the Kiviat chart should move toward the center circle (bullseye). Another way to use the assessment is to synthesize the results by category into a pie chart as shown in Figure 5. The benchmark profile is a hypothetical example of creating a composite profile for the leading organizations in the industry as a benchmark.

### Act

An outcome of the *check* stage is identification of nonconformities and areas for improvement. The organization usually takes two kinds of actions.

**Corrective Action (Rework).** This is aimed at fixing the observed problem(s), generally through rework. For example, after an inspection of a requirements document, the systems engineer owning the document is responsible for correcting the faults or problems identified. Similarly, at validation and verification, the test group identifies failures in the system. The developer, or the maintenance group (if the project has a dedicated maintenance staff), will have to create fixes for the problems and input them through the system until the product achieves an acceptable quality level.

**Action Plan.** To prevent problems from recurring, one has to identify the root causes of the problems which often

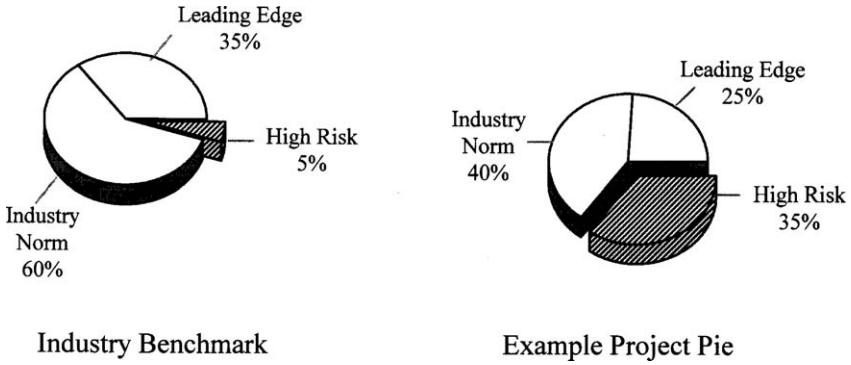


**Figure 5.** Comparison of SPR Results to Industry Benchmark

point to a process or capability deficiency and eliminate those root causes. A systematic approach to action planning and implementation includes the following:

- Prioritization of the improvement opportunities
- Identification of the root causes of the priority items
- Development of counter measures
- Evaluation of the counter measures for cost effectiveness
- Assignment of resources to implement selected counter measures
- Implement the counter measures
- Assess the impact
- Integrate the improvements into the current process to hold the gains.

Many organizations use the Six Sigma Methodology that incorporates the above steps into 5 stages: Define, Measure, Analyze, Improve, and Control (DMAIC) (17). This methodology was developed by GE® in applying Six Sigma quality improvement. Lockheed Martin is among the growing list of companies that regularly use Six Sigma to improve its software development process (18). Through rigorous and systematic improvement, a software organization can improve its ability to develop and deliver high quality software products within schedule and budget constraints. An SEI analysis of 35 organizations that have been following CMMI guidance shows a median improvement of 34% in cost, 50% in schedule and 61% in productivity (19).

## CONCLUSION

The software industry has been driven by ever increasing customer expectations and demands for higher quality, shorter delivery cycle, and lower cost. Software projects and organizations can systematically evaluate their own performance in order to determine how to meet these increasingly difficult customer expectations. The SEI Capability Maturity Model (CMM) has become largely an industry standard in terms of evaluating organizational capability. CMM-based assessments are used by many companies in the software industry to understand their current processes, and to determine how to make improvements to reduce software intervals, improve customer satisfaction and reduce costs. What is unique about the CMM is that it identifies the hierarchy of processes essential to effective management of software development. Organizations that are striving to reach higher levels of quality and productivity in software build their capabilities incrementally following the CMM road map. The application of PDCA provides a powerful mechanism to drive process improvement in an organization. Factors that affect its effectiveness include:

- Management Commitment – The organization should be committed to process improvement.
- Well-defined Process – Standardized assessment methodology and tool coupled with benchmarks for comparison, e.g. SEI CMM.
- Continuity – Repeating the assessment periodically to reinforce improvements and focus on current priorities.

## BIBLIOGRAPHY

1. C. Jones," Applied Software Measurement," McGraw-Hill, 1991.
2. M. Cusumano and R. Selby," Microsoft Secrets,"*The Free Press*, 1995.
3. AT&T," Quality By Design,"Issue 1, 1986.
4. M. E. Fagan," Advances in Software Inspections,"*IEEE Transactions of Software Engineering*, Vol.**SE-12**, No. 7,July 1986.
5. G. W. Russel," Experiences with Inspection in Ultralarge-Scale Development,"*IEEE Software*, January 1991.
6. T. Gilb and D. Graham," Software Inspection,"Addison-Wesley, 1993.
7. R. G. Ebenau and S. H. Strauss," Software Inspection Process," Mcgraw Hill, 1993.
8. R. A. Radic, " High Quality Low Cost Software Inspections," Paradoxicon Publishing, 2004.

[1] ISO/IEC 90003:2004 provides guidance for organizations in the application of ISO 9001:2000 to the acquisition, supply, development, operation and maintenance of computer software and related support services.
[2] TickIT procedures relate directly to the requirements set out in ISO 9001:2000. TickIT is supported by UK and Swedish software industries.
® CMM is a registered Trademark of Carnegie Mellon University.
® CMMI is a registered Trademark of Carnegie Mellon University.
® Registered Trade Mark of General Electric Company.

9. D. A. Christenson, S. T. Hung, A. J. Lamperez,“ Statistical Quality Control Applied to Code Inspections,” *IEEE Journal on Selected Areas in Communications*, Vol.**8**, No. 2,February 1990.
10. R. Radice,“ Statistical Process Control in Level 4 and 5 Organizations Worldwide,” Proceedings of the 12th Annual Software Technology Conference, 2000.
11. M. H. Fallah, J. P. Holtman, J. F. Maranzano, D. P. Smith, G. T. Tucker,“ Development Process Audits and Reviews,” *AT&T Technical Journal*, Vol.**70**, No. 2,March/April 1991.
12. W. Humphrey,“ Managing the Software Process,”*Addison Wesley*, 1989.
13. M. C. Paulk, W. Curtis, M. Chrissis and C. V. Weber,“Capability Maturity Model for Software, Version 1.1”, Technical Report, CMU/SEI-93-TR-024, ESC-TR-93-177, February 1993.
14. C. Jones,“ Assessment and Control of Software Risks,” Prentice-Hall, 1994.
15. Software Engineering Institute,“Capability Maturity Model Integration (CMMI) Version 1.2 Overview,” http://www.sei.cmu.edu/cmmi/general/general.html, 2006.
16. Software Engineering Institute,“ Standard CMMI Appraisal Method for Process Improvement (SCAMPISM) A, Version 1.2: Method Definition Document”, CMU/SEI-2006-HB-002.
17. C. B. Tayntor,“ Six Sigma Software Development,”AUERBACH; 1st edition, 2002.
18. L Heinz, “Using Six Sigma in Software Development,” News @ SEI, http://www.sei.cmu.edu/news-at-sei/features/2004/1/feature-3.htm.
19. D. L. Gibson, D. R. Goldenson, and K. Kost,“ Performance Results of CMMI®-Based Process Improvement,” Technical Report, CMU/SEI-2006-TR-004 and ESC-TR-2006-004, August 2006.

M. H. FALLAH
G. T. TUCKER
Stevens Institute of Technology, Hoboken, NJ
Formerly of Lucent Technologies Bell Laboratories, Holmdel, NJ