American Certification Institute

AMERICAN CERTIFICATION INSTITUTE

Clifford R. Kettemborough, Ph.D., DBA

is hereby Certified as a

Certified International Business Analyst Professional
(CIBAP)

As a business management professional, this includes the responsibility to maintain the highest ethical practice to favorably reflect upon the profession.

Given at Lewes, Delaware, the United States.

AMERICAN CERTIFICATION INSTITUTE

Certification ID:	*CIBAP2170901870*
Issue Date:	*September 20, 2017*
Expiration Date:	*September 20, 2022*

[illegible]

President, Certification Committee

CERTIFICATE OF COMPLETION

Clifford Kettemborough

has completed the course

LDP Essentials - Australia

CERTIFICATE OF ACHIEVEMENT

November 1, 2018

Certificate: 12108522

International Purchasing and Supply Chain Management Institute

International Purchasing and Supply Chain Management Institute

IPSCMI

Clifford Kettemborough

is hereby Certified as a

Certified International Purchasing Manager (CIPM)

As a purchasing and supply chain management professional, this includes the responsibility to maintain the highest ethical practice to favorably reflect upon the profession.

Given at Lewes, Delaware, the United States.

Certification ID: *CIPM2170421231*
Issue Date: *April 12, 2017*
Expiration Date: *April 12, 2022*

[illegible]

President, Certification Committee

American Certification Institute

AMERICAN CERTIFICATION INSTITUTE

Clifford Kettemborough

is hereby Certified as a

Certified International Professional Trainer
(CIPT)

As a business management professional, this includes the responsibility to maintain the highest ethical practice to favorably reflect upon the profession.

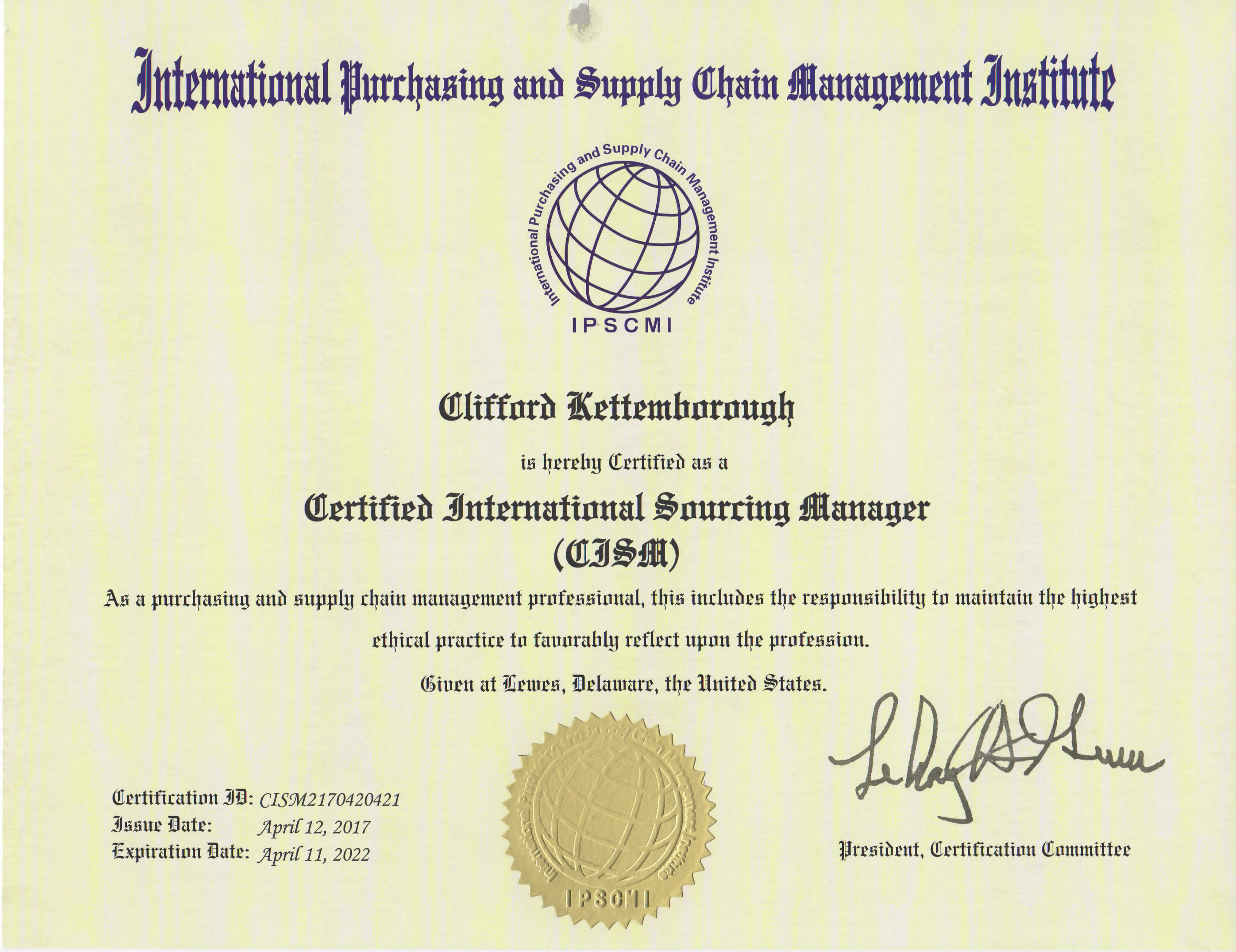
Given at Lewes, Delaware, the United States.

AMERICAN CERTIFICATION INSTITUTE

[illegible]

President, Certification Committee

Certification ID: *CIPT2170440860*
Issue Date: *April 12, 2017*
Expiration Date: *April 12, 2022*



International Purchasing and Supply Chain Management Institute

International Purchasing and Supply Chain Management Institute
IPSCMI

Clifford Kettemborough

is hereby Certified as a

**Certified International Sourcing Manager
(CISM)**

As a purchasing and supply chain management professional, this includes the responsibility to maintain the highest ethical practice to favorably reflect upon the profession.

Given at Lewes, Delaware, the United States.

[illegible]

President, Certification Committee

Certification ID: *CISM2170420421*
Issue Date: *April 12, 2017*
Expiration Date: *April 11, 2022*

IPSCMI

American Certification Institute

AMERICAN CERTIFICATION INSTITUTE

Clifford Kettemborough

is hereby certified as a

Certified International ITIL/ITSM Professional

(ITIL/ITSM)

As an it management professional, this includes the responsibility to maintain the highest ethical practice to favorably reflect upon the profession.

Given at Lewes, Delaware, the United States

Certification ID:	ACI: 2200614106
Issue Date:	June 16, 2017
Expiration Date:	Never

AMERICAN CERTIFICATION INSTITUTE

[illegible]

President, Certification Committee

International Purchasing and Supply Chain Management Institute

International Purchasing and Supply Chain Management Institute

IPSCMI

Clifford Kettemborough

is hereby Certified as a

Certified International Professional Negotiator (CIPN)

As a purchasing and supply chain management professional, this includes the responsibility to maintain the highest ethical practice to favorably reflect upon the profession.

Given at Lewes, Delaware, the United States.

IPSCMI

Certification ID:	*CIPN2200646808*
Issue Date:	*June 16, 2020*
Expiration Date:	*NEVER*

[illegible]

President, Certification Committee

American Certification Institute

AMERICAN CERTIFICATION INSTITUTE

Clifford Kettemborough

is hereby Certified as a

Certified International Professional Training Manager (CIPTM)

As a business management professional, this includes the responsibility to maintain the highest ethical practice to favorably reflect upon the profession.

Given at Lewes, Delaware, the United States.

Certification ID:	*CIPTM2200614316*
Issue Date:	*June 16, 2020*
Expiration Date:	*NEVER*

AMERICAN CERTIFICATION INSTITUTE

[illegible]

President, Certification Committee

International Purchasing and Supply Chain Management Institute

International Purchasing and Supply Chain Management Institute

IPSCMI

Clifford Kettemborough

is hereby Certified as a

Certified International Supply Chain Manager (CISCM)

As a purchasing and supply chain management professional, this includes the responsibility to maintain the highest ethical practice to favorably reflect upon the profession.

Given at Lewes, Delaware, the United States.

Certification ID:	*CISCM2200634730*
Issue Date:	*June 16, 2020*
Expiration Date:	*NEVER*

International Purchasing and Supply Chain Management Institute

IPSCMI

[illegible]

President, Certification Committee

American Certification Institute

AMERICAN CERTIFICATION INSTITUTE

Clifford Kettemborough

is hereby certified as a

Certified International IT Security Manager
(ITSM)

As an it management professional, this includes the responsibility to maintain the highest ethical practice to favorably reflect upon the profession.
Given at Lewes, Delaware, the United States

AMERICAN CERTIFICATION INSTITUTE

Certification ID:	ACI: 2200613107
Issue Date:	May 26, 2016
Expiration Date:	Never

[illegible]

President, Certification Committee

ScrumAlliance®

Clifford R Kettemborough, Ph.D., D.B.A., Ed.D

is awarded the designation Certified ScrumMaster® on this day, June 30, 2010, for completing the prescribed requirements for this certification and is hereby entitled to all privileges and benefits offered by SCRUM ALLIANCE®.

Scrum Alliance CSM CERTIFIED

Certificant ID: 000099653 Certification Active through: 07 November 2022

[illegible]

Certified Scrum Trainer®

[illegible]

Chairman of the Board

ScrumAlliance®

Clifford R Kettemborough, Ph.D., D.B.A., Ed.D

is awarded the designation Certified Scrum Professional® - ScrumMaster on this day, April 12, 2018, for completing the prescribed requirements for this certification and is hereby entitled to all privileges and benefits offered by SCRUM ALLIANCE®.

Scrum Alliance CSP-SM CERTIFIED

Certificant ID: 000099653 Certification Active through: 06 November 2022 Certified Since: 04 November 2010

[illegible]

Chairman of the Board

redstone learning | Creating Value That Lasts!

EZ Certifications STAY RELEVANT. STAY AHEAD

Certificate No: LSSGB.9891.001

Clifford R. Kettemborough

Has satisfactorily fulfilled the requirements established
By Redstone Learning for professional attainment in
Lean Six Sigma Green Belt
The certification acknowledges the technical expertise and the ability to apply quality methodologies to drive business improvement and increase customer satisfaction.

July 14th 2016

[illegible]

Nabin Roy, COO
Redstone Learning

ScrumAlliance®
transforming the world of work

Clifford R. Kettemborough, Ph.D., D.B.A.,

is awarded the designation Certified ScrumMaster
on this day, , for completing
the prescribed requirements for this certification and
is hereby entitled to all privileges and benefits
offered by the Scrum Alliance, Inc.

CSM ScrumAlliance CERTIFIED ScrumMaster™

[illegible]
Certified Scrum Trainer

Mike Cohn
Chairman of the Board

Ed.D

June 30th 2010

ScrumAlliance®
transforming the world of work

Clifford R. Kettemborough, Ph.D., D.B.A.,

is awarded the designation Certified Scrum Professional
on this day, , for completing
the prescribed requirements for this certification and
is hereby entitled to all privileges and benefits
offered by the Scrum Alliance, Inc.

CSP ScrumAlliance® CERTIFIED ScrumProfessional℠

Mike Cohn

Chairman of the Board

Ed.D

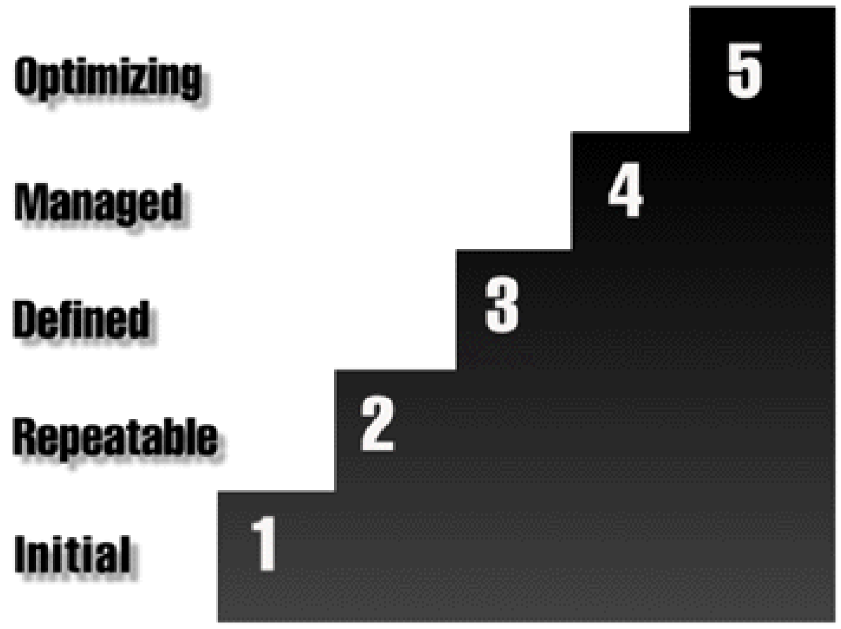
November 4th 2010

Capability Maturity Model® for Software (SW-CMM®)

The Capability Maturity Model for Software (also known as the CMM and SW-CMM) has been a model for judging the maturity of the software processes of an organization for many years now. This model helped organizations identify the key practices required to help them increase the maturity of these processes.

The SW-CMM was developed by the software community with stewardship by the SEI. This model is one of the models that provided the basis for the CMMI Product Suite. As a result, a sunset policy was established by the SEI to help SW-CMM users upgrade to CMMI. For more information about the sunset policy, see How Will Sunsetting of the Software CMM Be Conducted?

The Software CMM became a de facto standard for assessing and improving software processes. Through the SW-CMM, other CMMs, and now CMMI, the SEI and process improvement community established an effective means of modeling, defining, and measuring the maturity of the processes used by organizations developing and maintaining software-intensive systems.



For more information about the SW-CMM legacy model see

- A brief summary of SW-CMM concepts
- Links to SW-CMM model documents
- Links to CMM-related articles

- Who to contact for more information

Return to top of the page

- Welcome
- Capability Maturity Modeling
- Team & Personal Software Process
- IDEAL Model
- Risk Management
- Software Engineering Measurement & Analysis (SEMA)
- Software Engineering Information Repository (SEIR)
- Software Process Improvement Networks (SPINs)
- Appraiser Program
- Acronyms
- SEI Initiatives
- Conferences
- Education & Training

Capability Maturity Models®

The SEI is often identified with its CMM® work. Over the years, the SEI has developed six Capability Maturity Model products. Some are new and build on the work of the older ones.

CMMs that the SEI is currently involved in developing, expanding, or maintaining are

- CMMI®(Capability Maturity Model Integration)
- P-CMM (People Capability Maturity Model)
- SA-CMM (Software Acquisition Capability Maturity Model)

Legacy CMMs that have been incorporated into CMMI models, and therefore are no longer maintained are

- Capability Maturity Model for Software (SW-CMM)
- Systems Engineering Capability Maturity Model (SE-CMM)
- Integrated Product Development Capability Maturity Model (IPD-CMM)

SEI work that is very closely related to the development, support, and maintenance of CMMs includes

- Publishing appraisal results - in the Maturity Profile
- Working with standards organizations to help further the cause of for software process improvement
- Improving and supporting CMM-based appraisals of organizations

The SEI's goals in developing CMMs include

- addressing software engineering and other disciplines that have an affect on software development and maintenance
- providing integrated process improvement reference models
- building broad community consensus
- harmonizing with related standards
- enabling efficient improvement across disciplines relevant to software development and maintenance

---

Legacy Capability Maturity Models® (CMMs®)

CMM products that have been incorporated into CMMI® products include three legacy models. The SW-CMM, SE-CMM, IPD-CMM, and EIA 731 provided the basis for the CMMI Product Suite, initially released in 2001. The development characteristics and delivery methods of these models were integrated to form CMMI products that enable users to reduce the cost of performing appraisals and implementing improvements as they pursue enterprise-wide process improvement.

The CMMI Product Suite includes a framework for generating CMMI models and a set of CMMI models produced by the framework. The framework includes common elements and best features of the legacy models as well as rules and methods for generating CMMI models. Discipline-specific elements (e.g., software, systems engineering) of the CMMI Product Suite enable organizations to select elements most important to them.

For more information about CMMI, see Capability Maturity Model Integration (CMMI) and CMMI Frequently Asked Questions (FAQ)

For more information about the legacy CMMs, see

- Capability Maturity Model for Software (SW-CMM)
- Systems Engineering Capability Maturity Model (SE-CMM)
- Integrated Product Development Capability Maturity Model (IPD-CMM)

MANAGEMENT

- Welcome
- Capability Maturity Modeling
- Team & Personal Software Process
- IDEAL Model
- Risk Management
- Software Engineering Measurement & Analysis (SEMA)
- Software Engineering Information Repository (SEIR)
- Software Process Improvement Networks (SPINs)
- Appraiser Program
- Acronyms
- SEI Initiatives
- Conferences
- Education & Training

Capability Maturity Model® (SW-CMM®) for Software

The Capability Maturity Model for Software describes the principles and practices underlying software process maturity and is intended to help software organizations improve the maturity of their software processes in terms of an evolutionary path from ad hoc, chaotic processes to mature, disciplined software processes. The CMM is organized into five maturity levels:

1) Initial. The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.

2) Repeatable. Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

3) Defined. The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.

4) Managed. Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.

5) Optimizing. Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

Predictability, effectiveness, and control of an organization's software processes are believed to improve as the organization moves up these five levels. While not rigorous, the empirical evidence to date supports this belief.

Except for Level 1, each maturity level is decomposed into several key process areas that indicate the areas an organization should focus on to improve its software process.

The key process areas at Level 2 focus on the software project's concerns related to establishing basic project management controls. They are Requirements Management, Software Project Planning, Software Project Tracking and Oversight, Software Subcontract Management, Software Quality Assurance, and Software Configuration Management.

The key process areas at Level 3 address both project and organizational issues, as the organization establishes an infrastructure that institutionalizes effective software engineering and management processes across all projects. They are Organization Process Focus, Organization Process Definition, Training Program, Integrated Software Management, Software Product Engineering, Intergroup Coordination, and Peer

Reviews.

The key process areas at Level 4 focus on establishing a quantitative understanding of both the software process and the software work products being built. They are Quantitative Process Management and Software Quality Management.

The key process areas at Level 5 cover the issues that both the organization and the projects must address to implement continual, measurable software process improvement. They are Defect Prevention, Technology Change Management, and Process Change Management.

Each key process area is described in terms of the key practices that contribute to satisfying its goals. The key practices describe the infrastructure and activities that contribute most to the effective implementation and institutionalization of the key process area.

For a more detailed overview of the CMM, see:

- Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, and Charles V. Weber, "Capability Maturity Model, Version 1.1," IEEE Software, Vol. 10, No. 4, July 1993, pp. 18-27.

or the CMM itself. Version 1.1 of the CMM, which was released in 1993, is now available as a book:

- Carnegie Mellon University, Software Engineering Institute (Principal Contributors and Editors: Mark C. Paulk, Charles V. Weber, Bill Curtis, and Mary Beth Chrissis), *The Capability Maturity Model: Guidelines for Improving the Software Process*, ISBN 0-201-54664-7, Addison-Wesley Publishing Company, Reading, MA, 1995.

For information on the benefits of CMM-based software process improvement, see:

- James Herbsleb, Anita Carleton, et al., "Benefits of CMM-Based Software Process Improvement: Initial Results," Software Engineering Institute, CMU/SEI-94-TR-13, August 1994.
- Patricia K. Lawlis, Robert M. Flowe, and James B. Thordahl, "A Correlational Study of the CMM and Software Development Performance," Crosstalk: The Journal of Defense Software Engineering, Vol. 8, No. 9, September 1995, pp. 21-25.

Also see the CMM-related articles.

QUESTIONS

Return to top of the page

Return to main page

Software CMM® Articles and Papers

Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, and Charles V. Weber, "Capability Maturity Model, Version 1.1," IEEE Software, Vol. 10, No. 4, July 1993, pp. 18-27.

This paper provides an overview of the latest version of the Capability Maturity Model for Software, CMM v1.1. Based on over six years of experience with software process improvement and the contributions of hundreds of reviewers, CMM v1.1 describes the software engineering and management practices that characterize organizations as they mature their processes for developing and maintaining software. This paper stresses the need for a process maturity framework to prioritize improvement actions, describes the process maturity framework of five maturity levels and the associated structural components, and discusses future directions for the CMM.

Mark C. Paulk, "A Comparison of ISO 9001 and the Capability Maturity Model for Software," Software Engineering Institute, CMU/SEI-94-TR-12, July 1994.

The Capability Maturity Model for Software (CMM), developed by the Software Engineering Institute, and the ISO 9000 series of standards, developed by the International Standards Organization, share a common concern with quality and process management. The two are driven by similar concerns and intuitively correlated. The purpose of this report is to contrast the CMM and ISO 9001, showing both their differences and their similarities. The results of the analysis indicate that, although an ISO 9001- compliant organization would not necessarily satisfy all of the level 2 key process areas, it would satisfy most of the level 2 goals and many level 3 goals. Because there are practices in the CMM that are not addressed in ISO 9000, it is possible for a level 1 organization to receive 9001 registration; similarly, there are areas addressed by ISO 9001 that are not addressed in the CMM. A level 3 organization would have little difficulty in obtaining ISO 9001 certification, and a level 2 organization would have significant advantages in obtaining certification.

This report is based on the 1987 release of ISO 9001. The paper "How ISO 9001 Compares With the CMM," IEEE Software, January 1995, is less detailed, but it is based on the 1994 release of ISO 9001. Note that Figure 1 has become corrupted in the on-line version of the paper.

Mark C. Paulk, "Questions and Answers on the CMM," Issue #1, 5 April 1994.

Discusses tailoring the CMM, the key process area template for CMM v1.1, and documentation for small/prototyping projects.

PDF

Mark C. Paulk, "Questions and Answers on the CMM," Issue #2, 2 August 1994.

Questions and answers on general topics, Requirements Management, Software Project Planning, Software Quality Assurance, Organization Process Definition, Training Program, Integrated Software Management, and Peer Reviews.

PDF

Mark C. Paulk, "Questions and Answers on the CMM," Issue #3, 29 March 1996.

Questions and answers on Intergroup Coordination and Training Program.

PDF

Mark C. Paulk, "Questions and Answers on the CMM," Issue #4, 7 April 1997.

Questions and answers on general topics, including TQM and CMM, Organizational analysis, If the customer won't pay, Tailoring, Small projects, Common threads in the CMM, Incremental development, Incremental process improvement, Legacy systems and maintenance documentation, Software project dynamics, Not Applicable versus risk, Discipline versus bureaucracy, COTS, Required overtime, Methodologies, and Regular versus periodic.

PDF

Mark C. Paulk, "The Rational Planning of (Software) Projects," **Proceedings of the First World Congress for Software Quality**, ASQC, San Francisco, CA, 20-22 June 1995, section 4.

The software crisis has persisted for decades. Our difficulties in planning and managing software projects may be rooted in fundamental human nature, as suggested by research in rational decision making, more than in the inherent difficulty of building software. The Capability Maturity Model for Software, an application of the concepts of Total Quality Management to software development and maintenance, embodies one approach for improving the software process. The problems addressed by both the CMM and TQM seem to lie in the basic ways that human beings think and organize themselves. In many circumstances, normal human decision making can be characterized as "irrational" because of systematic biases and fallacies in the way people make decisions. Mechanisms such as those suggested by the CMM support rational decision making.

Mark C. Paulk, "Mapping from SW-CMM v1.1 to SE-CMM v1.1." Working paper. 31 May 1996.

Mark C. Paulk, "Effective CMM-Based Process Improvement," **Proceedings of the 6th International Conference on Software Quality**, Ottawa, Canada, 28-31 October 1996, pp. 226-237.

The Capability Maturity Model for Software developed by the Software Engineering Institute has had a major influence on software process and quality improvement around the world. Although the CMMSM has been widely adopted, there remain many misunderstandings about how to use it effectively for business-driven software process improvement. This paper discusses how to use the CMM correctly and effectively. It also discusses aspects of successful process improvement efforts that are not explicitly addressed by the CMM, but which are critical to achieving business and process improvement goals.

Mark C. Paulk, " Software Process Proverbs," Crosstalk: The Journal of Defense Software Engineering, Vol. 10, No. 1, January 1997, pp. 4-7.

The Software Engineering Institute (SEI) has developed several aids to help software organizations define, manage, and improve their software processes: the IDEAL (initiate, diagnose, establish, act, and leverage) model for process improvement, appraisal methods that include assessments for internal process improvement and evaluations to select and monitor contractors, and the Capability Maturity Model for Software (CMM or SW-CMM). This article focuses on the CMM and states some software process proverbs (principles, observations, axioms, assumptions) that underlie the CMM and its use that may help people better understand the appropriate application of the SEI's work.

Mark C. Paulk, "Mappings between ISO 12207, ISO 15504 (SPICE), Software CMM v1.1, and Software CMM v2 Draft C." Working paper. 23 February 1998.

Mark C. Paulk, "Using the Software CMM in Small Organizations," **The Joint 1998 Proceedings of the Pacific Northwest Software Quality Conference and the Eighth International Conference on Software Quality**, Portland, Oregon, 13-14 October 1998, pp. 350-361.

The Capability Maturity Model for Software developed by the Software Engineering

Institute has had a major influence on software process and quality improvement around the world. Although the CMM has been widely adopted, there remain many misunderstandings about how to use it effectively for business-driven software process improvement, particularly for small organizations and small projects. Some of the common problems with interpreting the Software CMM for the small project/organization include:

- What does "small" mean? In terms of people? Time? Size of project? Criticality of product?
- What are the CMM "requirements"? Are there key process areas or goals that should not be applied to small projects/organizations? Are there "invariants" of good processes?
- What are the drivers and motivations that cause abuse of the CMM?

This paper discusses how to use the CMM correctly and effectively in any business environment, with examples for the small organization. The conclusion is that the issues associated with interpreting the Software CMM for the small project or organization may be different in degree, but they are not different in kind, from those for any organization interested in improving its software processes. Using the Software CMM effectively and correctly requires professional judgment and an understanding of how the CMM is structured to be used for different purposes.

Mark C. Paulk, "Practices of High Maturity Organizations," **The 11th Software Engineering Process Group (SEPG) Conference**, Atlanta, Georgia, 8-11 March 1999.

Over the last few years the Software Engineering Institute has participated in several workshops and site visits with maturity level 4 and 5 software organizations. This paper summarizes the lessons learned from those interactions with high maturity organizations, while preserving the anonymity of the organizations involved. Specific areas of interest include statistical process and quality control and product lines/families, but the observations cover a variety of engineering and management practices, including issues outside the scope of the Capability Maturity Model for Software. A survey was distributed to informally test the anecdotal observations about high maturity practices.

Mark C. Paulk and David Putman, "Assessing a Level 5 Organization," Crosstalk: The Journal of Defense Software Engineering, Vol. 12, No. 5, May 1999, pp. 21-27.

This article describes the assessment of the Ogden Air Logistics Center (ALC) Software Engineering Division (TIS) that resulted in a Capability Maturity Model (CMM) Level 5 rating. It also discusses the issues in preparing for the assessment and reviewing the processes of TIS from both an internal (Putman) and external (Paulk) assessor's viewpoint: concerns going into the assessment and how they were resolved, alternate implementations that were discussed by the assessment team and how the Software CMM practices were judged to be satisfied, and controversial issues that sparked discussion in the assessment team and how a consensus was reached on their resolution. Specific issues include separating process and product assurance responsibilities, stability of continually improving processes and the related data, satisfactory evidence of institutionalization, and adequate implementation of Quantitative Process Management. The article concludes with a description of the challenges that TIS overcame and some of its strengths that may be of use to maturing organizations.

Mark C. Paulk, "Using the Software CMM With Good Judgment," ASQ Software Quality Professional, Vol. 1, No. 3, June 1999, pp. 19-29.

The Capability Maturity Model for Software (CMM) developed by the Software Engineering Institute (SEI) has had a major influence on software process and quality improvement around the world. Although the CMM has been widely adopted, there remain many misunderstandings about how to use it effectively for business-driven software process improvement, particularly for small organizations and small projects. This paper discusses how to use the CMM correctly and effectively in any business environment, with examples for small organizations, rapid prototyping projects, maintenance shops, R&D outfits, and other environments. The conclusion is that the issues associated with interpreting the Software CMM are essentially the same for any organization interested in improving its software processes - the differences are of degree rather than kind. Using the Software CMM effectively and correctly requires professional judgment and an understanding of how the CMM is structured to be used for different purposes.

Mark C. Paulk, " Toward Quantitative Process Management With Exploratory Data Analysis," **Proceedings of the Ninth International Conference on Software Quality**, Cambridge, MA, 4-6 Oct 1999, pp. 35-42.

The Capability Maturity Model for Software is a model for building organizational capability that has been widely adopted in the software community and beyond. The Software CMM is a five-level model that prescribes process improvement priorities for software organizations. Level 4 in the CMM focuses on using quantitative techniques, particularly statistical techniques, for controlling the software process. In statistical process control terms, this means eliminating assignable (or special) causes of variation. Organizations beginning to use quantitative management typically begin by "informally stabilizing" their process. This paper describes typical questions and issues associated with the exploratory data analysis involved in initiating quantitative process management.

Mark C. Paulk, "Analyzing the Conceptual Relationship Between ISO/IEC 15504 (Software Process Assessment) and the Capability Maturity Model for Software," **Proceedings of the Ninth International Conference on Software Quality**, Cambridge, MA, 4-6 Oct 1999, pp. 293-303.

The Capability Maturity Model for Software (Software CMM) is probably the best known and most widely used model world-wide for software process improvement. ISO/IEC 15504 is a suite of standards currently under development for software process assessment, which can be expected to affect the continuing evolution of the Software CMM. This paper discusses the similarities and differences between the two models and how they may influence each other as they both continue to evolve.

Mark C. Paulk, "Structured Approaches to Managing Change," Crosstalk: The Journal of Defense Software Engineering, Vol. 12, No. 11, November 1999, pp. 4-7.

Change management is crucial in today's fast-moving world. Three perspectives may be of value in thinking about change management: internally driven vs. externally driven change; change directed at products and services vs. those directed at design and production; and incremental vs. revolutionary change. A number of structured approaches have been developed for thinking about the implications of change. This paper summarizes three of those approaches: Geoffrey Moore's "crossing the chasm," Robert Fichman and Chris Kemerer's "assimilation gap," and Abdelkader Daghfous and George White's "innovation analysis model."

Mark C. Paulk, Dennis Goldenson, and David M. White, " The 1999 Survey of High Maturity Organizations," Software Engineering Institute, Carnegie Mellon University, CMU/SEI-2000-SR-002, February 2000.

Over the last few years the Software Engineering Institute has investigated the high maturity practices of Maturity Level 4 and 5 software organizations via assessments, site visits, workshops, and surveys. This report summarizes the observations from the 1999 survey of high maturity organizations. Areas covered in the survey include management, engineering, tools and technology, quantitative analysis, and people issues. A specific area of interest is statistical process control, which is addressed in some detail in this report, but the observations cover a variety of engineering and management practices, including issues outside the scope of the Capability Maturity Model for Software.

Mark C. Paulk and Mary Beth Chrissis, "The November 1999 High Maturity Workshop," Software Engineering Institute, Carnegie Mellon University, CMU/SEI-2000-SR-003, March 2000.

A workshop for high maturity organizations was held on November 16-18, 1999, at the Software Engineering Institute (SEI) in Pittsburgh. The purpose of this workshop was to better understand practices that characterize Level 4 and 5 organizations. Topics of discussion included both practices described in the CMM (Capability Maturity Model) and other practices that have a significant impact in mature organizations. Two themes were anticipated to be important to the workshop participants: statistical process control for software and the reliability and credibility of Level 4 and 5 assessments. Additional topics were solicited from the participants on CMM integration, measurement, technology, human issues, and quality assurance. This report contains brief summaries of the high maturity organizations participating in the workshop and the various working group reports.

Mark C. Paulk, "Applying SPC to the Personal Software Process," **Proceedings of the Tenth International Conference on Software Quality**, New Orleans, LA, 16-18 October 2000.

In recent years, a growing number of software organizations have begun to focus on applying the concepts of statistical process control (SPC) to the software process, usually as part of an improvement program based on the Software CMM. There are a number of technical challenges to the successful use of these statistical techniques, primarily centered on the issues associated with high variation between individual software professionals. A growing number of organizations, however, are demonstrating that SPC techniques can be applied to the software process, even if questions remain on the specific processes, measures, and statistical techniques that will provide significant business value. This paper illustrates the application of the XmR control chart to the Personal Software Process.

Note that both the paper and the slides have useful information. The paper primarily deals with the conceptual issues of SPC and PSP; the slides have the results of the analysis that were presented at the conference.

Mark C. Paulk, " Extreme Programming from a CMM Perspective," IEEE Software, Vol. 18, No. 6, November/December 2001, pp. 19-26.

Extreme Programming (XP) has been advocated recently as an appropriate programming method for the high-speed, volatile world of Internet and Web software development. This popular methodology is reviewed from the Capability Maturity Model (CMM) for Software, a five-level model that prescribes process improvement priorities for software organizations. Overviews of both XP and CMM are provided, and XP is critiqued from a Software CMM perspective. The conclusion is that agile methodologies such as XP advocate many good engineering practices, although some practices may be controversial and counter-productive outside a narrow domain. For those interested in process improvement, the ideas in XP should be carefully considered for adoption where appropriate in an organization's business environment since XP can be used to address many of the CMM Level 2 and 3 practices. In turn, organizations using XP should carefully consider the management and infrastructure issues described in the CMM.

Mark Paulk, "List of Maturity Level 4 and 5 Organizations." Periodically revised.

This list of high maturity organizations provides organization names, maturity levels, assessment dates, Lead Assessors, and points of contact. Please be aware of the following issues regarding this list.

- The SEI does not certify companies at maturity levels.
- The SEI does not confirm the accuracy of the maturity levels reported by the Lead Assessors or organizations.
- This list of Level 4 and 5 organizations is by no means exhaustive; we know of other high maturity organizations that have chosen not to be listed.
- The SEI did not use information stored within its Process Appraisal Information System to produce this document.
- The organizations listed gave explicit permission to publish this information.
- No information obtained in confidence was used to produce this list.

Updates to this list should be sent to Mark Paulk (mcp@sei.cmu.edu). Other information on organizational maturity levels is available in the "Process Maturity Profiles of the Software Community" and the "Compiled List of Published Maturity Levels."

Mark C. Paulk, "A Software Process Bibliography." Periodically revised.

This bibliography was developed for people interested in learning more about software process management. It is neither authoritative nor exhaustive and should not be construed as an endorsement for any of the books and papers that may be referenced. In some cases, papers listed present diametrically opposed perspectives and are included to provide a balanced view of the issues.

Presentations related to the Software CMM are also available as PDF slide sets.

Other articles and working papers related to the Software CMM may be found in the Software CMM v2 archives.

Papers published by the IEEE and the ASQ (formerly ASQC) are posted here with their permission.

QUESTIONS

Return to top of the page

Return to main page

MANAGEMENT

- Welcome
- ● Capability Maturity Modeling
- Team & Personal Software Process
- IDEAL Model
- Risk Management
- Software Engineering Measurement & Analysis (SEMA)
- Software Engineering Information Repository (SEIR)
- Software Process Improvement Networks (SPINs)
- Appraiser Program
- Acronyms
- SEI Initiatives
- Conferences
- Education & Training

Obtaining the Software CMM$^{®}$ V1.1

Software CMM Model Documents

The SW-CMM consists of two SEI technical reports. You can download these reports from the SEI Web site from the links provided below:

- Capability Maturity Model for Software, Version 1.1, Paulk, Mark C.; Curtis, Bill; Chrissis, Mary Beth Chrisis, and Weber, Charles, Software Engineering Institute, CMU/SEI-93-TR-24, DTIC Number ADA263403, February 1993.
- Key Practices of the Capability Maturity Model, Version 1.1, Paulk, Mark C.; Weber, Charles V.; Garcia, Suzanne M. Garcia, Chrissis, Mary Beth; and Bush, Marilyn W., Software Engineering Institute, CMU/SEI-93-TR-25, DTIC Number ADA263432, February 1993.

The Software CMM Book

A hard bound copy of the Software CMM, The Capability Maturity Model: Guidelines for Improving the Software Process, is published by the Addison Wesley publishing company as part of the SEI Series on Software Engineering.

Copyright Issues and Commercial Use of Software CMM V1.1

Permission to reproduce the Software CMM V1.1 documents or to prepare derivative works for internal use is granted in the front matter of these documents, provided the copyright and "no warranty" statements are included with all reproductions and derivative works.

Request for permission to reproduce these documents or to prepare derivative works for external and commercial use should be addressed to the SEI Licensing Agent at:

Sarah Strauss
Phone: 412 / 268-3947
E-mail: permission@sei.cmu.edu

For further information regarding the Software CMM and its associated products, contact:

SEI Customer Relations
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
Telephone: 412 / 268-5800
FAX: 412-268-5758
E-mail: customer-relations@sei.cmu.edu

top of the page | CMM main page

The Capability Maturity Model for Software

Mark C. Paulk
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Bill Curtis
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Mary Beth Chrissis
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Charles V. Weber
IBM Federal Systems Company
6300 Diagonal Highway
Boulder, CO 80301

Abstract

This paper provides an overview of the latest version of the Capability Maturity Model for Software, CMM v1.1. Based on over six years of experience with software process improvement and the contributions of hundreds of reviewers, CMM v1.1 describes the software engineering and management practices that characterize organizations as they mature their processes for developing and maintaining software. This paper stresses the need for a process maturity framework to prioritize improvement actions, describes the process maturity framework of five maturity levels and the associated structural components, and discusses future directions for the CMM.

Keywords: capability maturity model, CMM, process maturity framework, software process improvement, process capability, process performance, maturity level, key process area, software process assessment, software capability evaluation.

1 Introduction

After two decades of unfulfilled promises about productivity and quality gains from applying new software methodologies and technologies, organizations are realizing that their fundamental problem is the inability to manage the software process. In many organizations, projects are often excessively late and over budget, and the benefits of better methods and tools cannot be realized in the maelstrom of an undisciplined, chaotic project.

In November 1986, the Software Engineering Institute (SEI), with assistance from the Mitre Corporation, began developing a process maturity framework that would help organizations improve their software process. In September 1987, the SEI released a brief description of the process maturity framework [Humphrey 87a] which was later expanded in Humphrey's book, *Managing the Software Process* [Humphrey89]. Two methods, software process assessment[1] and software capability evaluation[2] and a maturity questionnaire [Humphrey87b] were developed to appraise software process maturity.

After four years of experience with the software process maturity framework and the preliminary version of the maturity questionnaire, the SEI evolved the maturity framework into the Capability Maturity Model for Software (CMM) [Paulk91, Weber91]. The CMM presents sets of recommended practices in a number of key process areas that have been shown to enhance software process capability. The CMM is based on knowledge acquired from software process assessments and extensive feedback from both industry and government.

The Capability Maturity Model for Software provides software organizations with guidance on how to gain control of their processes for developing and maintaining software and how to evolve toward a culture of software engineering and management excellence. The CMM was designed to guide software organizations in selecting process improvement strategies by determining current process maturity and identifying the few issues most

[1] A software process assessment is an appraisal by a trained team of software professionals to determine the state of an organization's current software process, to determine the high-priority software process-related issues facing an organization, and to obtain the organizational support for software process improvement.

[2] A software capability evaluation is an appraisal by a trained team of professionals to identify contractors who are qualified to perform the software work or to monitor the state of the software process used on an existing software effort.

critical to software quality and process improvement. By focusing on a limited set of activities and working aggressively to achieve them, an organization can steadily improve its organization-wide software process to enable continuous and lasting gains in software process capability.

The initial release of the CMM, v1.0, was reviewed and used by the software community during 1991 and 1992. A workshop was held in April, 1992 on CMM v1.0, and was attended by about 200 software professionals. The current version of the CMM, v1.1 [Paulk93a, Paulk93b], is the result of the feedback from that workshop and ongoing feedback from the software community.

1.1 Immature Versus Mature Software Organizations

Setting sensible goals for process improvement requires an understanding of the difference between immature and mature software organizations. In an immature software organization, software processes are generally improvised by practitioners and their management during the course of the project. Even if a software process has been specified, it is not rigorously followed or enforced. The immature software organization is reactionary, and managers are usually focused on solving immediate crises (better known as fire fighting). Schedules and budgets are routinely exceeded because they are not based on realistic estimates. When hard deadlines are imposed, product functionality and quality are often compromised to meet the schedule.

In an immature organization, there is no objective basis for judging product quality or for solving product or process problems. Therefore, product quality is difficult to predict. Activities intended to enhance quality such as reviews and testing are often curtailed or eliminated when projects fall behind schedule.

On the other hand, a mature software organization possesses an organization-wide ability for managing software development and maintenance processes. The software process is accurately communicated to both existing staff and new employees, and work activities are carried out according to the planned process. The processes mandated are usable and consistent with the way the work actually gets done. These defined processes are updated when necessary, and improvements are developed through controlled pilot-tests and/or cost benefit analyses. Roles and responsibilities within the defined process are clear throughout the project and across the organization.

In a mature organization, managers monitor the quality of the software products and the process that produced them. There is an objective, quantitative basis for judging product quality and analyzing problems with the product and process. Schedules and budgets are based on historical performance and are realistic; the expected results for cost, schedule, functionality, and quality of the product are usually achieved. In general, a disciplined process is consistently followed because all of the participants understand the value of doing so, and the necessary infrastructure exists to support the process.

1.2 Fundamental Concepts Underlying Process Maturity

A ***software process*** can be defined as a set of activities, methods, practices, and transformations that people use to develop and maintain software and the associated products (e.g., project plans, design documents, code, test cases, and user manuals). As an organization matures, the software process becomes better defined and more consistently implemented throughout the organization.

Software process capability describes the range of expected results that can be achieved by following a software process. The software process capability of an organization provides one means of predicting the most likely outcomes to be expected from the next software project the organization undertakes.

Software process performance represents the actual results achieved by following a software process. Thus, software process performance focuses on the results achieved, while software process capability focuses on results expected.

Software process maturity is the extent to which a specific process is explicitly defined, managed, measured, controlled, and effective. Maturity implies a potential for growth in capability and indicates both the richness of an organization's software process and the consistency with which it is applied in projects throughout the organization.

As a software organization gains in software process maturity, it institutionalizes its software process via policies, standards, and organizational structures. Institutionalization entails building an infrastructure and a corporate culture that supports the methods, practices, and procedures of the business so that they endure after those who originally defined them have gone.

2 The Five Levels of Software Process Maturity

Continuous process improvement is based on many small, evolutionary steps rather than revolutionary innovations. The staged structure of the CMM is based on principles of product quality espoused by Walter Shewart, W. Edwards Deming, Joseph Juran, and Philip Crosby. The CMM provides a framework for organizing these evolutionary steps into five maturity levels that lay successive foundations for continuous process improvement. These five maturity levels define an ordinal scale for measuring the maturity of an organization's software process and for evaluating its software process capability. The levels also help an organization prioritize its improvement efforts.

A ***maturity level*** is a well-defined evolutionary plateau toward achieving a mature software process. Each maturity level comprises a set of process goals that, when satisfied, stabilize an important component of the software process. Achieving each level of the maturity framework establishes a different component in the software process, resulting in an increase in the process capability of the organization.

Organizing the CMM into the five levels shown in Figure 2.1 prioritizes improvement actions for increasing software process maturity. The labeled arrows in Figure 2.1 indicate the type of process capability being institutionalized by the organization at each step of the maturity framework.

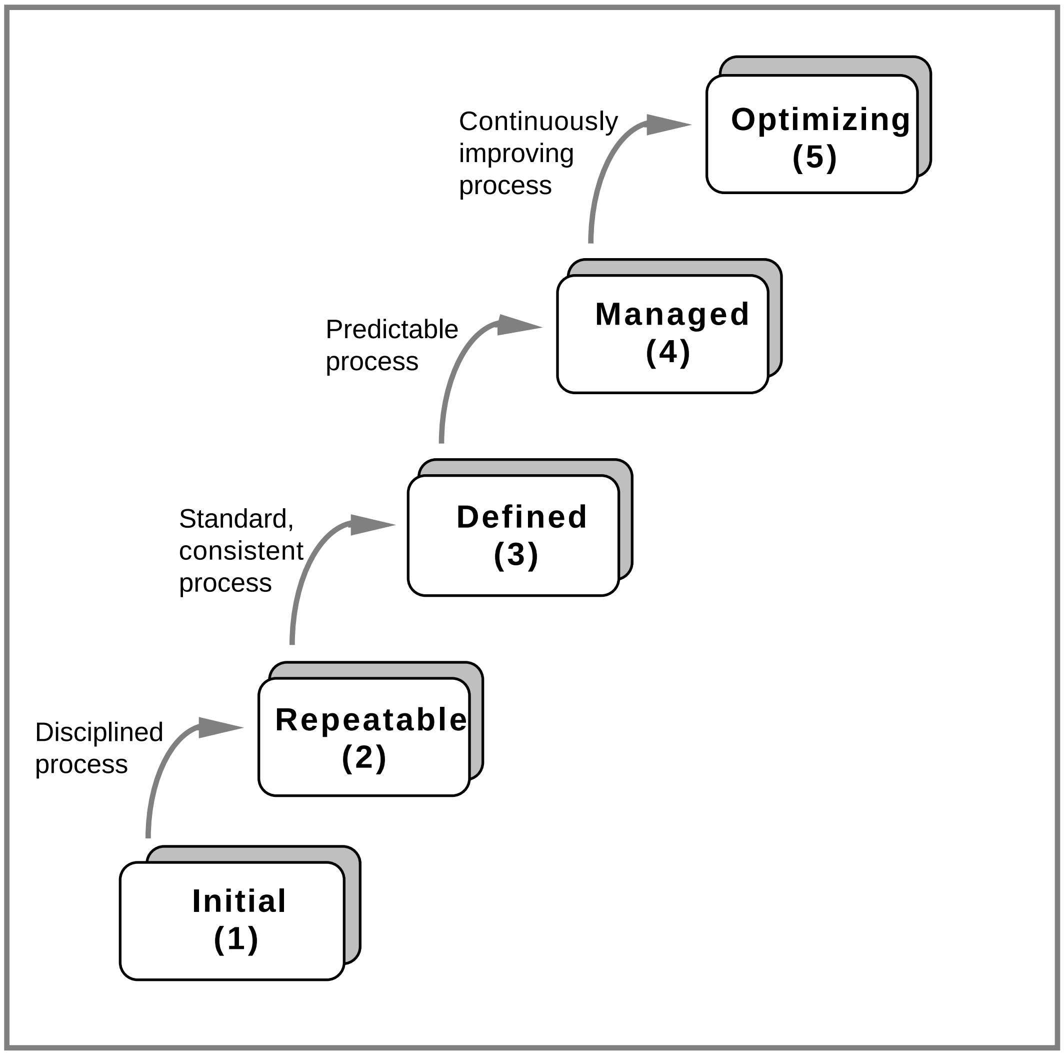


Figure 2.1 The Five Levels of Software Process Maturity

2.1 Behavioral Characterization of the Maturity Levels

Maturity Levels 2 through 5 can be characterized through the activities performed by the organization to establish or improve the software process, by activities performed on each project, and by the resulting process capability across projects. A behavioral characterization of Level 1 is included to establish a base of comparison for process improvements at higher maturity levels.

2.1.1 Level 1 - The Initial Level

At the Initial Level, the organization typically does not provide a stable environment for developing and maintaining software. Such organizations frequently have difficulty making commitments that the staff can meet with an orderly engineering process, resulting in a series of crises. During a crisis, projects typically abandon planned procedures and revert to coding and testing. Success depends entirely on having an exceptional manager and a seasoned and effective software team. Occasionally, capable and forceful software managers can withstand the pressures to take shortcuts in the software process; but when they leave the project, their stabilizing influence leaves with them. Even a strong engineering process cannot overcome the instability created by the absence of sound management practices.

In spite of this ad hoc, even chaotic, process, Level 1 organizations frequently develop products that work, even though they may be over the budget and schedule. Success in Level 1 organizations depends on the competence and heroics of the people in the organization[3] and cannot be repeated unless the same competent individuals are assigned to the next project. Thus, at Level 1, capability is a characteristic of the individuals, not of the organization.

2.1.2 Level 2 - The Repeatable Level

At the Repeatable Level, policies for managing a software project and procedures to implement those policies are established. Planning and managing new projects is based on experience with similar projects. Process capability is enhanced by establishing basic process management discipline on a project by project basis. An effective process can be characterized as one which is practiced, documented, enforced, trained, measured, and able to improve.

Projects in Level 2 organizations have installed basic software management controls. Realistic project commitments are based on the results observed on previous projects and on the requirements of the current project. The software managers for a project track software costs, schedules, and functionality; problems in meeting commitments are identified when they arise. Software requirements and the work products developed to satisfy them are baselined,

---

[3] Selecting, hiring, developing, and/or retaining competent people are significant issues for organizations at all levels of maturity, but they are largely outside the scope of the CMM.

and their integrity is controlled. Software project standards are defined, and the organization ensures they are faithfully followed. The software project works with its subcontractors, if any, to establish a customer-supplier relationship.

Processes may differ between projects in a Level 2 organization. The organizational requirement for achieving Level 2 is that there are policies that guide the projects in establishing the appropriate management processes.

The software process capability of Level 2 organizations can be summarized as disciplined because planning and tracking of the software project is stable and earlier successes can be repeated. The project's process is under the effective control of a project management system, following realistic plans based on the performance of previous projects.

2.1.3 Level 3 - The Defined Level

At the Defined Level, the standard process for developing and maintaining software across the organization is documented, including both software engineering and management processes, and these processes are integrated into a coherent whole. This standard process is referred to throughout the CMM as the organization's standard software process. Processes established at Level 3 are used (and changed, as appropriate) to help the software managers and technical staff perform more effectively. The organization exploits effective software engineering practices when standardizing its software processes. There is a group that is responsible for the organization's software process activities, e.g., a software engineering process group, or SEPG [Fowler90]. An organization-wide training program is implemented to ensure that the staff and managers have the knowledge and skills required to fulfill their assigned roles.

Projects tailor the organization's standard software process to develop their own defined software process, which accounts for the unique characteristics of the project. This tailored process is referred to in the CMM as the project's defined software process. A defined software process contains a coherent, integrated set of well-defined software engineering and management processes.   A well-defined process can be characterized as including readiness criteria, inputs, standards and procedures for performing the work, verification mechanisms (such as peer reviews), outputs, and completion criteria. Because the software process is well defined, management has good insight into technical progress on all projects.

The software process capability of Level 3 organizations can be summarized as standard and consistent because both software engineering and management activities are stable and repeatable. Within established product lines, cost, schedule, and functionality are under control, and software quality is tracked. This process capability is based on a common, organization-wide understanding of the activities, roles, and responsibilities in a defined software process.

2.1.4 Level 4 - The Managed Level

At the Managed Level, the organization sets quantitative quality goals for both software products and processes. Productivity and quality are measured for important software process activities across all projects as part of an organizational measurement program. An organization-wide software process database is used to collect and analyze the data available from the projects' defined software processes. Software processes are instrumented with well-defined and consistent measurements at Level 4. These measurements establish the quantitative foundation for evaluating the projects' software processes and products.

Projects achieve control over their products and processes by narrowing the variation in their process performance to fall within acceptable quantitative boundaries. Meaningful variations in process performance can be distinguished from random variation (noise), particularly within established product lines. The risks involved in moving up the learning curve of a new application domain are known and carefully managed.

The software process capability of Level 4 organizations can be summarized as being quantifiable and predictable because the process is measured and operates within measurable limits. This level of process capability allows an organization to predict trends in process and product quality within the quantitative bounds of these limits. Because the process is both stable and measured, when some exceptional circumstance occurs, the "special cause" of the variation can be identified and addressed. When the known limits of the process are exceeded, action is taken to correct the situation. Software products are of predictably high quality.

2.1.5 Level 5 - The Optimizing Level

At the Optimizing Level, the entire organization is focused on continuous process improvement. The organization has the means to identify weaknesses and strengthen the process proactively, with the goal of preventing the occurrence of defects. Data on the effectiveness of the software process is used to perform cost benefit analyses of new technologies and proposed changes to the organization's software process. Innovations that exploit the best software engineering practices are identified and transferred throughout the organization.

Software project teams in Level 5 organizations analyze defects to determine their causes. Software processes are evaluated to prevent known types of defects from recurring, and lessons learned are disseminated to other projects.

There is chronic waste, in the form of rework, in any system simply due to random variation. Waste is unacceptable; organized efforts to remove waste result in changing the system, i.e., improving the process by changing "common causes" of inefficiency to prevent the waste from occurring. While this is true of all the maturity levels, it is the focus of Level 5.

The software process capability of Level 5 organizations can be characterized as continuously improving because Level 5 organizations are continuously striving to improve the range of their process capability, thereby improving the process performance of their projects. Improvement occurs both by incremental advancements in the existing process and by innovations using new technologies and methods. Technology and process improvements are planned and managed as ordinary business activities.

2.2 Process Capability and the Prediction of Performance

The maturity of an organization's software process helps to predict a project's ability to meet its goals. Projects in Level 1 organizations experience wide variations in achieving cost, schedule, functionality, and quality targets. As illustrated in Figure 2.4, three improvements in meeting targeted goals are expected as the organization's software process matures. These expectations are based on the quantitative results process improvement has achieved in other industries, and they are consistent with the initial case study results reported from software organizations [Dion92, Humphrey91b, Lipke92, Wohlwend93].

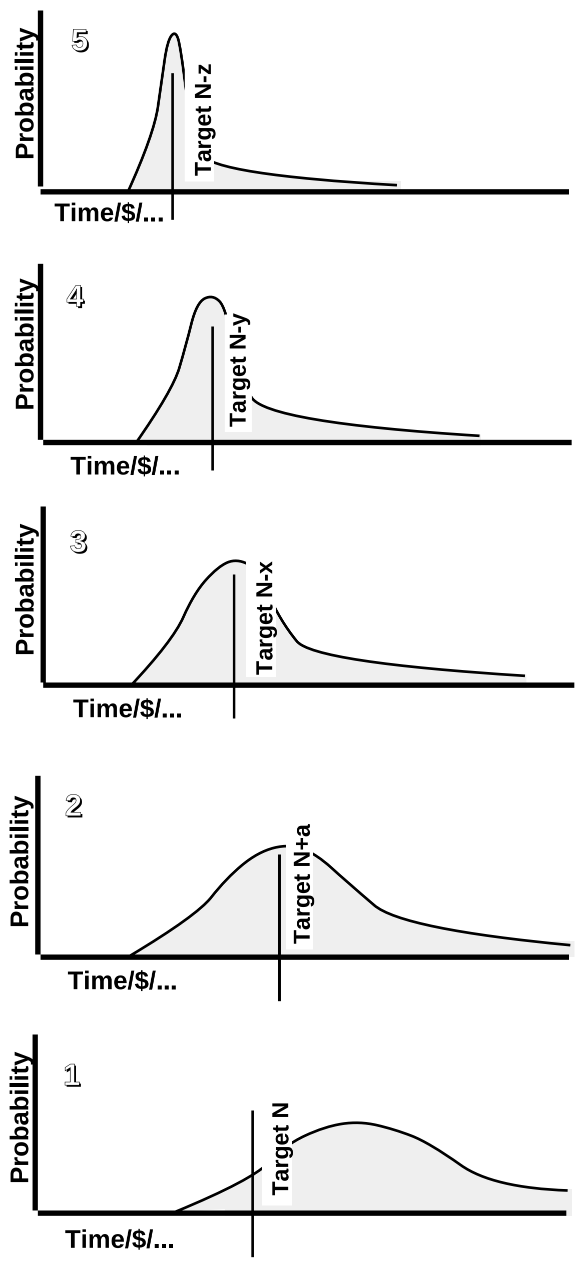
First, as maturity increases, the difference between targeted results and actual results decreases across projects. For instance, Level 1 organizations often miss their originally scheduled delivery dates by a wide margin, whereas higher maturity level organizations should be able to meet targeted dates with increased accuracy. (This is illustrated in Figure 2.4 by how much of the area under the curve lies to the right of the target line.)

Second, as maturity increases, the variability of actual results around targeted results decreases. For instance, in Level 1 organizations delivery dates for projects of similar size are unpredictable and vary widely. Similar projects in a higher maturity level organization, however, will be delivered within a smaller range. (This is illustrated in Figure 2.4 by how much of the area under the curve is concentrated near the target line.)

Third, targeted results improve as the maturity of the organization increases. That is, as a software organization matures, costs decrease, development time becomes shorter, and productivity and quality increase. In a Level 1 organization, development time can be quite long because of the amount of rework that must be performed to correct mistakes. In contrast, higher maturity level organizations have increased process efficiency and reduce costly rework, allowing development time to be shortened. (This is illustrated in Figure 2.4 by the horizontal displacement of the target line from the origin.)

The improvements in predicting a project's results represented in Figure 2.4 assume that the software project's outcomes become more predictable as noise, often in the form of rework, is removed from the software process. Unprecedented systems complicate the picture since new technologies and applications lower the process capability by increasing variability. Even in the case of unprecedented systems, the management and engineering practices characteristic of more mature organizations help identify and address problems earlier in the development cycle than they would have been detected in less mature organizations. In some cases a mature process means that "failed" projects are identified early in the software life cycle and investment in a lost cause is minimized.

The documented case studies of software process improvement indicate that there are significant improvements in both quality and productivity as a result of the improvement effort [Dion92, Humphrey91b, Lipke92, Wohlwend93]. The return on investment seems to typically be in the 5:1 to 8:1 range for successful process improvement efforts.



Performance continuously improves in Level 5 organizations

Based on quantitative understanding of process and product, performance continues to improve in Level 4 organizations

With well-defined processes, performance improves in Level 3 organizations

Plans based on past performance are more realistic in Level 2 organizations

Schedule and cost targets are typically overrun by Level 1 organizations.

Figure 2.4 Process Capability as Indicated by Maturity Level

2.3 Skipping Maturity Levels

Trying to skip levels is counterproductive because each maturity level in the CMM forms a necessary foundation from which to achieve the next level. The CMM identifies the levels through which an organization should evolve to establish a culture of software engineering excellence. Organizations can institute specific process improvements at any time they choose, even before they are prepared to advance to the level at which the specific practice is recommended. However, organizations should understand that the stability of these improvements is at greater risk since the foundation for their successful institutionalization has not been completed. Processes without the proper foundation fail at the very point they are needed most – under stress – and they provide no basis for future improvement.

For instance, a well-defined software process that is characteristic of a Level 3 organization, can be placed at great risk if management makes a poorly planned schedule commitment or fails to control changes to the baselined requirements. Similarly, many organizations have collected the detailed data characteristic of Level 4, only to find that the data were uninterpretable because of inconsistency in the software development processes.

At the same time, it must be recognized that process improvement efforts should focus on the needs of the organization in the context of its business environment, and higher-level practices may address the current needs of an organization or project. For example, when prescribing what steps an organization should take to move from Level 1 to Level 2, frequently one of the recommendations is to establish a software engineering process group (SEPG), which is an attribute of Level 3 organizations. While an SEPG is not a necessary characteristic of a Level 2 organization, they can be a useful part of the prescription for achieving Level 2.

3 Operational Definition of the Capability Maturity Model

The CMM is a framework representing a path of improvements recommended for software organizations that want to increase their software process capability. This operational elaboration of the CMM is designed to support the many ways it will be used. There are at least four uses of the CMM that are supported:

- Assessment teams will use the CMM to identify strengths and weaknesses in the organization.
- Evaluation teams will use the CMM to identify the risks of selecting among different contractors for awarding business and to monitor contracts.
- Upper management will use the CMM to understand the activities necessary to launch a software process improvement program in their organization.
- Technical staff and process improvement groups, such as an SEPG, will use the CMM as a guide to help them define and improve the software process in their organization.

Because of the diverse uses of the CMM, it must be decomposed in sufficient detail that actual process recommendations can be derived from the structure of the maturity levels. This decomposition also indicates the key processes and their structure that characterize software process maturity and software process capability.

3.1 Internal Structure of the Maturity Levels

Each maturity level has been decomposed into constituent parts. With the exception of Level 1, the decomposition of each maturity level ranges from abstract summaries of each level down to their operational definition in the key practices, as shown in Figure 3.1. Each maturity level is composed of several key process areas. Each key process area is organized into five sections called common features. The common features specify the key practices that, when collectively addressed, accomplish the goals of the key process area.

3.2 Maturity Levels

A maturity level is a well-defined evolutionary plateau toward achieving a mature software process. Each maturity level indicates a level of process capability, as was illustrated in Figure 2.1. For instance, at Level 2 the process capability of an organization has been elevated from ad hoc to disciplined by establishing sound project management controls.

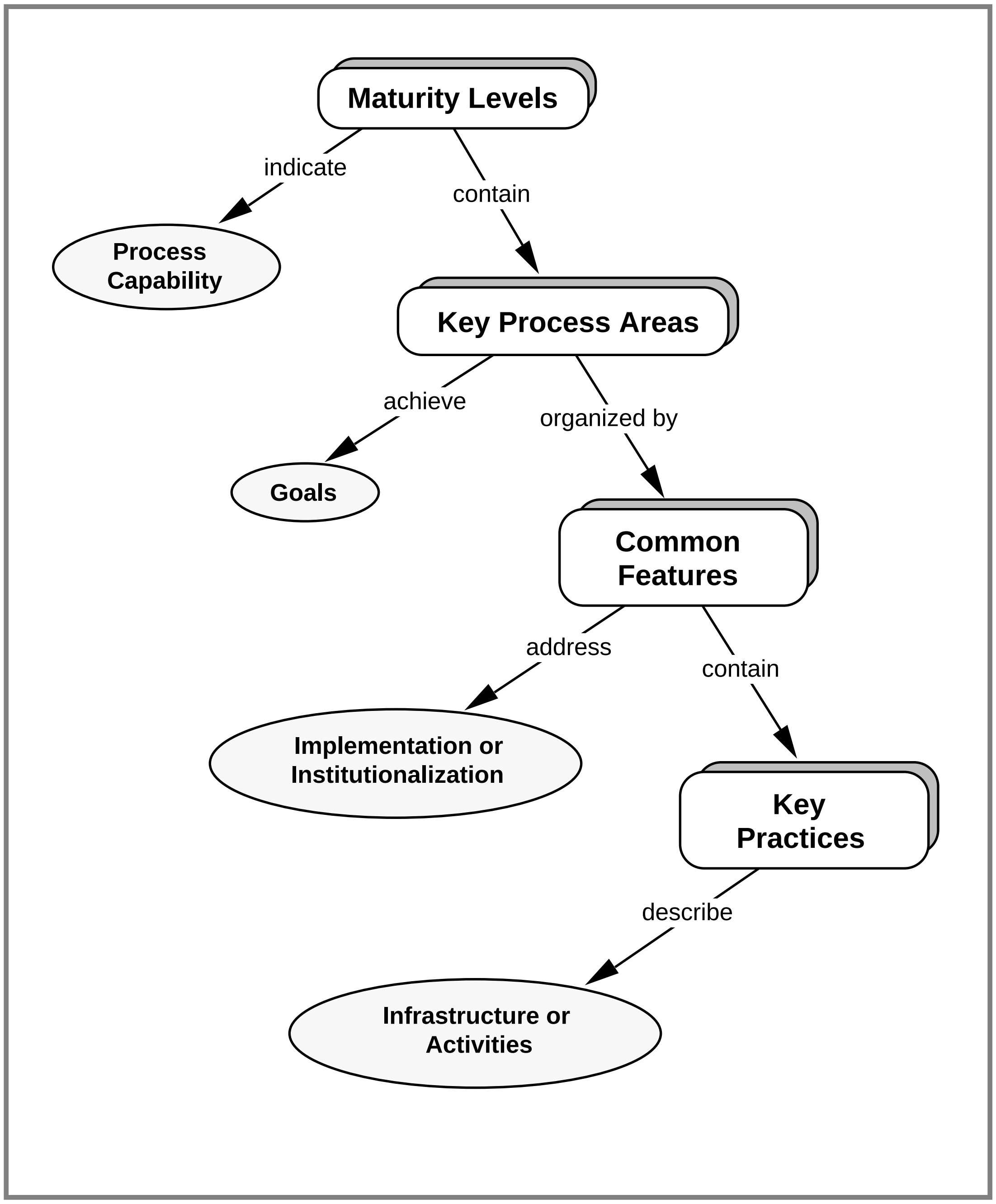


Figure 3.1 The CMM Structure

3.3 Key Process Areas

Except for Level 1, each maturity level is decomposed into several key process areas that indicate where an organization should focus on to improve its software process. Key process areas identify the issues that must be addressed to achieve a maturity level.

Each ***key process area*** identifies a cluster of related activities that, when performed collectively, achieve a set of goals considered important for enhancing process capability. The key process areas have been defined to reside at a single maturity level as shown in Figure 3.2. The path to achieving the goals of a key process area may differ across projects based on differences in application domains or environments. Nevertheless, all the goals of a key process area must be achieved for the organization to satisfy that key process area.

The adjective "key" implies that there are process areas (and processes) that are not key to achieving a maturity level. The CMM does not describe all the process areas in detail that are involved with developing and maintaining software. Certain process areas have been identified as key determiners of process capability; these are the ones described in the CMM.

The key process areas may be considered the requirements for achieving a maturity level. To achieve a maturity level, the key process areas for that level must be satisfied.

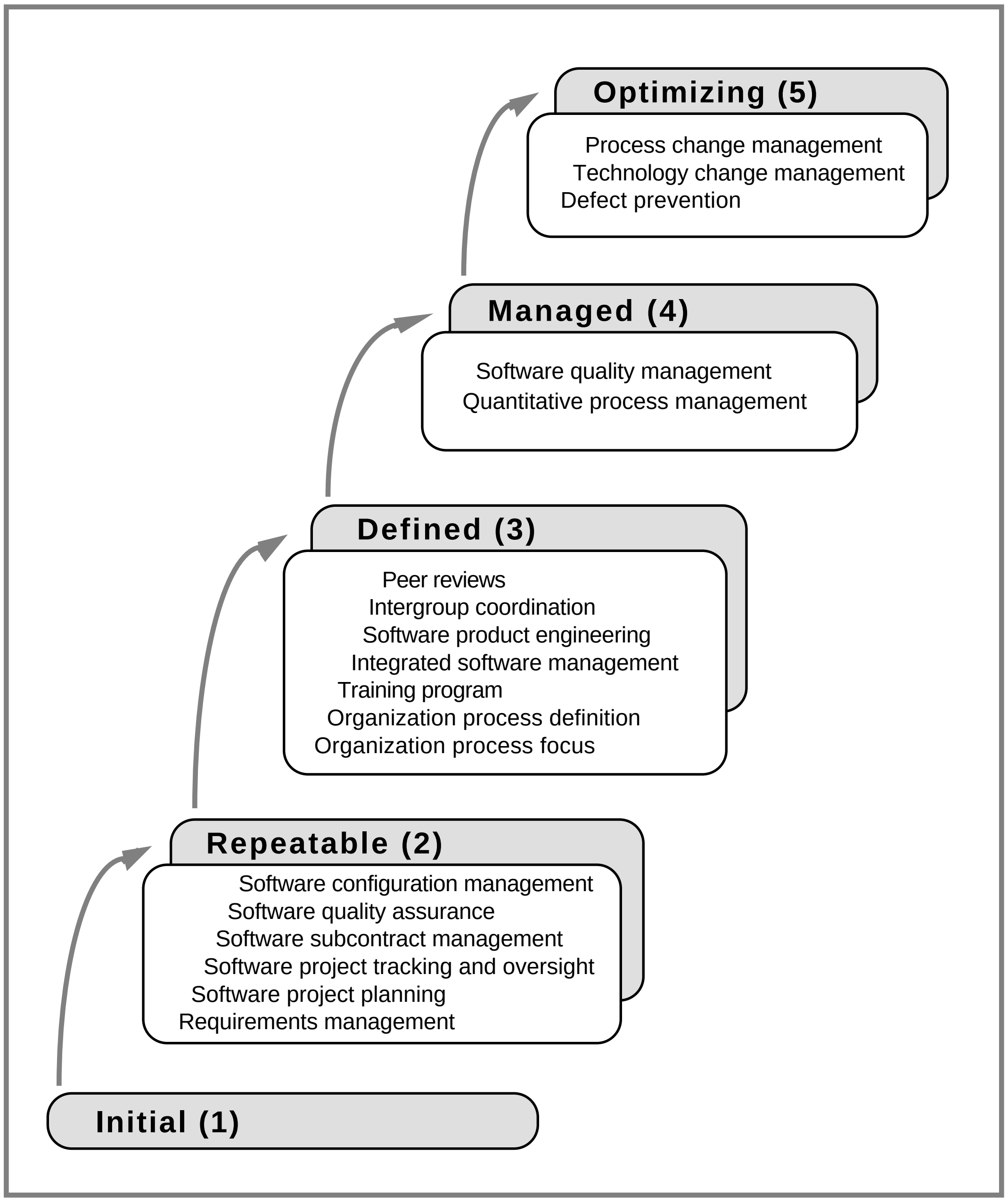

Optimizing (5)
Process change management
Technology change management
Defect prevention
Managed (4)
Software quality management
Quantitative process management
Defined (3)
Peer reviews
Intergroup coordination
Software product engineering
Integrated software management
Training program
Organization process definition
Organization process focus
Repeatable (2)
Software configuration management
Software quality assurance
Software subcontract management
Software project tracking and oversight
Software project planning
Requirements management
Initial (1)

Figure 3.2 The Key Process Areas by Maturity Level

The specific practices to be executed in each key process area will evolve as the organization achieves higher levels of process maturity. For instance, many of the project estimating capabilities described in the Software Project Planning key process area at Level 2 must evolve to handle the additional project data available at Level 3, as is described in Integrated Software Management.

The key process areas at Level 2 focus on the software project's concerns related to establishing basic project management controls.

- ° The purpose of Requirements Management is to establish a common understanding between the customer and the software project of the customer's requirements that will be addressed by the software project. This agreement with the customer is the basis for planning and managing the software project.
- ° The purpose of Software Project Planning is to establish reasonable plans for performing the software engineering and for managing the software project. These plans are the necessary foundation for managing the software project.
- ° The purpose of Software Project Tracking and Oversight is to establish adequate visibility into actual progress so that management can take effective actions when the software project's performance deviates significantly from the software plans.
- ° The purpose of Software Subcontract Management is to select qualified software subcontractors and manage them effectively.
- ° The purpose of Software Quality Assurance is to provide management with appropriate visibility into the process being used by the software project and of the products being built.
- ° The purpose of Software Configuration Management is to establish and maintain the integrity of the products of the software project throughout the project's software life cycle.

The key process areas at Level 3 address both project and organizational issues, as the organization establishes an infrastructure that institutionalizes effective software engineering and management processes across all projects.

- ° The purpose of Organization Process Focus is to establish the organizational responsibility for software process activities that improve the organization's overall software process capability.

- ° The purpose of Organization Process Definition is to develop and maintain a usable set of software process assets that improve process performance across the projects and provide a basis for defining meaningful data for quantitative process management. These assets provide a stable foundation that can be institutionalized via mechanisms such as training.

- ° The purpose of Training Program is to develop the skills and knowledge of individuals so they can perform their roles effectively and efficiently. Training is an organizational responsibility, but the software projects should identify their needed skills and provide the necessary training when the project's needs are unique.

- ° The purpose of Integrated Software Management is to integrate the software engineering and management activities into a coherent, defined software process that is tailored from the organization's standard software process and related process assets. This tailoring is based on the business environment and technical needs of the project.

- ° The purpose of Software Product Engineering is to consistently perform a well-defined engineering process that integrates all the software engineering activities to produce correct, consistent software products effectively and efficiently. Software Product Engineering describes the technical activities of the project, e.g., requirements analysis, design, code, and test.

- ° The purpose of Intergroup Coordination is to establish a means for the software engineering group to participate actively with the other engineering groups so the project is better able to satisfy the customer's needs effectively and efficiently.

- ° The purpose of Peer Reviews is to remove defects from the software work products early and efficiently. An important corollary effect is to develop a better understanding of the software work products and of the defects that can be prevented. The peer review is an important and effective engineering method that can be implemented via inspections, structured walkthroughs, or a number of other collegial review methods.

The key process areas at Level 4 focus on establishing a quantitative understanding of both the software process and the software work products being built.

° The purpose of Quantitative Process Management is to control the process performance of the software project quantitatively. Software process performance represents the actual results achieved from following a software process. The focus is on identifying special causes of variation within a measurably stable process and correcting, as appropriate, the circumstances that drove the transient variation to occur.

° The purpose of Software Quality Management is to develop a quantitative understanding of the quality of the project's software products and achieve specific quality goals.

The key process areas at Level 5 cover the issues that both the organization and the projects must address to implement continuous and measurable software process improvement.

° The purpose of Defect Prevention is to identify the causes of defects and prevent them from recurring. The software project analyzes defects, identifies their causes, and changes its defined software process.

° The purpose of Technology Change Management is to identify beneficial new technologies (i.e., tools, methods, and processes) and transfer them into the organization in an orderly manner. The focus of Technology Change Management is on performing innovation efficiently in an ever-changing world.

° The purpose of Process Change Management is to continually improve the software processes used in the organization with the intent of improving software quality, increasing productivity, and decreasing the cycle time for product development.

3.4 Goals

The ***goals*** summarize the key practices of a key process area and can be used to determine whether an organization or project has effectively implemented the key process area. The goals signify the scope, boundaries, and intent of each key process area. Satisfaction of a KPA is determined by achievement of the goals.

3.5 Common Features

For convenience, the practices that describe the key process areas are organized by common features. The common features are attributes that indicate whether the implementation and institutionalization of a key process area is effective, repeatable, and lasting. The five common features are:

Commitment to Perform	Commitment to Perform describes the actions the organization must take to ensure that the process is established and will endure. Commitment to Perform typically involves establishing organizational policies and senior management sponsorship.
Ability to Perform	Ability to Perform describes the preconditions that must exist in the project or organization to implement the software process competently. Ability to Perform typically involves resources, organizational structures, and training.
Activities Performed	Activities Performed describes the roles and procedures necessary to implement a key process area. Activities Performed typically involve establishing plans and procedures, performing the work, tracking it, and taking corrective actions as necessary.
Measurement and Analysis	Measurement and Analysis describes the need to measure the process and analyze the measurements. Measurement and Analysis typically includes examples of the measurements that could be taken to determine the status and effectiveness of the Activities Performed.
Verifying Implementation	Verifying Implementation describes the steps to ensure that the activities are performed in compliance with the process that has been established. Verification typically encompasses reviews and audits by management and software quality assurance.

The practices in the common feature Activities Performed describe what must be implemented to establish a process capability. The other practices, taken as a whole, form the basis by which an organization can institutionalize the practices described in the Activities Performed common feature.

3.6 Key Practices

Each key process area is described in terms of the key practices that contribute to satisfying its goals. The ***key practices*** describe the infrastructure and activities that contribute most to the effective implementation and institutionalization of the key process area.

Each key practice consists of a single sentence, often followed by a more detailed description, which may include examples and elaboration. These key practices, also referred to as the top-level key practices, state the fundamental policies, procedures, and activities for the key process area. The components of the detailed description are frequently referred to as sub practices. The key practices describe "what" is to be done, but they should not be interpreted as mandating "how" the goals should be achieved. Alternative practices may accomplish the goals of the key process area. The key practices should be interpreted rationally to judge whether the goals of the key process area are effectively, although perhaps differently, achieved. The key practices are contained in the "Key Practices of the Capability Maturity Model, Version 1.1" [Paulk93b], along with guidance on their interpretation.

4 Future Directions of the CMM

Achieving higher levels of software process maturity is incremental and requires a long-term commitment to continuous process improvement. Software organizations may take ten years or more to build the foundation for, and a culture oriented toward, continuous process improvement. Although a decade-long process improvement program is foreign to most U.S. companies, this level of effort is required to produce mature software organizations.

The CMM is not a silver bullet and does not address all of the issues that are important for successful projects. For example, the CMM does not currently address expertise in particular application domains, advocate specific software technologies, or suggest how to select, hire, motivate, and retain competent people. Although these issues are crucial to a project's success, they have not been integrated into the CMM.

During the next few years, the CMM will continue to undergo extensive testing through use in software process assessments, software capability evaluations, and process improvement programs. CMM-based products and training materials will be developed and revised as appropriate. The CMM is a living document that will be improved, but it is anticipated that CMM v1.1 will remain the baseline until at least 1996. This provides an appropriate and realistic balance between the needs for stability and for continued improvement. A book on the CMM is in progress for the SEI series published by Addison-Wesley.

The SEI is also working with the International Standards Organization (ISO) in its efforts to build international standards for software process assessment, improvement, and capability evaluation. This effort will integrate concepts from many different process improvement methods. The development of the ISO standards (and the contributions of other methods) will influence CMM v2.0, even as the SEI's process work will influence the activities of the ISO.

5 Conclusion

The CMM represents a "common sense engineering" approach to software process improvement. The maturity levels, key process areas, common features, and key practices have been extensively discussed and reviewed within the software community. While the CMM is not perfect, it does represent a broad consensus of the software community and is a useful tool for guiding software process improvement efforts.

The CMM provides a conceptual structure for improving the management and development of software products in a disciplined and consistent way. It does not guarantee that software products will be successfully built or that all problems in software engineering will be adequately resolved. However, current reports from CMM-based improvement programs indicate that it can improve the likelihood with which a software organization can achieve its cost, quality, and productivity goals.[Dion92, Humphrey91b, Lipke92, Wohlwend93]

The CMM identifies practices for a mature software process and provides examples of the state-of-the-practice (and in some cases, the state-of-the-art), but it is not meant to be either exhaustive or dictatorial. The CMM identifies the characteristics of an effective software process, but the mature organization addresses all issues essential to a successful project, including people and technology, as well as process.

6 References

Dion92	Raymond Dion, "Elements of a Process-Improvement Program," IEEE Software, Vol. 9, No. 4, July 1992, pp. 83-85.
Fowler90	P. Fowler and S. Rifkin, *Software Engineering Process Group Guide*, Software Engineering Institute, CMU/SEI-90-TR-24, ADA235784, September, 1990.
Humphrey87a	W.S. Humphrey, *Characterizing the Software Process: A Maturity Framework*, Software Engineering Institute, CMU/SEI-87-TR-11, ADA182895, June 1987. Also published in IEEE Software, Vol. 5, No. 2, March 1988, pp.73-79.
Humphrey87b	W.S. Humphrey and W.L. Sweet, *A Method for Assessing the Software Engineering Capability of Contractors*, Software Engineering Institute, CMU/SEI-87-TR-23, ADA187320, September 1987.
Humphrey89	W.S. Humphrey, *Managing the Software Process*, Addison-Wesley, Reading, MA, 1989.
Humphrey91a	W.S. Humphrey, D.H. Kitson, and J. Gale, "A Comparison of U.S. and Japanese Software Process Maturity," *Proceedings of the 13th International Conference on Software Engineering*, Austin, TX, 13-17 May 1991, pp. 38-49.
Humphrey91b	Watts S. Humphrey, T.R. Snyder, and Ronald R. Willis, "Software Process Improvement at Hughes Aircraft," IEEE Software, Vol. 8, No. 4, July 1991, pp. 11-23.
Kitson92	D.H. Kitson and S. Masters, *An Analysis of SEI Software Process Assessment Results: 1987-1991*, Software Engineering Institute, CMU/SEI-92-TR-24, July 1992.
Lipke92	W.H. Lipke and K.L. Butler, "Software Process Improvement: A Success Story," Crosstalk: The Journal of Defense Software Engineering, No. 38, November 1992, pp. 29-31.

Paulk91	M.C. Paulk, B. Curtis, M.B. Chrissis, et al, *Capability Maturity Model for Software*, Software Engineering Institute, CMU/SEI-91-TR-24, ADA240603, August 1991.
Paulk93a	M.C. Paulk, B. Curtis, M.B. Chrissis, and C.V. Weber, *Capability Maturity Model for Software, Version 1.1*, Software Engineering Institute, CMU/SEI-93-TR-24, February 1993.
Paulk93b	M.C. Paulk, C.V. Weber, S. Garcia, M.B. Chrissis, and M. Bush, *Key Practices of the Capability Maturity Model, Version 1.1*, Software Engineering Institute, CMU/SEI-93-TR-25, February 1993.
Weber91	C.V. Weber, M.C. Paulk, C.J. Wise, and J.V. Withey, *Key Practices of the Capability Maturity Model*, Software Engineering Institute, CMU/SEI-91-TR-25, ADA240604, August 1991.
Wohlwend93	H. Wohlwend and S. Rosenbaum, "Software Improvements in an International Company," *Proceedings of the 15th International Conference of Software Engineering*, Washington D.C, May 1993.

The Capability Maturity Model for Software was produced by a dedicated group of people who spent many hours discussing the model and its features and then trying to document it in the two versions of the CMM. Contributors to the CMM, other than the authors, include Edward Averill, Judy Bamberger, Joe Besselman, Marilyn Bush, Anita Carleton, Marty Carlson, Susan Dart, Betty Deimel, Lionel Deimel, Peter Feiler, Julia Gale, Suzie Garcia, Jim Hart, Ron Higuera, Watts Humphrey, Purvis Jackson, Tim Kasse, Richard Kauffold, David Kitson, Mike Konrad, Peter Malpass, Mark Manduke, Steve Masters, Mary Merrill, Judah Mogilensky, Warren Moseley, Jim Over, George Pandelios, Bob Park, Jeff Perdue, Dick Phillips, Mike Rissman, Jim Rozum, Jane Siegel, Christer von Schantz, Cynthia Wise, and Jim Withey.

We appreciate the administrative help from Todd Bowman, Dorothy Josephson, Debbie Punjack, Carolyn Tady, Marcia Theoret, Andy Tsounos, and David White; and the editorial assistance from Suzanne Couturiaux and Bill Pollak.

Special thanks go to the members of the CMM Correspondence Group, who contributed their time and effort to reviewing drafts of the CMM and providing insightful comments and recommendations, and to the members of the CMM Advisory Board, who helped guide us in our efforts. The current members of the Advisory Board are Constance Ahara, Kelley Butler, Bill Curtis, Conrad Czaplicki, Raymond Dion, Judah Mogilensky, Martin Owens, Mark Paulk, Sue Stetak, Charlie Weber, and Ron Willis. Former members who worked with us on CMM v1.0 include Harry Carl, Jim Hess, Jerry Pixton, and Jim Withey.

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

For Further Information

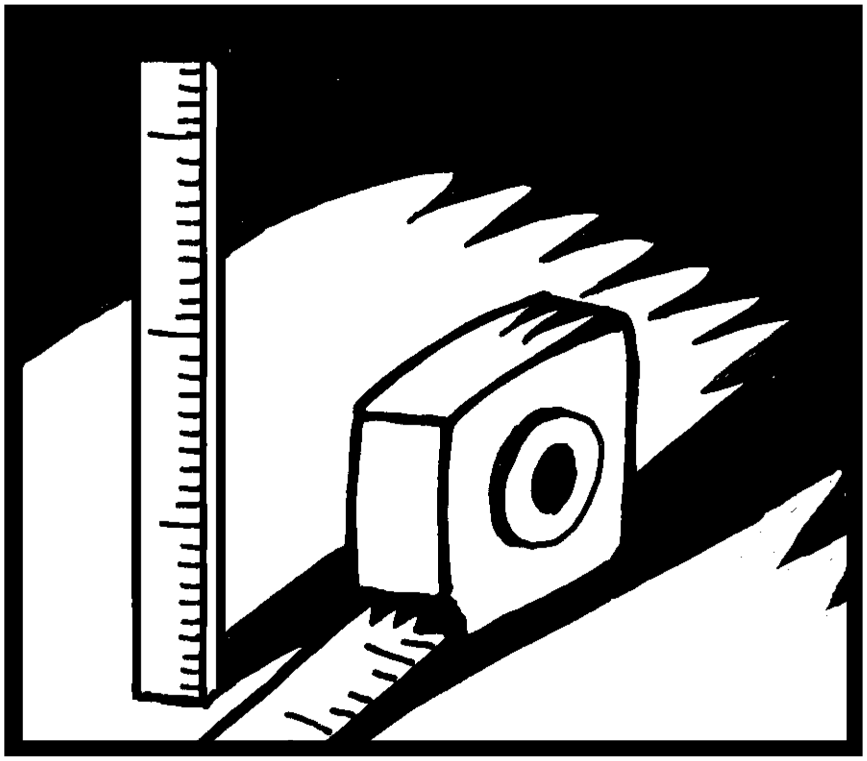
For further information regarding the CMM and its associated products, including training on the CMM and how to perform software process assessments and software capability evaluations, contact:

SEI Customer Relations
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-5800
Internet: customer-relations@sei.cmu.edu

FEATURE

HOW ISO 9001 COMPARES WITH THE CMM

Organizations concerned with ISO 9001 certification often question its overlap with the Software Engineering Institute's Capability Maturity Model. The author looks at 20 clauses in ISO 9001 and maps them to practices in the CMM. The analysis provides answers to some common questions about the two documents.



MARK C. PAULK
Software Engineering Institute

The Capability Maturity Model Model for Software, developed by the Software Engineering Institute, and the ISO 9000 series of standards, developed by the International Organization for Standardization, have the common concern of quality and process management. The two are driven by similar issues and are intuitively correlated, but they differ in their underlying philosophies: ISO 9001, the standard in the 9000 series that pertains to software development and maintenance, identifies the minimal requirements for a quality system, while the CMM underlines the need for continuous process improvement. This statement is somewhat subjective, of course; some members of the international standards community maintain that if you read ISO 9001 with insight, it *does* address continuous process improvement. Corrective action, for example, can be construed as continuous improvement. Nonetheless, the CMM tends to address the issue of continuous process improvement more explicitly than ISO 9001.

This article examines how the two documents relate. I have essentially mapped clauses of ISO 9001 to CMM key practices. The mapping is based on an analysis of ISO 9001, ISO 9000-3, TickIt (a British guide to using ISO 9000-3 and 9001), and the TickIt training materials.[1] ISO 9000-3 elaborates significantly on ISO 9001, while the TickIt training materials help in interpreting both ISO 9000-3 and ISO 9001.

As part of the analysis, I attempt to answer some frequently asked questions, including

♦ At what level in the CMM would an ISO 9001-compliant organization be?

♦ Can a level 2 (or 3) organization be considered compliant with ISO 9001?

♦ Should my software-quality-management and process-improvement efforts be based on ISO 9001 or on the CMM?

I assume the reader is familiar with or has ready access to both ISO 9001 and the CMM. For those who need a refresher, the box on pp.76-77 gives an overview.

MAPPING SPECIFICS

My analysis involved mapping ISO 9001's 20 clauses to CMM key practices at the sentence to subpractice level.[2,3] The analysis is admittedly subjective — others may interpret both ISO 9001 and the CMM differently (indeed, reliable and consistent interpretation and assessment are common challenges for CMM-based appraisals and ISO 9001 certification) — but hopefully there is enough objectivity to make the analysis worthwhile to those who wonder where ISO 9001 certification fits into a continuous quality-improvement strategy.

Table 1 is an overview of the mapping from ISO 9001 clause to CMM key process areas and key practices. The column labeled "Strong relationship" contains key process areas and common features for which the relationship is relatively straightforward. The column labeled "Judgmental relationship" contains key process areas and common features that may require a significant degree of subjectivity in determining a reasonable relationship. Table A in the box on pp. 76-77 describes the focus of the key process areas and common features. In the Activities Performed common feature, key practices focus on systematically implementing a process, while the key practices in other common features focus on institutionalizing it.

Clause 4.1: Management responsibility. ISO 9001 requires an organization to

♦ define, document, understand, implement, and maintain a quality policy;

♦ define responsibility and authority for personnel who manage, perform, and verify work affecting quality; and

♦ identify and provide verification resources.

A designated manager ensures that the quality program is implemented and maintained.

The CMM addresses responsibility for quality policy and verification at level 2. This includes identifying responsibility for performing all project roles, establishing a trained software quality assurance group, and assigning senior management oversight of SQA activities.

As practices within common features, the CMM identifies management's responsibility at both the senior- and project-management levels to oversee the software project, support SQA audits, provide leadership, establish organizational structures to support software engineering, and allocate resources.

You could argue that this clause also addresses the quality policy described at level 4, but the level 4 quality policy is quantitative. ISO 9001 is somewhat ambiguous about the role of measurement in the quality-management system (see discussion under "Clause 4.20: Statistical techniques"); an organization is required to define and document quality objectives, but it does not have to quantify them.

THIS ANALYSIS IS SUBJECTIVE, BUT I HOPE IT IS OBJECTIVE ENOUGH TO BE WORTHWHILE.

Clause 4.2: Quality system. ISO 9001 requires an organization to establish a documented quality system, including a quality manual and plans, procedures, and instructions. ISO 9000-3 characterizes this quality system as an integrated process throughout the life cycle.

The CMM addresses quality-system activities for verifying compliance and for management processes at level 2. The specific procedures and standards a software project would use are specified in the software-development plan. At level 3, the organization must have defined software-engineering tasks that are integrated with management processes, and it must be performing them consistently. These requirements correspond directly with the ISO 9000-3 guidance for interpreting this clause.

As a practice in the Verifying Implementation common feature, the CMM identifies auditing to assure compliance with the specified standards and procedures.

One arguable correspondence is to the software process assets, including standards, procedures, and process descriptions, defined across the organization at level 3. Establishing such organizational assets would certainly contribute to implementing the quality system, but the standards and procedures in this clause could be addressed at the project level. ISO 9001 discusses the supplier's quality system, but it does not specifically address the relationship between organizational support and project implementation, as the CMM does. ISO 9000-3, on the other hand, has two sections on quality planning: clause 4.2.3 discusses quality planning across projects; clause 5.5 discusses quality planning within a particular development.

Clause 4.3: Contract review. ISO 9001 requires organizations to review contracts to determine if requirements are adequately defined, agree with the bid, and can be implemented.

The CMM addresses establishing a contract at level 2. The organization must document and review customer requirements, as allocated to software, and clarify any missing or ambiguous requirements. However, because the CMM is constrained to the software perspective, customer requirements in general are beyond the scope of the Requirements Management key process area.

Also at level 2, the CMM describes the proposal, statement of work, and software-development plan that establish external (contractual) commitments, which the software-engineering group and senior management review.

Finally, the CMM explicitly addresses how the organization can acquire software through subcontracting with an external customer or other type of subcontractor (the supplier may also be a customer). ISO 9001's contract-review clause does not explicitly describe the supplier's role when it is acting as a customer to a subcontractor.

CMM AND ISO 9000 DOCUMENT OVERVIEW

Below are highlights of the Capability Maturity Model Version 1.1 and ISO 9001 and 9000-3, the ISO 9000 standards that apply to software development and maintenance. For more detail on the CMM, see the CMM document.[1,2] For more details on using ISO 9000-3 and 9001, see those documents[3,4] and TickIt, the British guide for applying ISO 9001 to software.[5]

CMM. The Capability Maturity Model describes the principles and practices underlying software-process maturity and is intended to help organizations improve the maturity of their software processes through an evolutionary path from ad hoc, chaotic to mature, disciplined. It may also be used by an organization's customers to identify the strengths, weaknesses, and risks associated with their software suppliers. Authorized appraisers must go through both CMM and appraisal training. (For more information on CMM-based appraisal programs, contact SEI customer relations at (412) 268-5800.)

As Table A shows, the CMM is organized into five levels. Except for level 1, each level has a set of key process areas that an organization should focus on to improve its software process. Each key process area comprises a set of key practices that indicate if the implementation and institutionalization of that area is effective, repeatable, and lasting.

For convenience, the key practices in each key process area are organized by common features:

♦ *Commitment to Perform.* What actions must the organization take to ensure that the process is established and will endure? Includes practices concerning policy and leadership.

♦ *Ability to Perform.* What preconditions must exist in the project or organization to implement the software process competently? Includes practices that concern resources, training, orientation, organizational structure, and tools.

♦ *Activities Performed.* What roles and procedures are necessary to implement a key process area? Includes practices on plans, procedures, work performed, tracking, and corrective action.

♦ *Measurement and Analysis.* What procedures are needed to measure the process and analyze the measurements? Includes practices on process measurement and analysis.

♦ *Verifying Implementation.* What steps are needed to ensure that activities are performed in compliance with the established process? Includes practices on management reviews and audits.

Satisfying a key process area depends on both implementing and institutionalizing the process. Implementation is described in the Activities Performed common feature; institutionalization is described by the other common features.

ISO 9001, 9000-3. The ISO 9000 standards specify quality-system requirements for use when a contract between two parties requires the demonstration of a supplier's capability to design and supply a product. The two parties could be an external client and a supplier, or both could be internal, such as the marketing and engineering groups within the same company.

Of the ISO 9000 series, ISO 9001 is the standard most pertinent to software development and maintenance. Organizations use it when they must ensure that the supplier conforms to specified requirements during several stages of development, including design, development, production, installation, and servicing. ISO 9000-3 provides guidelines for applying ISO 9001 to the development, supply, and maintenance of software.

Organizations typically use ISO 9000 standards to regulate their internal quality system and assure the quality system of their suppliers. In fact, the standards are frequently used to register a third-party's quality system. Certificates of registration have a defined scope within an organization and are issued by quality-system registrars. Auditors are trained in the ISO 9000 standards, but they may not be trained in or knowledgeable about software-specific issues. If the scope of an audit specifies software, software-knowledgeable auditors should be included on the auditing team.

Status. Version 1.1 of the CMM was published in February 1993. The SEI is now collecting change requests and investigating

Clause 4.4: Design control. ISO 9001 requires an organization to establish procedures to control and verify design. These include

♦ planning, design, and development activities;

♦ defining organizational and technical interfaces;

♦ identifying inputs and outputs;

♦ reviewing, verifying, and validating the design; and

♦ controlling design changes.

ISO 9000-3 elaborates this clause with clauses on the purchaser's requirements specification (5.3), development planning (5.4), quality planning (5.5), design and implementation (5.6), testing and validation (5.7), and configuration management (6.1).

The CMM describes the life-cycle activities of requirements analysis, design, code, and test at level 3. Level 2 addresses planning and tracking of all project activities, including these, as

TABLE A
KEY PROCESS AREAS IN THE CMM

Level	Key Process Areas
5 Optimizing Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.	Defect prevention Technology change management Process change management
4 Managed Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.	Quantitative process management Software quality management
3 Defined The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.	Organization process focus Organization process definition Training program Integrated software management Software product engineering Intergroup coordination Peer reviews
2 Repeatable Basic project-management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.	Requirements management Software project planning Software project tracking and oversight Software subcontract management Software quality assurance Software configuration management
1 Initial The software process is characterized as ad hoc, occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.	—

potential additions. The next release, planned for late 1996, may add key process areas and will harmonize the CMM with ISO 9001 and other standards. The ISO 9000 series was published in 1987. A minor revision to ISO 9001 was published in July 1994, and a major revision of the entire series is planned for 1996.

REFERENCES

1. M. Paulk et al., *Capability Maturity Model for Software, Version 1.1*, Tech. Report CMU/SEI-93-TR-24, Software Eng. Inst., Pittsburgh, 1993.
2. M. Paulk et al., *Key Practices of the Capability Maturity Model, Version 1.1*, Tech. Report CMU/SEI-93-TR-25, Software Eng. Inst., Pittsburgh, 1993.
3. *ISO 9000-3: Guidelines for the Application of ISO 9001 to the Development, Supply, and Maintenance of Software*, Int'l Org. for Standardization, Geneva, 1991.
4. ISO 9001: Quality Systems — Model for Quality Assurance in Design/Development, Production, Installation, and Servicing, Int'l Org. for Standardization, Geneva, 1994.
5. *TickIT: A Guide to Software Quality Management System Construction and Certification Using EN29001, Issue 2.0*, UK Dept. of Trade and Industry and the British Computer Society, London, 1992.

TABLE 1
SUMMARY MAPPING BETWEEN ISO 9001 AND THE CMM

ISO 9001 Clause	Strong Relationship	Judgmental Relationship
4.1: Management responsibility	Commitment to perform Software project planning Software project tracking and oversight Software quality assurance	Ability to perform Verifying implementation Software quality management
4.2: Quality system	Verifying implementation Software project planning Software quality assurance Software product engineering	Organization process definition
4.3: Contract review	Requirements management Software project planning	Software subcontract management
4.4: Design control	Software project planning Software project tracking and oversight Software configuration management Software product engineering	Software quality management
4.5: Document and data control	Software configuration management Software product engineering	
4.6: Purchasing	Software subcontract management	
4.7: Control of customer-supplied product	—	Software subcontract management
4.8: Product identification and traceability	Software configuration management Software product engineering	
4.9: Process control	Software project planning Software quality assurance Software product engineering	Quantitative process management Technology change management
4.10: Inspection and testing	Software product engineering Peer reviews	
4.11: Control of inspection, measuring, and test equipment	Software product engineering	
4.12: Inspection and test status	Software configuration management Software product engineering	
4.13: Control of nonconforming product	Software configuration management Software product engineering	
4.14: Corrective and preventive action	Software quality assurance Software configuration management	Defect prevention
4.15: Handling, storage, packaging, preservation, and delivery	—	Software configuration management Software product engineering
4.16: Control of quality records	Software configuration management Software product engineering Peer reviews	
4.17: Internal quality audits	Verifying implementation Software quality assurance	
4.18: Training	Ability to perform Training program	
4.19: Servicing	—	
4.20: Statistical techniques	Measurement and analysis	Organization process definition Quantitative process management Software quality management

well as configuration management of software work products.

ISO 9001, as revised in 1994, requires design reviews. ISO 9000-3 states that the supplier should carry out reviews to ensure that requirements are met and design methods are correctly carried out. However, although design reviews are required, organizations have a range of options for satisfying this clause, from technical reviews to inspections. In contrast, the CMM specifically calls out peer reviews at level 3 and identifies a number of work products that should undergo such a review.

TickIt training clarifies the ISO 9001 perspective by listing three examples of design reviews: Fagan inspections, structured walkthroughs, and peer reviews (in the sense of a desk check). The training also states (on page 17.10) that "an auditor will need to be satisfied from the procedures and records available that the reviews within an organization are satisfactory considering the type and criticality of the project under review."[1]

The CMM describes more formal, quantitative aspects of the design process at level 4, but ISO 9001 does not require this degree of formality.

Clause 4.5: Document and data control. ISO 9001 requires an organization to control the distribution and modification of documents and data. The CMM describes the configuration-management practices characterizing document and data control at level 2. The documentation required to operate and maintain the system is specifically called out at level 3. The specific procedures, standards, and other documents that may be placed under configuration management are identified in the different key process areas in the Activities Performed common feature.

Clause 4.6: Purchasing. ISO 9001 requires organizations to ensure that purchased products conform with specified requirements. This includes evaluating potential subcontractors and verifying purchased products.

The CMM addresses custom software development at level 2, including the evaluation of subcontractors and acceptance testing of subcontracted software.

Clause 4.7: Control of customer-supplied product. ISO 9001 requires an organization to verify, control, and maintain any customer-supplied material. ISO 9000-3 discusses this clause in the context of included software product (6.8), also addressing commercial-off-the-shelf software.

The only CMM practice describing the use of purchased software is a subpractice at level 3, and the context is identifying off-the-shelf or reusable software as part of planning. The integration of off-the-shelf and reusable software is one of the CMM's weaker areas. In fact, this clause, especially as expanded in ISO 9000-3, cannot be considered adequately covered by the CMM. It would be reasonable, though not sufficient, to apply the acceptance testing practice for subcontracted software at level 2 to any included software product.

I have written a change request to CMM version 1.1 to incorporate practices that address product evaluation and the inclusion of off-the-shelf software and other types of software that have not been developed internally.

Clause 4.8: Product identification and traceability. ISO 9001 requires an organization to be able to identify and trace a product through all stages of production, delivery, and installation. The CMM covers this clause primarily at level 2 in the context of configuration management, but states the need for consistency and traceability between software work products at level 3.

Clause 4.9: Process control. ISO 9001 requires an organization to define and plan its production processes. This includes carrying out production under controlled conditions, according to documented instructions. When an organization cannot fully verify the results of a process after the fact, it must continuously monitor and control the process. ISO 9000-3 clauses include design and implementation (5.6); rules, practices, and conventions (6.5); and tools and techniques (6.6).

In the CMM, the specific procedures and standards that would be used in the software-production process are specified in the software-development plan at level 2. The definition and integration of software-production processes, and the tools to support these processes, are described at level 3. Level 4 addresses the quantitative aspect of control, exemplified by statistical process control, but an organization typically would not have to demonstrate this level of control to satisfy this clause. Also, clause 6.6 in ISO 9000-3 states that "the supplier should improve these tools and techniques as required." This corresponds to transitioning new technology into the organization, a level 5 focus.

Clause 4.10: Inspection and testing. ISO 9001 requires an organization to inspect or verify incoming materials before use and to perform in-process inspection and testing. The organization must also perform final inspection and testing before the finished product is released and keep inspection and test records.

I have already described how the CMM deals with issues surrounding the inspection of incoming material ("Clause 4.7: Control of customer-supplied product"). The CMM describes testing and in-process inspections (strictly for software) at level 3.

Clause 4.11: Control of inspection, measuring, and test equipment. ISO 9001 requires an organization to control, calibrate, and maintain any equipment used to demonstrate conformance. When test hardware or software is used, it must be checked before use and rechecked at prescribed intervals. ISO 9000-3 clarifies this clause with clauses on testing and validation (5.7); rules, practices, and conventions (6.5); and tools and techniques (6.6).

The CMM generically addresses this clause under the testing practices in Software Product Engineering. Test software is specifically called out in the Ability to Perform common feature in the practice that describes tools that support testing (Ability 1.2).

Clause 4.12: Inspection and test status. ISO 9001 requires an organization to maintain the status of inspections and tests for items as they move through various processing steps. The CMM addresses this clause with practices on problem reporting and configuration status at level 2 and by testing practices at level 3.

Clause 4.13: Control of nonconforming product. ISO 9001 requires an organization to control a nonconforming product — one that does not satisfy specified requirements — to prevent inadvertent use or installation. ISO 9000-3 maps this concept to clauses on design and implementation (5.6); testing and validation (5.7); replication, delivery, and installation (5.9); and configuration management (6.1).

The CMM does not specifically address nonconforming products. In ISO 9000-3, the control issue essentially disappears among a number of related processes spanning the software life-cycle. In the CMM, the status of configuration items, which would include the status of items that contain known defects not yet fixed, is maintained at level 2. Design, implementation, testing, and validation are addressed at level 3.

Clause 4.14: Corrective and preventive action. ISO 9001 requires an organization to identify the causes of a nonconforming product. Corrective action is directed toward eliminating the causes of actual nonconformities. Preventive action is directed toward eliminating the causes of potential nonconformities. ISO 9000-3 quotes this clause verbatim, with no elaboration, from the 1987 release of ISO 9001.

A literal reading of this clause would imply many of the CMM's practices in the level 5 key process area, Defect Prevention. According to the TickIt auditors' guide[4] (pages 139-140) and discussions with ISO 9000 auditors, corrective action is driven primarily by customer complaints. The software-engineering group should look at field defects, analyze why they occurred, and take corrective action. This would typically occur through software updates and patches distributed to the fielded software.

Under this interpretation of the clause, an appropriate mapping would be to level 2's problem reporting, followed by controlled maintenance of baselined work products.

Another interpretation described in section 23 of the TickIt training literature[1] is that corrective action is to address noncompliance identified in an audit, whether external or internal. This interpretation maps to the CMM's level 2 key process area, Software Quality Assurance.

How you interpret "preventive action" is a controversial issue in applying ISO 9001 to software. Some auditors seem to expect a defect-prevention process similar to that found in a manufacturing environment. Others require only that an organization address user-problem reports. It is debatable how much of the CMM's level 5 in-process causal analysis and defect prevention is necessary to satisfy this clause.

Clause 4.15: Handling, storage, packaging, preservation, and delivery. ISO 9001 requires organizations to establish and maintain procedures for handling, storage, packaging, and delivery. ISO 9000-3 maps this to clauses on acceptance (5.8) and replication, delivery, and installation (5.9).

The CMM does not cover replication, delivery, and installation. It addresses the creation and release of software products at level 2, and acceptance testing at level 3. The CMM does not, however, describe practices for delivering and installing the product. I have written a change request to CMM version 1.1 to incorporate a practice for these areas.

Clause 4.16: Control of quality records. ISO 9001 requires an organization to collect and maintain quality records. In the CMM, the practices defining the maintenance of quality records are distributed throughout the key process areas as part of the Activities Performed common feature. Specific to this clause are the problem reporting described at level 2 and the testing and peer review practices, especially the collection and analysis of defect data, at level 3.

Clause 4.17: Internal quality audits. ISO 9001 requires an organization to plan and perform audits. The results of audits are communicated to management, and any deficiencies found are corrected.

The CMM describes the auditing process at level 2. Auditing practices to ensure compliance with the specified standards and procedures are identified in the Verifying Implementation common feature.

Clause 4.18: Training. ISO 9001 requires an organization to identify training needs, provide training (since selected tasks may require qualified personnel), and maintain training records.

The CMM identifies specific training needs in the training and orientation practices in the Ability to Perform common feature. It describes the general training infrastructure, including maintaining training records, at level 3.

Clause 4.19: Servicing. ISO 9001 requires an organization to perform servicing activities when such activities are part of a specified requirement. ISO 9000-3 addresses this clause as maintenance (5.10).

Although the CMM is intended to

be applied in both the software development and maintenance environments, the practices in the CMM do not directly address the unique aspects that characterize the maintenance environment. Maintenance is embedded throughout the CMM, but organizations must correctly interpret these practices in the development or maintenance context. Maintenance is not, therefore, a separate process in the CMM. Change requests for CMM version 1.0 expressed a concern about using the CMM for maintenance projects, and the SEI changed some wording for CMM version 1.1 to better address the maintenance environment. The SEI anticipates that this will remain a topic of discussion as it provides guidance for tailoring the CMM to different environments, such as maintenance, and begins the next revision cycle for the CMM.

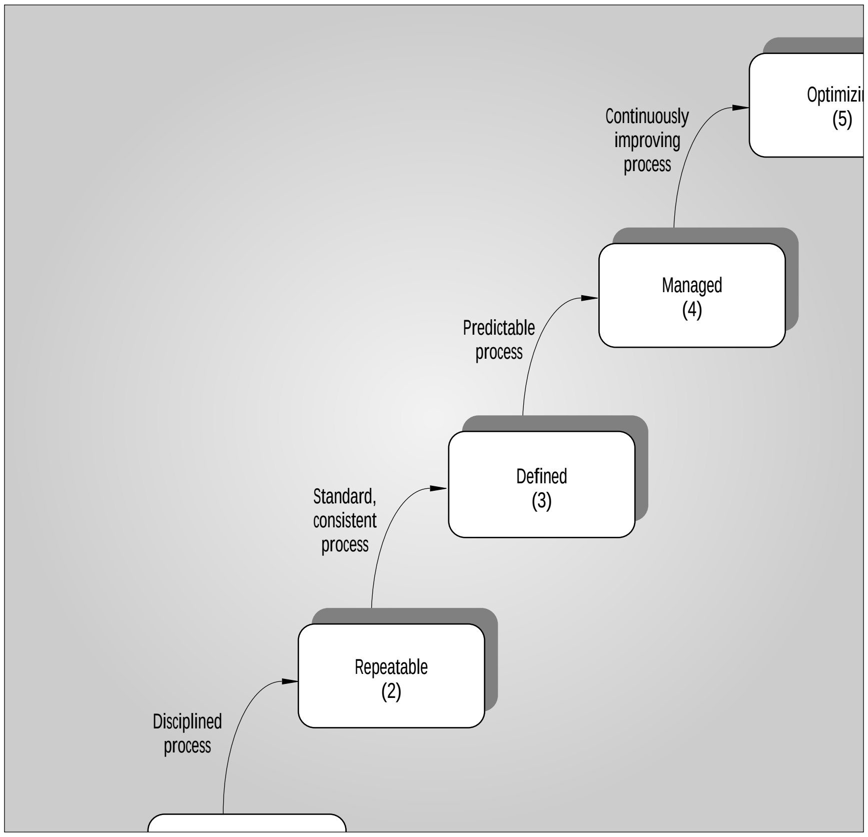


Figure 1. Key process area profile for an ISO 9001-compliant organization. Dark shading represents practices that ISO 9001 or ISO 9000-3 directly address; light shading indicates practices that may be addressed, depending on how you interpret ISO 9001; and unshaded areas indicate practices not specifically addressed.

Clause 4.20: Statistical techniques. ISO 9001 states that organizations must identify adequate statistical techniques and use them to verify the acceptability of process capability and product characteristics. ISO 9000-3 simply characterizes this clause as measurement (6.4).

In the CMM, product measurement is typically incorporated into the various practices within the Activities Performed common feature. Process measurement is described as part of the Measurement and Analysis common feature.

Level 3 describes the establishment of an organization-wide process database for collecting process and product data. It seems likely that most auditors would accept project-level data (as described at level 2) to satisfy this clause. However, at least a few auditors require an organization-level historical database and the use of simple statistical control charts.

If you infer statistical process control from this clause, an organization would satisfy it at level 4. To quote ISO 9000-3, however, "there are currently no universally accepted measures of software quality." Some auditors look for the use of statistical tools, such as Pareto analysis. Others are satisfied by any consistently collected and used measurement data. In general, the only absolute is that auditors vary significantly in how they interpret this clause.

Summary. Clearly there is a strong correlation between ISO 9001 and the CMM, although some issues in ISO 9001 are not covered in the CMM, and vice versa. The level of detail differs significantly: section 4 in ISO 9001 is about five pages long; sections 5, 6, and 7 in ISO 9000-3 comprise about 11 pages; and the CMM is more than 500 pages. Judgment is needed to determine the exact correspondence, given the different levels of abstraction.

As Table 1 shows, the clauses in ISO 9001 with no strong relationships to the CMM key process areas, and that are not well addressed in the CMM, are control of customer-supplied product (4.7) and handling, storage, packaging, preservation, and delivery (4.15). The clause in ISO 9001 that is addressed in the CMM in a completely distributed fashion is servicing (4.19). The clauses in ISO 9001 for which the exact relationship to the CMM is subject to significant debate are corrective and preventive action (4.14) and statistical techniques (4.20).

As I stated earlier, the biggest difference between the two documents is the explicit emphasis of the CMM on continuous process improvement. ISO 9001 addresses only the minimum criteria for an acceptable quality system. Another difference is that the CMM focuses strictly on software, while ISO 9001 has a much broader scope that encompasses hardware, software, processed materials, and services.

The biggest similarity between the two documents is their bottom line: "Say what you do; do what you say." The fundamental premise of ISO 9001 is that organizations should document every important process and check the quality of every deliverable through a quality-control activity. ISO 9001 requires documentation that contains instructions or guidance on what should be done or how it should be done. The CMM shares this emphasis on processes that are documented and practiced as documented. Phrases such as conducted "according to a documented procedure" and following "a written organizational policy" characterize the key process areas in the CMM.

On a more detailed level, some clauses in ISO 9001 are easily mapped to their equivalent CMM practices. Other relationships map in a many-to-many fashion, since the two documents are structured differently. For example, the training clause (4.18) in ISO 9001 maps to both the Training Program key process area and the training and orientation practices in all the key process areas.

EVERY CMM KEY PROCESS AREA IS AT LEAST WEAKLY RELATED TO ISO 9001 IN SOME WAY.

COMPLIANCE ISSUES

At first glance, an organization with an ISO 9001 certificate would have to be at level 3 or 4 in the CMM. In reality, some level 1 organizations have been certified. One reason for this discrepancy is ISO 9001's high level of abstraction, which causes auditors to interpret it in different ways. If the auditor certifying the organization has had TickIt training, for example, the design reviews in ISO 9001 will correspond directly to the CMM's peer reviews, which are at level 3. But not all auditors are well-versed in software development. The virtue of a program like TickIt is that it produces auditors who understand how to apply ISO 9001 to software.

Another reason for the discrepancy is that an auditor may not require mastery to satisfy the corresponding ISO 9001 clause.

Figure 1 shows how an ISO-9001-compliant organization that has implemented no other management or engineering practices except those called out by ISO 9001 rates on the CMM. The size of the bar indicates the percentage of practices within the key process area that are addressed in either ISO 9001 or ISO 9000-3. The figure shows areas that have a direct relationship to clauses in these documents (dark shading), areas for which the relationship is subject to interpretation (light shading), and areas that the clauses do not directly address (white).

Note the following about Figure 1:

♦ Every key process area at level 2 is strongly related to ISO 9001.

♦ Every key process area is at least weakly related to ISO 9001 under some interpretation.

On the basis of this profile, an organization assessed at level 1 could be certified as compliant with ISO 9001. That organization would, however, have to have significant process strengths at level 2 and noticeable strengths at level 3. Private discussions indicate that many level 1 organizations have received ISO 9001 certificates. If an organization is following the spirit of ISO 9001, it is likely to be near or above level 2. However, organizations have identified significant problems during a CMM-based assessment that had not surfaced during a previous ISO 9001 audit.[5] This seems to be related to the greater depth of a CMM-based investigation.

Although the CMM does not adequately address some specific issues, in general it encompasses the concerns of ISO 9001. The converse is less true. ISO 9001 describes only the minimum criteria for an adequate quality-management system, rather than addressing the entire continuum of process improvement, although future revisions of ISO 9001 may address this concern. The differences are sufficient to make a rigid mapping impractical, but the similarities provide a high degree of overlap.

To answer the three questions I listed in the beginning of this article:

♦ An ISO 9001-compliant organization would not necessarily satisfy all the key process areas in level 2 of the CMM, but it would satisfy most of the level 2 and many of the level 3 goals. Further, because ISO 9001 doesn't address all the CMM practices, a level 1 organization could receive ISO 9001 registration.

♦ A level 2 (or 3) organization would probably be considered compliant with ISO 9001 but even a level 3 organization would need to ensure that it adequately addressed the delivery and installation process described in clause 4.15 of ISO 9001, and it should consider the use of included software products, as described in clause 6.8 of ISO 9000-3. With this caveat, obtaining certification should be relatively straightforward for a level 2 or higher organization.

♦ As to whether software process improvement should be based on the CMM or ISO 9001, the short answer is that an organization may want to consider both, given the significant degree of overlap. A market may require ISO 9001 certification; addressing the concerns of the CMM would help organizations prepare for an ISO 9001 audit. Conversely, level 1 organizations would certainly profit from addressing the concerns of ISO 9001. Although either document can be used alone to structure a process-improvement program, the more detailed guidance and software specificity provided by the CMM suggests that it is the better choice, although admittedly this answer may be biased.

In any case, organizations should focus on improvement to build a competitive advantage, not on achieving a score — whether that is a maturity level or a certificate. The SEI advocates addressing continuous process improvement as encompassed by the CMM, but even then there is a need to address the larger business context in the spirit of Total Quality Management. ◆

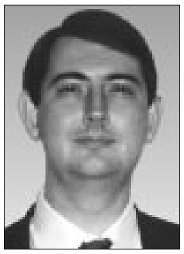
ACKNOWLEDGMENTS

I thank the many people who commented on the early drafts of this article and who discussed the relationships between ISO 9001 and the CMM. In some cases, we have agreed to disagree, but the discussions were always interesting. Specifically, I thank Peter Anderson, Robert Bamford, Kelley Butler, Gary Coleman, Taz Daughtrey, Darryl Davis, Bill Deibler, Alec Dorling, George Kambic, Dwight Lewis, Stan Magee, Helen Mooty, Don O'Neill, Neil Potter, Jim Roberts, John Slater, and Charlie Weber.

This work is sponsored by the US Department of Defense under contract F19628-90-C-003.

REFERENCES

1. *Lloyd's Register TickIT Auditors' Course, Issue 1.4*, Lloyd's Register, Mar. 1994.
2. Mark C. Paulk, "A Comparison of ISO 9001 and the Capability Maturity Model for Software," Tech. Report CMU/SEI-94-TR-2, Software Eng. Inst., Pittsburgh, July 1994.
3. M. Paulk, "Comparing ISO 9001 and the Capability Maturity Model for Software," *Software Quality J.*, Dec. 1993, pp. 245-256.
4. *TickIT: A Guide to Software Quality Management System Construction and Certification Using EN29001, Issue 2.0*, UK Dept. of Trade and Industry and the British Computer Society, London, 1992.
5. F. Coallier, "How ISO 9001 Fits Into the Software World," *IEEE Software*, Jan. 1994, pp. 98-100.

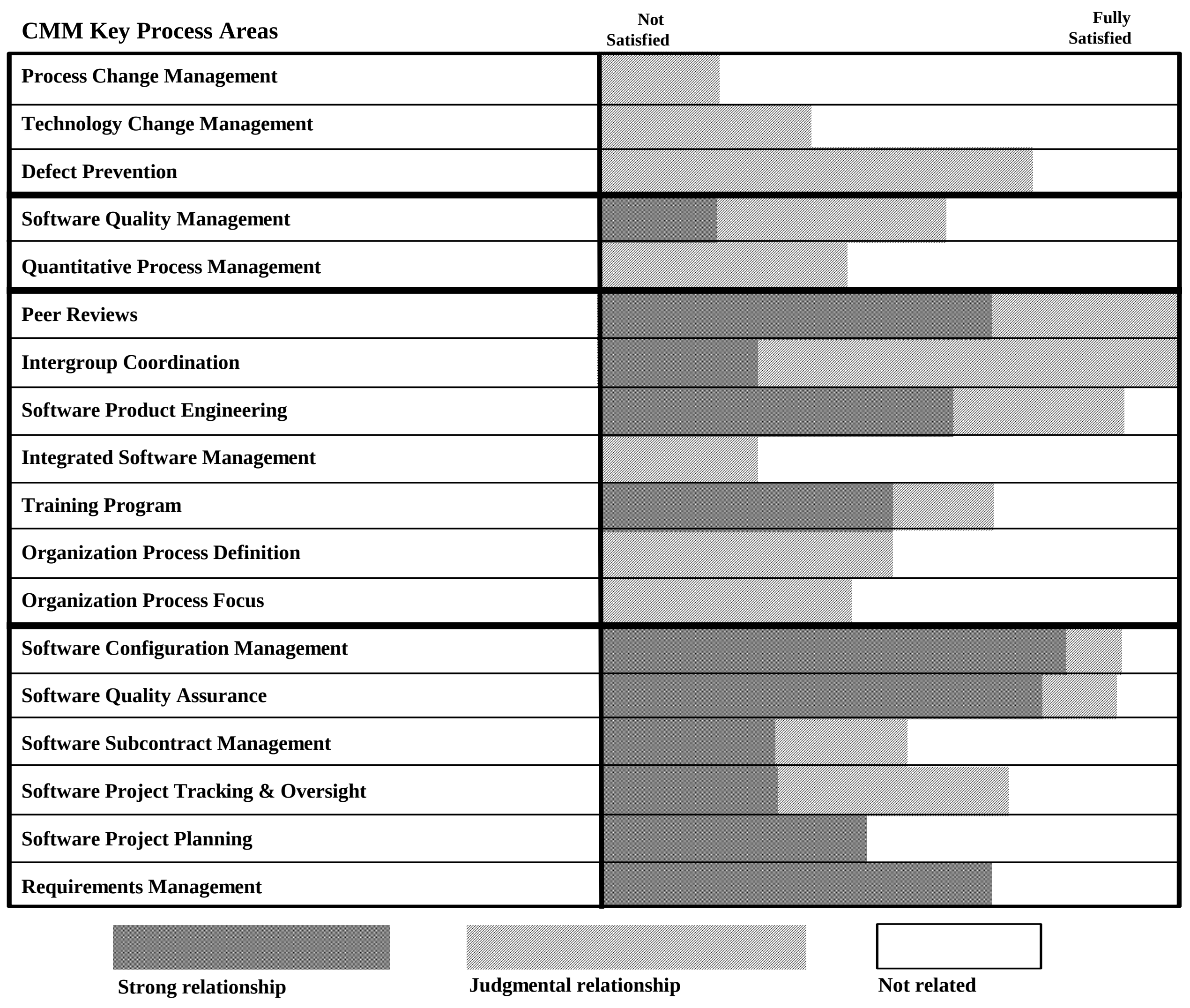


Mark C. Paulk is a senior member of the technical staff at the Software Engineering Institute, where he is product manager for version 2 of the Capability Maturity Model. At the SEI, he was also project leader for the CMM version 1.1 development. Before joining the SEI, Paulk worked on distributed real-time systems for System Development Corp. (later Unisys Defense Systems) at the Ballistic Missile Defense Advanced Research Center.

Paulk received a BS in mathematics from the University of Alabama, Huntsville, and an MS in computer science from Vanderbilt University. He is a senior member of the IEEE and a member of the American Society for Quality Control.

Address questions about this article to Paulk at Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213-3890; mcp@sei.cmu.edu.

KPA Profile for an ISO 9001 Compliant Organization



Questions and Answers about the Software Capability Maturity Model® (SW-CMM® (Q&A #1)

Mark C. Paulk
Issue #1, 5 April 1994

I am frequently asked questions on how to interpret the CMM in various contexts. The following "newsletter" may be of general use. It reproduces the 5 April 1994 mailing to the CMM Correspondence Group. I have to caveat these answers by saying that they represent my opinion and do not represent official SEI positions. They have typically not been internally reviewed, but they do represent a well-informed opinion on what the CMM means.

You are welcome to ask questions on the CMM. I do not guarantee quick answers, although I'll try to get back to you as soon as possible. Sometimes that's the same day I receive a question, sometimes it may be a month, depending on my travel schedule and work load. I prefer to receive questions in writing by e-mail. I will sanitize both question and answer before distribution in this newsletter.

Mark Paulk
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
FAX: 412-268-5758
E-mail: mcp@sei.cmu.edu

The following "questions and answers" are based on discussions held at the CMM workshop on tailoring the CMM held 29-30 March 1994.

Tailoring the CMM

Tailoring the CMM to the specific needs of an organization or class of project is a hot topic right now. These are some of my personal opinions on tailoring; they may or may not be incorporated into the planned SEI report on tailoring guidelines.

As a concept, tailoring lies somewhere between interpreting the CMM and implementing a software process improvement program. I would define interpretation as "a statement, preferably documented, of how an implementation(s) of a process relates to the CMM." I would define tailoring as "a documented description of how a process should be implemented across multiple [1] projects at about the same level of abstraction as the CMM (i.e., what, but shading into how), with a statement of the relationship to the CMM's concepts and practices." A tailored CMM might have to be interpreted when appraising a specific implementation, although the interpretation should be minimal.

The difference between tailoring and interpreting is based on the need for reliability and consistency. Interpretation implies judgement, which may lead to some degree of unreliability and inconsistency. Tailoring may be cost effective if an organization is concerned about SCEs or if the tailoring is a first step in process definition and improvement that will be applied across projects.

Although the key practices and sub-practices describe the normative behavior of large, government contracting organizations, I consider application domain and environment the dominant factors in determining the degree of formality and rigor that is needed by a project. For example, a small commercial project developing life-critical medical software would be expected to follow a formal, rigorous process.

Nonetheless, an organization does not need to implement every practice exactly as stated to fully satisfy a key process area and to achieve its goals. For example, an organization using the Cleanroom approach would not implement some of the testing practices in Software Product Engineering as described.

One reason that tailoring may be needed (at least for consistency and reliability of appraisals) is that a practice or process which is adequate to satisfy the CMM in one context may be inadequate in another context based on the degree of formality and rigor that is needed. For example, I would not normally expect small project SCM to require the same degree of documentation and formality as large project SCM, although I would expect even small projects to achieve the goals of that KPA.

Can key practices be modified?

This decomposes into three cases that I can think of:

1. Mapping of roles, concepts, and terminology. The terminology in the CMM is hopefully reasonably clear, but the different organizational structures, cultures, process implementations, technologies, methodologies, etc., mean that it is necessary to map the roles in the CMM to job titles/functional responsibilities in the organization, to map process concepts to the specific methodologies and technologies of the implemented process(es), and to map key process areas to life cycle activities. Recasting the CMM in the language of an organization can significantly help communication and consistency.
2. Alternate implementations. For example, an organization that does Cleanroom has an alternate implementation for some of the testing practices in Software Product Engineering. Whether one views this as an interpretation or tailoring issue, we certainly want to allow for these kinds of legitimate variations, so long as we understand the relationship between the implemented process and the practices in the CMM.
3. Specific implementations. For example, requiring the use of COCOMO in planning or OO methods in development. This implies a more specific requirement and might be useful in guiding process definition and improvement. This tailoring would, however, be more specific than the CMM and a violation of the tailoring would not necessarily imply that the equivalent practices in the CMM were not satisfactorily addressed.

The first case is very useful in interpreting the CMM. The last two cases edge into defining the organization's standard software process but may still be in the gray area between tailoring and defining.

A version of the CMM with modified practices could be legitimately claimed as "a tailored version of the CMM." This kind of tailoring was the focus of the CMM Tailoring Workshop held on 29-30 March 1994. There are more extensive adaptations of the CMM that might be desirable, however, and these were raised as issues at the workshop.

Can key process areas be modified?

For example, an organization might want to specify inspections for Peer Reviews. Tailoring and process definition have become almost indistinguishable at this point. So long as the modification is more specific than the KPA, satisfying the more specific requirement would satisfy the KPA. Not satisfying a more specific requirement, however, would not necessarily imply that the KPA was not satisfied (although it would indicate that the defined process was not implemented and this is an SQA problem).

I am unable to identify off-hand a reasonable meaning for "less specific variation of a KPA," although I can for a key practice (e.g., for small projects). For example, someone might want to do "Reviews" rather than Peer Reviews and implement them via PDR/CDR kinds of formal review. If such a tailoring was created, it would mean that the CMM KPA (Peer Reviews for the example) was not investigated, and no maturity level could be assigned that depended on the "less specific" KPA.

Modifying key process areas is essentially just a variation of modifying key practices, and I would consider this a tailored version of the CMM, with the caveats on scoring already mentioned. The essential point is a clear understanding of the relationship between the KPA in the CMM (and its goals) and the tailored KPA.

Can key process areas be deleted from the CMM?

That may be a legitimate business decision, but it should imply that those KPAs are explicitly identified as Not Investigated, and no maturity level can be assigned which has "deleted" KPAs at or below it.

I would consider a tailoring that deleted KPAs a legitimate one, but care would have to be taken in reporting the score. It is legitimate to claim that an organization is Level 2 that does not do Software Subcontract Management (i.e., it was scored as Not Applicable), but it would not be legitimate to claim an organization was Level 2 if Software Subcontract Management was tailored out of the CMM. This may seem a subtle distinction, but it is necessary to maintain integrity and comparability of appraisal results. Any KPA that is "not investigated" (as distinct from "not applicable") limits the SEI maturity level that can be assigned and should be noted in briefings and reports.

Can key process areas be added to the CMM?

For example, an organization might identify a need for KPAs on testing and system engineering. This could be a legitimate approach to process improvement (or contract monitoring) to address specific business or project needs, but there are a couple of provisos:

- The new KPAs should be documented, preferably using the template used in the CMM, and available to everyone who needs them.
- The new KPAs should not be assigned to maturity levels, since they are not part of the maturity framework (e.g., no Level 2 KPA on testing).

Adding KPAs is a particular concern for software capability evaluations. Some contractors are concerned that such tailorings prevent a "level playing field" and that the tailoring will be aimed at selecting a particular contractor. The SEI, however, has no control over how acquisition agencies choose to implement SCEs, although we provide guidelines, training, and recommendations to help acquisition agencies implement SCEs consistently. (Also note that industry can and does evaluate software contractors, and there are similar concerns in that environment.)

The CMM provides a publicly available set of criteria for appraising the software process and guiding an improvement effort. Any tailorings of the CMM that an organization uses should be appropriately documented and communicated to all concerned parties in a timely manner. Any tailoring of the CMM should not be advertised as "the" CMM, but rather as "the tailoring of the CMM for XYZ."

The relevant point, as Charlie Weber has pointed out, is whether you are tailoring a top-level key practice or a sub-practice. To tailor a goal, you should sweat blood (and then be conservative in mapping the relationship to "the" CMM). To tailor a key practice, you should just sweat. To tailor a sub-practice, your conscience should hardly bother you.

The CMM is not exhaustive, and it is not a silver bullet. Tailoring the CMM may be appropriate and desirable to address the business needs of an organization. While recognizing that need, we also need to maintain the integrity of the CMM as an asset to the software community.

The key process area template

A template was generally used to express the key process areas using a consistent structure and phrasing. This key process area template, with the "standard" wording used in the various key process areas, follows.

I hope this template will help people understand the CMM and its structure better and help them to use the CMM more efficiently.

There are cases where the practices deliberately deviate from this template. For example, in Verifying Implementation for SQA, the SQA group does not audit itself.

Please note that the numbering of the key practices is not significant, since there are exceptions and additions to the template.

Also note that:

- goals and common features are delimited by underlined headers
- key practices are labeled with a noun representing their common feature and a number, e.g., Commitment 1 and Ability 2
- key practices are in bold type, e.g., **Ability 1 A group that is responsible for exists.**
- sub-practices follow their associated key practice in smaller type and are labeled with a number, e.g., 1.
- practices may be followed by boxed text containing elaboration, cross references, or examples

{Key Process Area X}

The purpose of {Key Process Area X} is {statement}.

{Key Process Area X} involves {summary}.

{Additional elaboration on Key Process Area X as appropriate.}

Goals

Goal 1 - {Process summary statement as goal.}

Goal 2 - {Process summary statement as goal...}

Commitment to perform

Commitment 1 - The project follows a written organizational policy for {X}.

-- or --

The organization follows a written policy for {X}.
This policy typically specifies that:
1. {Sub-practices for Commitment 1...}

Ability to perform

Ability 1 - A group that is responsible for {X} exists.
1. {Sub-practices for Ability 1...}

Ability 2 - Adequate resources and funding are provided for {X}.
1. {Sub-practices for Ability 2...}
2. Tools to support the activities for {X} are made available.

Examples of {X} tools include:
- {examples of tools}

Ability 3 - {Roles} are trained {to perform their X activities}.
-- or --
{Roles} - receive required training {to perform their X activities}.

Examples of training include:
- {examples of training}

Ability 4 - {Roles} receive orientation in {X}.

Examples of orientation include:
- {examples of orientation training}

Activities performed

Activity 1 - {Activity performed in Key Process Area X.}
1. {Sub-practice for Activity 1, possibly affecting different groups.}

Examples of affected groups include:
- {list of affected groups}
----
2. {Additional sub-practices for Activity 1...}
3. {Software work products, as appropriate} are placed under configuration management.

Refer to the Software Configuration Management key process area.

Activity 2 - {Activity performed in Key Process Area X} according to a documented procedure.
This procedure typically specifies that:
1. {Sub-practices for Activity 2, possibly with cross reference(s) to key practice(s) in another key process area.}

Refer to Activity N of the {Z} key process area for practices {related to Activity 2.1}.
----
2. {Additional sub-practices for Activity 2...}
3. {Software work products} undergo peer review {according to appropriate criteria}.

Refer to the Peer Reviews key process area.
----
4. {Software work products, as appropriate} are managed and controlled.

"Managed and controlled" implies that the version of the work product in use at a given time (past or present) is known (i.e., version control), and changes are incorporated in a controlled manner (i.e., change control).
If a greater degree of formality than is implied by "managed and controlled" is desired, the work product can be placed under the full discipline of configuration management, as is described in the Software Configuration Management key process area.

Measurement and analysis

Measurement 1 - Measurements are made and used to determine the status of the activities for {X}.

Examples of measurements include:
- {measurement examples}

Verifying implementation

Verification 1 - The activities for {X} are reviewed with senior management on a periodic basis.
1. {Sub-practices for Verification 1...}

Verification 2 - The activities for {X} are reviewed with the project manager on both a periodic and event-driven basis.

1. {Sub-practices for Verification 1...}

Verification 3 - The software quality assurance group reviews and/or audits the activities and work products for {X} and reports the results.

Refer to the Software Quality Assurance key process area.

At a minimum, these reviews and/or audits verify that:
1. {Sub-practices for Verification 3...}

Documentation for small/prototyping projects

The following thoughts are my personal opinions on interpreting the CMM for small or prototyping projects. The key practices of the CMM are written from the perspective of large, government contracting organizations, but the CMM has been successfully applied across a broad range of environments.

It is fair to say that the practices set expectations for what the normative behavior of organizations will be in satisfying the CMM. I think it is also fair to say that the practices apply to small or prototyping projects, but the expectations for degree of documentation and formality may be quite different from those for a large, government contracting organization. There is a chapter in the TR-25 on "Interpreting the CMM," and this note is basically a restatement of some of that material with examples.

The CMM has a strong emphasis on documenting your process and implementing the process as documented -- say what you do; do what you say. It is our expectation that even small projects or prototyping projects can be expected to satisfy the goals of each key process area that is applicable -- including the documentation requirements. Understanding the appropriate degree of documentation that is critical to consistent, high-quality performance of the process is essential to defining usable processes regardless of the application domain, environment, size, or any of the other drivers that influence interpretation of the CMM.

Some of the issues that must be considered in judging a specific implementation include:

- Mapping of roles. For example, in a small project, the project manager/task leader may fulfill the functions of SCM manager, SCM group, SCCB, project manager, project software manager, software manager, estimator, system engineering group, customer liaison, etc. Is the functionality in the CMM addressed? Who is addressing it?
- Understanding "documented." Documentation may be via SDFs, e-mail, notes, memos, a formal plan/standard/procedure, etc. Documents may be on-line or on paper.
- Formality of the process. For example, an SCCB is an appropriate control mechanism for large projects, but for small projects the approval and review of changes may be the responsibility of the project manager/task leader.
- Granularity, packaging, and frequency. For example, software plans may be packaged in several volumes for a large project; for a small project, plans and requirements may all be gathered together in a single loose-leaf binder. Phrases such as "periodic" and "as appropriate" need to be defined or interpreted to suit the needs of a particular environment.

The challenge in using the CMM intelligently for small/prototyping projects is making a reasonable professional judgement as to the sufficiency of a process. Adding to the challenge is the fact that the criteria bounding these judgements are continuous. How small is "small"? How informal is appropriate for prototyping?

Small projects

What is a small project? Some organizations would say 10-20 people; others 2-3. The general concepts of the CMM apply even at the level of the individual professional, as Watts Humphrey is currently demonstrating with his work on the Personal Software Process.

Consider an organization, perhaps with a thousand people, but projects are small: typically 3-5 people for 1-3 months. Can such an organization satisfy the project management KPAs at Level 2 without incurring excessive overhead?

From a project perspective, a two-week slip is not a big deal. If the project was scheduled to last a month, a two-week slip is still not a major risk. If there are a thousand projects, and each slips its schedule by 50%, that is a big deal for the organization as a whole. It may be appropriate to interpret "project" and "organization" in the CMM as synonyms for this organization and treat the 3-5 person efforts as "tasks" that need to be collectively planned and tracked. Tracking will be almost binary -- is it planned? is it complete? -- since the entire development cycle is near the granularity of a monthly status reporting cycle. The plan for each project/task will be on the order of a one-page task order (which will include the written requirements for what is to be built).

For the small organization, say less than 20 people, a more-of-the-same argument would hold. The process focus might be the responsibility of the president of the company. There might not be an independent SQA group; individuals from other "projects" might act in the (objective) SQA role as part of a peer review system (this could also be true for large organizations that have a "TQM culture"). Policies, standards, and procedures might be combined into a single (short) document, perhaps in a loose-leaf binder.

Prototyping projects

Consider a prototyping project. The entire project may last six months and go through four or five prototyping cycles. The customer does not know exactly what is needed -- that's why they're prototyping. The requirements for the first prototype will be documented by the customer, but neither the requirements nor the plan will be very elaborate. Even the ending criteria may be vaguely stated as "whenever the customer doesn't want to prototype any more." The requirements and plans for successive prototypes may be embedded in the code/design/action items from the last prototype, but they should identify when further refinement of the plan will be performed.

This is certainly not as formally stated as the practices in the CMM would indicate, yet it could be a well-controlled and effective process for managing a prototyping project.

There is a distinction between prototyping and hacking. Hacking has become a pejorative term, denoting an undisciplined, ad hoc approach to software. Prototyping, if properly implemented, is not hacking. There is a range of approaches to developing software: prototyping, rapid prototyping, evolutionary development, incremental delivery, full-scale development, etc. Each has different expectations for degree of rigor and formality that are appropriate in its environment; each can be misapplied in the wrong environment; each can fully satisfy the CMM in an effective manner if intelligently implemented.

Interpreting the CMM

The CMM is aimed at solving systematic problems in the way we manage software projects. Perhaps the best way to tell if a process implementation is sufficient is whether or not you're having significant problems related to the process. If you and your customer disagree over what a product should be doing, you may want to look at Requirements Management; if you're having schedule and cost problems, you may want to think about Software Project Planning and Software Project Tracking & Oversight; if you've lost the product, you might want to consider Software Configuration Management.

If you don't have any project management problems, maybe you're at Level 2 (or higher) -- although I'd make the caveat that managers should ask the people doing the work whether there are problems before concluding there aren't any.

The CMM is not a process description or a process model. It is a description of key process attributes that may be realized in many different ways. Unfortunately, a side effect of making the CMM flexible enough to apply across a broad range of application domains, environments, sizes, etc., is that we create ambiguity that needs to be appropriately interpreted in each context in which the CMM is applied.

[1] I characterize tailoring as involving multiple projects and interpreting as involving a single project, although the documentation might be identical in both cases, based on the use.

QUESTIONS

Return to top of the page

Return to main page

Questions and Answers on the CMM

Mark C. Paulk
Issue #1
5 April 1994

I am frequently asked questions on how to interpret the CMM in various contexts. The following “newsletter” may be of general use. It reproduces the 5 April 1994 mailing to the CMM Correspondence Group. I have to caveat these answers by saying that they represent my opinion and do not represent official SEI positions. They have typically not been internally reviewed, but they do represent a well-informed opinion on what the CMM means.

You are welcome to ask questions on the CMM. I do not guarantee quick answers, although I’ll try to get back to you as soon as possible. Sometimes that’s the same day I receive a question, sometimes it may be a month, depending on my travel schedule and work load. I prefer to receive questions in writing by e-mail. I will sanitize both question and answer before distribution in this newsletter.

Mark Paulk
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
Fax #: (412) 268-5758
Internet: mcp@sei.cmu.edu

The following “questions and answers” are based on discussions held at the CMM workshop on tailoring the CMM held 29-30 March 1994.

Tailoring the CMM

Tailoring the CMM to the specific needs of an organization or class of project is a hot topic right now. These are some of my personal opinions on tailoring; they may or may not be incorporated into the planned SEI report on tailoring guidelines.

As a concept, tailoring lies somewhere between interpreting the CMM and implementing a software process improvement program. I would define interpretation as “a statement, preferably documented, of how an implementation(s) of a process relates to the CMM.” I would define tailoring as “a documented description of how a process should be implemented across multiple[1] projects at about the same level of abstraction as the CMM (i.e., what, but shading into how), with a statement of the relationship to the CMM's concepts and practices." A tailored CMM might have to be interpreted when appraising a specific implementation, although the interpretation should be minimal.

The difference between tailoring and interpreting is based on the need for reliability and consistency. Interpretation implies judgement, which may lead to some degree of unreliability and inconsistency. Tailoring may be cost effective if an organization is concerned about SCEs or if the tailoring is a first step in process definition and improvement that will be applied across projects.

Although the key practices and subpractices describe the normative behavior of large, government contracting organizations, I consider application domain and environment the dominant factors in determining the degree of formality and rigor that is needed by a project. For example, a small commercial project developing life-critical medical software would be expected to follow a formal, rigorous process.

Nonetheless, an organization does not need to implement every practice exactly as stated to fully satisfy a key process area and to achieve its goals. For example, an organization using the Cleanroom approach would not implement some of the testing practices in Software Product Engineering as described.

One reason that tailoring may be needed (at least for consistency and reliability of appraisals) is that a practice or process which is adequate to satisfy the CMM in one context may be inadequate in another context based on the degree of formality and rigor that is needed. For example, I would not normally expect small project SCM to require the same degree of documentation and formality as large project SCM, although I would expect even small projects to achieve the goals of that KPA.

Can key practices be modified? This decomposes into three cases that I can think of:

- Mapping of roles, concepts, and terminology. The terminology in the CMM is hopefully reasonably clear, but the different organizational structures, cultures, process implementations, technologies, methodologies, etc., mean that it is necessary to map the roles in the CMM to job titles/functional responsibilities in the organization, to map process concepts to the specific methodologies and technologies of

[1] I characterize tailoring as involving multiple projects and interpreting as involving a single project, although the documentation might be identical in both cases, based on the use.

  the implemented process(es), and to map key process areas to life cycle activities. Recasting the CMM in the language of an organization can significantly help communication and consistency.
- Alternate implementations. For example, an organization that does Cleanroom has an alternate implementation for some of the testing practices in Software Product Engineering. Whether one views this as an interpretation or tailoring issue, we certainly want to allow for these kinds of legitimate variations, so long as we understand the relationship between the implemented process and the practices in the CMM.
- Specific implementations. For example, requiring the use of COCOMO in planning or OO methods in development. This implies a more specific requirement and might be useful in guiding process definition and improvement. This tailoring would, however, be more specific than the CMM and a violation of the tailoring would not necessarily imply that the equivalent practices in the CMM were not satisfactorily addressed.

The first case is very useful in interpreting the CMM. The last two cases edge into defining the organization's standard software process but may still be in the gray area between tailoring and defining.

A version of the CMM with modified practices could be legitimately claimed as "a tailored version of the CMM." This kind of tailoring was the focus of the CMM Tailoring Workshop held on 29-30 March 1994. There are more extensive adaptations of the CMM that might be desirable, however, and these were raised as issues at the workshop.

Can key process areas be modified? For example, an organization might want to specify inspections for Peer Reviews. Tailoring and process definition have become almost indistinguishable at this point. So long as the modification is more specific than the KPA, satisfying the more specific requirement would satisfy the KPA. Not satisfying a more specific requirement, however, would not necessarily imply that the KPA was not satisfied (although it would indicate that the defined process was not implemented and this is an SQA problem).

I am unable to identify off-hand a reasonable meaning for "less specific variation of a KPA," although I can for a key practice (e.g., for small projects). For example, someone might want to do "Reviews" rather than Peer Reviews and implement them via PDR/CDR kinds of formal review. If such a tailoring was created, it would mean that the CMM KPA (Peer Reviews for the example) was not investigated, and no maturity level could be assigned that depended on the "less specific" KPA.

Modifying key process areas is essentially just a variation of modifying key practices, and I would consider this a tailored version of the CMM, with the caveats on scoring already mentioned. The essential point is a clear understanding of the relationship between the KPA in the CMM (and its goals) and the tailored KPA.

Can key process areas be deleted from the CMM? That may be a legitimate business decision, but it should imply that those KPAs are explicitly identified as Not Investigated, and no maturity level can be assigned which has "deleted" KPAs at or below it.

I would consider a tailoring that deleted KPAs a legitimate one, but care would have to be taken in reporting the score. It is legitimate to claim that an organization is Level 2 that does not do Software Subcontract Management (i.e., it was scored as Not Applicable), but it would not be legitimate to claim an organization was Level 2 if Software Subcontract Management was tailored out of the CMM. This may seem a subtle distinction, but it is necessary to maintain integrity and comparability of appraisal results. Any KPA that is "not investigated" (as distinct from "not applicable") limits the SEI maturity level that can be assigned and should be noted in briefings and reports.

Can key process areas be added to the CMM? For example, an organization might identify a need for KPAs on testing and system engineering. This could be a legitimate approach to process improvement (or contract monitoring) to address specific business or project needs, but there are a couple of provisos:

- The new KPAs should be documented, preferably using the template used in the CMM, and available to everyone who needs them.
- The new KPAs should not be assigned to maturity levels, since they are not part of the maturity framework (e.g., no Level 2 KPA on testing).

Adding KPAs is a particular concern for software capability evaluations. Some contractors are concerned that such tailorings prevent a "level playing field" and that the tailoring will be aimed at selecting a particular contractor. The SEI, however, has no control over how acquisition agencies choose to implement SCEs, although we provide guidelines, training, and recommendations to help acquisition agencies implement SCEs consistently. (Also note that industry can and does evaluate software contractors, and there are similar concerns in that environment.)

The CMM provides a publicly available set of criteria for appraising the software process and guiding an improvement effort. Any tailorings of the CMM that an organization uses should be appropriately documented and communicated to all concerned parties in a timely manner. Any tailoring of the CMM should not be advertised as “the” CMM, but rather as “the tailoring of the CMM for XYZ.”

The relevant point, as Charlie Weber has pointed out, is whether you are tailoring a top-level key practice or a subpractice. To tailor a goal, you should sweat blood (and then be conservative in mapping the relationship to “the” CMM). To tailor a key practice, you should just sweat. To tailor a subpractice, your conscience should hardly bother you.

The CMM is not exhaustive, and it is not a silver bullet. Tailoring the CMM may be appropriate and desirable to address the business needs of an organization. While recognizing that need, we also need to maintain the integrity of the CMM as an asset to the software community.

The key process area template

A template was generally used to express the key process areas using a consistent structure and phrasing. This key process area template, with the “standard” wording used in the various key process areas, follows.

I hope this template will help people understand the CMM and its structure better and help them to use the CMM more efficiently.

There are cases where the practices deliberately deviate from this template. For example, in Verifying Implementation for SQA, the SQA group does not audit itself.

Please note that the numbering of the key practices is not significant, since there are exceptions and additions to the template.

Also note that:

- goals and common features are delimited by underlined headers, e.g., Goals and Commitment to Perform
- key practices are labeled with a noun representing their common feature and a number, e.g., Commitment 1 and Ability 2
- key practices are in bold type, e.g.,

  Ability 1 A group that is responsible for <X> exists.
- subpractices follow their associated key practice in smaller type and are labeled with a number, e.g.,

  1. <A subpractice for Activity 3.>
- practices may be followed by boxed text containing elaboration, cross references, or examples

<Key Process Area X>

The purpose of <Key Process Area X> is <statement>.

<Key Process Area X> involves <summary>.

<Additional elaboration on Key Process Area X as appropriate.>

Goals

Goal 1	**<Process summary statement as goal.>**
Goal 2	**<Process summary statement as goal...>**

Commitment to perform

Commitment 1 **The project follows a written organizational policy for <X>.**

– or –

The organization follows a written policy for <X>.

This policy typically specifies that:

1. <Subpractices for Commitment 1...>

Ability to perform

Ability 1 **A group that is responsible for <X> exists.**

1. <Subpractices for Ability 1...>

Ability 2 **Adequate resources and funding are provided for <X>.**

1. <Subpractices for Ability 2...>

2. Tools to support the activities for <X> are made available.

> Examples of <X> tools include:
>
> - <examples of tools>

Ability 3 **<Roles> are trained <to perform their X activities>.**

– or –

<Roles> receive required training <to perform their X activities>.

> Examples of training include:
>
> - <examples of training>

Ability 4 **<Roles> receive orientation in <X>.**

> Examples of orientation include:
>
> - <examples of orientation training>

Activities performed

Activity 1 **<Activity performed in Key Process Area X.>**

1. <Subpractice for Activity 1, possibly affecting different groups.>

   Examples of affected groups include:

   - <list of affected groups>

2. <Additional subpractices for Activity 1...>

3. <Software work products, as appropriate> are placed under configuration management.

   Refer to the Software Configuration Management key process area.

Activity 2 **<Activity performed in Key Process Area X> according to a documented procedure.**

This procedure typically specifies that:

1. <Subpractices for Activity 2, possibly with cross reference(s) to key practice(s) in another key process area.>

   Refer to Activity N of the <Z> key process area for practices <related to Activity 2.1>.

2. <Additional subpractices for Activity 2...>

3. <Software work products> undergo peer review <according to appropriate criteria>.

> Refer to the Peer Reviews key process area.

4. <Software work products, as appropriate> are managed and controlled.

> "Managed and controlled" implies that the version of the work product in use at a given time (past or present) is known (i.e., version control), and changes are incorporated in a controlled manner (i.e., change control).
>
> If a greater degree of formality than is implied by "managed and controlled" is desired, the work product can be placed under the full discipline of configuration management, as is described in the Software Configuration Management key process area.

Measurement and analysis

Measurement 1 **Measurements are made and used to determine the status of the activities for <X>.**

> Examples of measurements include:
>
> - <measurement examples>

Verifying implementation

Verification 1 **The activities for <X> are reviewed with senior management on a periodic basis.**

1. <Subpractices for Verification 1...>

Verification 2 **The activities for <X> are reviewed with the project manager on both a periodic and event-driven basis.**

1. <Subpractices for Verification 1...>

Verification 3 **The software quality assurance group reviews and/or audits the activities and work products for <X> and reports the results.**

> Refer to the Software Quality Assurance key process area.

At a minimum, these reviews and / or audits verify that:

1. <Subpractices for Verification 3...>

Documentation for small/prototyping projects

The following thoughts are my personal opinions on interpreting the CMM for small or prototyping projects. The key practices of the CMM are written from the perspective of large, government contracting organizations, but the CMM has been successfully applied across a broad range of environments.

It is fair to say that the practices set expectations for what the normative behavior of organizations will be in satisfying the CMM. I think it is also fair to say that the practices apply to small or prototyping projects, but the expectations for degree of documentation and formality may be quite different from those for a large, government contracting organization. There is a chapter in the TR-25 on "Interpreting the CMM," and this note is basically a restatement of some of that material with examples.

The CMM has a strong emphasis on documenting your process and implementing the process as documented – say what you do; do what you say. It is our expectation that even small projects or prototyping projects can be expected to satisfy the goals of each key process area that is applicable – including the documentation requirements. Understanding the appropriate degree of documentation that is critical to consistent, high-quality performance of the process is essential to defining usable processes regardless of the application domain, environment, size, or any of the other drivers that influence interpretation of the CMM.

Some of the issues that must be considered in judging a specific implementation include:

- Mapping of roles. For example, in a small project, the project manager/task leader may fulfill the functions of SCM manager, SCM group, SCCB, project manager, project software manager, software manager, estimater, system engineering group, customer liaison, etc. Is the functionality in the CMM addressed? Who is addressing it?
- Understanding "documented." Documentation may be via SDFs, e-mail, notes, memos, a formal plan/standard/procedure, etc. Documents may be on-line or on paper.
- Formality of the process. For example, an SCCB is an appropriate control mechanism for large projects, but for small projects the approval and review of changes may be the responsibility of the project manager/task leader.
- Granularity, packaging, and frequency. For example, software plans may be packaged in several volumes for a large project; for a small project, plans and requirements may all be gathered together in a single loose-leaf binder. Phrases such as "periodic" and "as appropriate" need to be defined or interpreted to suit the needs of a particular environment.

The challenge in using the CMM intelligently for small/prototyping projects is making a reasonable professional judgement as to the sufficiency of a process. Adding to the challenge is the fact that the criteria bounding these judgements are continuous. How small is "small"? How informal is appropriate for prototyping?

Small projects

What is a small project? Some organizations would say 10-20 people; others 2-3. The general concepts of the CMM apply even at the level of the individual professional, as Watts Humphrey is currently demonstrating with his work on the Personal Software Process.

Consider an organization, perhaps with a thousand people, but projects are small: typically 3-5 people for 1-3 months. Can such an organization satisfy the project management KPAs at Level 2 without incurring excessive overhead?

From a project perspective, a two-week slip is not a big deal. If the project was scheduled to last a month, a two-week slip is still not a major risk. If there are a thousand projects, and each slips its schedule by 50%, that is a big deal for the organization as a whole. It may be appropriate to interpret "project" and "organization" in the CMM as synonyms for this organization and treat the 3-5 person efforts as "tasks" that need to be collectively planned and tracked.

Tracking will be almost binary – is it planned? is it complete? – since the entire development cycle is near the granularity of a monthly status reporting cycle. The plan for each project/task will be on the order of a one-page task order (which will include the written requirements for what is to be built).

For the small organization, say less than 20 people, a more-of-the-same argument would hold. The process focus might be the responsibility of the president of the company. There might not be an independent SQA group; individuals from other "projects" might act in the (objective) SQA role as part of a peer review system (this could also be true for large organizations that have a "TQM culture"). Policies, standards, and procedures might be combined into a single (short) document, perhaps in a loose-leaf binder.

Prototyping projects

Consider a prototyping project. The entire project may last six months and go through four or five prototyping cycles. The customer does not know exactly what is needed – that's why they're prototyping. The requirements for the first prototype will be documented by the customer, but neither the requirements nor the plan will be very elaborate. Even the ending criteria may be vaguely stated as "whenever the customer doesn't want to prototype any more." The requirements and plans for successive prototypes may be embedded in the code/design/action items from the last prototype, but they should identify when further refinement of the plan will be performed.

This is certainly not as formally stated as the practices in the CMM would indicate, yet it could be a well-controlled and effective process for managing a prototyping project.

There is a distinction between prototyping and hacking. Hacking has become a pejorative term, denoting an undisciplined, ad hoc approach to software. Prototyping, if properly implemented, is not hacking. There is a range of approaches to developing software: prototyping, rapid prototyping, evolutionary development, incremental delivery, full-scale development, etc. Each has different expectations for degree of rigor and formality that are appropriate in its environment; each can be misapplied in the wrong environment; each can fully satisfy the CMM in an effective manner if intelligently implemented.

Interpreting the CMM

The CMM is aimed at solving systematic problems in the way we manage software projects. Perhaps the best way to tell if a process implementation is sufficient is whether or not you're having significant problems related to the process. If you and your customer disagree over what a product should be doing, you may want to look at Requirements Management; if you're having schedule and cost problems, you may want to think about Software Project Planning and Software Project Tracking & Oversight; if you've lost the product, you might want to consider Software Configuration Management.

If you don't have any project management problems, maybe you're at Level 2 (or higher) – although I'd make the caveat that managers should ask the people doing the work whether there are problems before concluding there aren't any.

The CMM is not a process description or a process model. It is a description of key process attributes that may be realized in many different ways. Unfortunately, a side effect of making the CMM flexible enough to apply across a broad range of application domains, environments, sizes, etc., is that we create ambiguity that needs to be appropriately interpreted in each context in which the CMM is applied.

Questions and Answers about the Software Capability Maturity Model® (SW-CMM® (Q&A #2)

Mark C. Paulk
Issue #2 - 2 August 1994

I am frequently asked questions on how to interpret the CMM in various contexts. The following "newsletter" may be of general use. It reproduces the 2 August 1994 mailing to the CMM Correspondence Group. I have to caveat these answers by saying that they represent my opinion and do not represent official SEI positions. They have typically not been internally reviewed, but they do represent a well-informed opinion on what the CMM means.

General Questions and Answers on the CMM

Q. In our thinking

- A policy specifies what will happen.
- A procedure specifies how it will happen.
- A standard specifies the quality of what will happen.

As you developed the KPA of the CMM document were there rules such as these for interpretation of these terms:

A. Our general guidelines were similar:

- policy sets the cultural expectations of the organization
  - "that's how we do things around here"
- procedure is process-oriented (how)
  - a set of steps for doing something
- standard is product-oriented (both functional and quality)
  - what the resulting work product will look like

Requirements Management

Q. In the CMM, each KPA has a measurements and analysis section. However, unlike the other KPAs, under Requirements Management, there does not appear to be an example measurement of "work completed, effort expended in Requirements Management compared to the plan." Understanding that all of KPA measurements are only "examples" was this merely an omission, or is there some other reason why requirements costs are not relevant?

A. It was merely an omission. A complicating factor may be that much of the work in

RM may be done in proposal phase and early in the life cycle. Depending on your project management process, you may only have "draft plans" in place early in the life cycle, so you have some complications in tracking against the plan. This is a particular case, however, of tracking in a rework/revised plan environment. Also, much of this work may be done by Systems Engineering, so the SDP part may (or may not) be separately managed from the Project Management Plan part - depending on the degree of concurrency/integration between your engineering groups.

Q. As I understand, the CMM is directed to mission-critical, development efforts. For (new) development projects, one can determine a baseline of software requirements. In measuring the "change activity" for allocated requirements, this would suggest to me evaluating the volatility of requirements by measuring the number of changes to the baseline. In your experience, is there value to comparing the number of changes to the number of requirements in the baseline? This could be expressed as a % of change, as opposed to a count only.

A. In a development environment, yes, because it helps you understand the relationship of volatility to eventual delivery. For maintenance, you may be more interested in trends as related to block updates.

Q. In our environment, most of our work is in the arena of MAINTENANCE, and as such we do not have the baseline requirements for the original production software on legacy systems. What we could do is as follows: Using the medium of the System Service Requests (SSRs) (a form to report problems, request enhancements to production systems, etc.), we could define, itemize and enumerate the requirements related to each SSR, thereby determining a maintenance baseline. If changes ensue during the course of working to satisfy these requirements, then one could monitor change to this baseline. (Note that (1) costs are tracked via estimates and actuals to SSRs and (2) time cards have the capability of tracking time spent in Requirements Management activities.) Over time, one could (1) collect costs to manage requirements and (2) determine % of requirements changed over the entire SSR implementation and/or the phases/sub-phases. Both of these would be indicators of how much time to estimate for requirements management relative to number of requirements and how much time should be expected to accommodate changes to requirements.

A. I would assume that you do block updates and have multiple releases based on different maintenance baselines per update. SSRs would be tracked to a particular baseline ... but it looks like you want/need to track changes to SSR changes, so your SSRs may be the equivalent of a functional release (operational increment, etc.). In that case, I'd treat the SSR as a baseline and measure volatility in terms of change requests to that baseline.

Q. What experience if any, have you had with enterprises dealing with maintenance software vs. development software, with regard to requirements management?

A. Look at some of the experience papers coming out of the IBM Houston (now Loral Space Information Systems) Space Shuttle software project. They're in an environment that would seem to be somewhat similar to what you're dealing with. Also telecom companies have a similar set of problems and have published on their update/release process. Check some of the recent international conference proceedings for up-to-date citations.

Q. The first question arose out of an assessment of a set of maintenance projects involving legacy systems. The legacy systems are poorly documented in general, and lack documentation of the original requirements, in particular. The organization handles software problem reports (SPRs) and baseline change requests (BCRs), which are documented explicitly to ensure clarity, completeness, feasibility and testability. Testing of the changes is made somewhat awkward because testers lack documentation of requirements for the entire system (as opposed to just the specific SPRs and BCRs, whose changes they can verify directly). The question is whether the organization meets the spirit and letter of the Requirements Management KPA if they do not back-document all the system's requirements (an activity that might need several man-years' effort to complete). Does the SEI have an official position on this situation? Your opinion?

A. I can't give you an "official" position, but I'm happy to give you my opinion.

Requiring extensive back documentation would be unreasonable, although it might be desirable to maintain an architectural view of what's going on in the system. In this environment there would seem to be two critical needs:

- a good regression test suite, to make sure you aren't accidentally blowing away functionality that was there, but which may be undocumented (this may just be an add-on to the undocumented requirements problem, but it's certainly another aspect to worry about), although I suppose you could use the "if it goes away and nobody complains, we really didn't need it anyway" argument.
- good documentation and tracking of changes and change requests/problem reports, which you seem to have

The second need would address the spirit of RM, although I think you'd agree that there's a noticeable vulnerability here. It's characteristic of many legacy systems, however, and we're just going to have to live with them (at least until the year 2000 when most of them will turn belly up on two-digit years).

I would classify this KPA as satisfactorily addressed, but document a finding that there is a risk here that needs to be tracked over time. If the legacy system is to be kept, and it's critical, it might be worthwhile to consider re-engineering it.

Software Project Planning

Q. A question came up concerning the interpretation of the Project Planning KPA practice that says "Size, cost, and schedule estimates must have a BASIS IN REALITY" (in SEI training materials). When contractors said they use informal personal experience that is undocumented or tracked, many team members said that their estimates satisfied this criteria since developers were basing their estimates on their personal experience. I however said that this is not sufficient for a Defined or Repeatable process. I was looking for historical logs or data bases, estimation procedures, and a formal comparison of actuals vs. planned to improve the estimation process. What does "basis in reality" mean?

A. PP.AC.9.3 says "historical data are used where available." PP.AC.9.4 and 9.5 go on to talk about documenting assumptions and estimates. There is no explicit requirement that the historical data be documented initially, and initially I would doubt that it would be. One purpose of documenting, however, is to institutionalize corporate knowledge. If the organization is documenting and plans to use this historical data more formally in the future, I would have no concerns. If they aren't, then I would be concerned they're looking at the letter of the law rather than the spirit. Personal experience is a beginning of "a realistic basis" and the major concern of realism could be (at least partially) addressed by estimating procedures that prevented/controlled a proposal manager who said "we have to cut this by 25% to win the contract."

In a similar vein, Measurement & Analysis is the beginning of improvement for the planning process, but note that we do not close the loop (in the PDCA cycle sense, where this is the Check aspect) at Level 2. Improvement of the planning process is not required at Level 2, although we do expect to see the infrastructure being put into place (i.e., M&A).

Level 3 is supposed to be using the organization's software process database, which contains this kind of historical data, so I infer the question applies only to the Level 2 instantiation.

Q. Humphrey's Book: To clarify the above criteria concerning "basis of reality" I wanted the team to refer to Humphrey's book for clarification. But some team members said that Humphrey's book is NOT a reference book for appraisal teams and should not be used to clarify the intent of some of the bullets presented in the training . Should teams refer to Humphrey's book for clarification of KPAs?

A. I would infer that this is a very audit oriented team. To use the CMM correctly, one must apply professional judgement. (As many of the team members seemed willing to do in the case of personal experience.) Any tool, whether it's Humphrey's book or someone else's, that helps the team understand the technical and management issues underlying a point of discussion is of value. No book, and I include the CMM here, should be used to replace thinking and consideration on the part of the team. Appropriate implementations of the CMM practices must be judged in a thoughtful and professional way, and the team should come to consensus on issues. If there are strong minority opinions on the team, I would expect those to come out as risks in the teams report, even though the "score" might be satisfactory in that area (I believe you should be able to have "fully satisfied with findings" for any KPA).

The issue that's being struggled with is whether this is a "good enough" practice. I believe the CMM provides significant guidance for making these kinds of judgement, but there are times when flexibility/ambiguity clouds an issue. Do not remove the possibility of reporting a concern, even if the conservative decision is that the practice is okay.

There's another underlying issue here, which is the training as it relates to the CMM. My perhaps biased judgement is that the CMM overrides any particular set of training materials, which of necessity has to summarize and abstract the concerns of the CMM. In its turn, the CMM summarizes and abstracts the software engineering and management practices that a team must judge - professionally - when doing an assessment or evaluation. The CMM is not, and cannot be, an absolute objective set of criteria that applies equally in all environments.

Software Quality Assurance

Q. I am working to tailor the CMM for small-medium IS organizations. The team has agreed that the CMM, with [minimal] tailoring of terminology, does apply to IS organizations. One area of concern though is Software Quality Assurance. I think everyone agrees with the intent behind the SQA KPA, but there is an issue with the "objectivity" of the SQA group. [It is understood that there does not have to be a dedicated "group" of people to do these activities.] IS projects/organizations are typically small (some as small as 1 or 2 people, for short durations [less than 3 weeks]). It seems that a common practice in this company's IS organizations is for project managers to perform the role of SQA. Herein lies the gotcha with the SQA KPA - the objectivity of the SQA group. I consider this implementation of SQA to be in violation of the intent of the KPA. Can you provide any insight into how an organization (with many small projects) can implement SQA without incurring too much cost, but also comply with the intent of the CMM?

A. I agree that this would be in violation of the intent. Although SQA acts as management's window on the project, the manager may not be objective either - and is usually juggling several priorities. That's the reason for bucking noncompliance issues up to senior management.

I might assign someone outside the project just to run through the SQA function on a part-time basis. Without an SQA group, the risk you run is a "you scratch my back, I'll scratch yours" situation developing. Clearly, organization culture will be a major determinant of whether an ad hoc SQA function is viable. I would have to know the organization before I could determine whether this would work or not.

What seems to have been very successful in small projects is setting up a culture where using the standards and procedures is "just the way we do things around here," reinforcing the culture with process definitions that are clearly stated (and trained where possible, maybe through mentor), and using peer pressure as the enforcement mechanism. Then an ad hoc function with part time SQA people can be pretty effective. You can even embed it into your peer review process.

Q. Do you feel there is a lower bound to the size (head count) and duration of projects that is outside the domain of CMM applicability?

A. In terms of the goals, no. In terms of implementation, definitely. I've worked in the two-person project environment before, and our SQA function was definitely driven by peer pressure - and professional pride. Ultimately that's what is need in any organization to make Quality work.

Added comment from Judah Mogilensky:
Let me add just a couple of comments to what Mark has already said.

The implicit assumption of the CMM's SQA KPA, founded upon a great deal of experience with Level 1 and Level 2 organizations, is that the Software Project Manager role is more concerned with schedule and budget than with process integrity. I might be willing to entertain an argument that, in a particular organization, the Software Project Manager is expected to sacrifice budget and schedule considerations in favor of process integrity, and that person is rewarded by the organization when he/she does so, and therefore that person is "objective" about compliance with policies, procedures, and standards. But I would be very skeptical, and I'd need a lot of convincing. And I would not consider having the Software Project Manager do SQA acceptable under any other circumstances.

If an organization is one in which most work is done by very small teams over very short durations, then I would think that kind of work would be very important to the organization. I would think that the quality of the products, and the predictability of the projects, would be crucial to the health of the organization. And I would think that having standard approaches to planning and tracking such short, small projects would be a top priority, to save projects from wasting time inventing their own or not following proven successful approaches. And therefore someone should be spending at least part of their time verifying that these small projects are being managed according to organizational standards, someone other than the Project Manager of each project.

Someone who is in a position to raise a flag with the organization's management when a project is not being managed according to organizational standards. This, in a nutshell, is what the SQA KPA calls for.

As Mark knows, I am a strong advocate of the idea that organizations can evolve to the point where process compliance is part of the culture, where failure to comply with process is viewed as "socially aberrant behavior," and therefore is exceptionally rare or totally non-existent. In such a case, I believe that the formal organizational mechanisms of SQA may no longer be needed, like modern sutures that are absorbed into the skin and disappear when no longer needed to hold an incision together. However, to make this case, the organization needs to show both what mechanisms they have for sustaining this culture and indoctrinating new individuals, and they need to show that process compliance can be verified, and has been sustained at very high levels.

Small teams and short duration projects are not exemptions from the concept of process maturity (although they can typically make effective use of simpler, lighter forms). Watts Humphrey's next book advocates a Personal Software Process, where beginning programming students working alone on toy problems learn to apply a personal process from the start, and collect data, so that they can scale up their process as their software development skills grow and project scope expands.

End of Judah's Comment

Q. Do you know what Activity 5 in Software Quality Assurance is looking for? "SQA group audits" - Is this looking for an audit of every designated work product or could it be a spot audit?

A. Designation can cover a variety of possibilities. If the Software Requirements Spec is simply designated to be audited, then it should be fully audited. However, the designation could be to spot-audit by some criteria that are specified as part of the designation. The key point is to decide in advance what the criteria are for auditing (random sampling, 100%, these products only, etc.) lest schedule pressures "crunch" the SQA process.

Q. "products are evaluated." Is this an evaluation by the QA group or could it be an evaluation by another group which the QA group verifies happens?

Either is valid. For example, evaluation could include the conduct of system testing performed by a testing group and the SQA group verifies that the system tests were performed according to the specified test process/criteria.

Q. "customer." There are internal and external customers, in fact every task produces a product which is then delivered to a customer, what sort of customer is meant here?

A. Both. For example, SRS may be evaluated via peer review before handover to the design "customers" and SQA reviews conduct of the peer review. Previous example was handover to external customer.

Q. "designated." Can designated be something that QA and the project agree to or is there some other criteria that should be used to determine designated.

A. Depends on what the corporate standards/procedures and project tailorings are. If the corporate standard cannot be tailored to remove some criteria, then project cannot prevent QA from dinging them. If project has validly tailored something, with the concurrence of the affected parties - as specified in the tailoring guidelines (I'm hitting this from a Level 3 perspective, sorry about that), then that's what QA audits.

Short answer: if organization has criteria, use them; if not, affected parties should agree on what designated means - early in process rather than when crunch is on.

Q. Can you or the SEI provide any clarification or guidance to CMM users for SQA Commitment 1 on page L2-61 of 93-TR-25. We are concerned about whether the recommendation about "performance appraisal by the management of the software project they are reviewing." We found this recommendation in two places, but one of them is bracketed supplementary information which can be construed as either guidance or elaboration in this context. (The quotes follow below.) Could the appraisals be done by a manager in the same project, but above or independent of the software manager?

A. You're talking about specific implementations here. This concept maps up to Goal 2, which states that "Adherence of software products and activities to the applicable standards, procedures, and requirements is verified objectively" If an organization can demonstrate objective verification, then independence is not required. That demonstration is the responsibility of the organization however; an SCE team, for example, might have a different judgement, depending on how convincing the argument was.

To answer the questions specifically as asked, however, no, the appraisal would have to be done outside the project UNLESS you can DEMONSTRATE that there is an effective escalation mechanism in place that is not affected by the fact that the person (potentially) halting the project receives raises and promotions from the individual responsible for meeting schedules. That may not be easy to do. A close examination of noncompliances and their disposition would be in order.

Q. Does the appraisal have to be done by a manager to whom the program manager of the whole project (including software) reports?

A. No. Could be matrixed in from a separate QA organization, for example.

Q. Is there are general issue here that we are missing (indicated by the appearance of the requirement in two places?

Well, you're quoting from the Overview and the Key Practices. Realizing that there would be interpretation concerns about organizational structure, section 4.4.3 was written to try and clarify potential concerns and issues in thinking about different implementations. In going back and looking at it, I think I've just re-stated what it said. Similarly for the supplementary boxes.

Q. We recognize how and why this independence is useful, so we have several solutions in mind, but we are also concerned that if our customers (users of the CMM and SCE) have a different interpretation, then our solution may not be satisfactory. Any clarification you may provide will be gratefully received.

A. I infer that your real concern is whether an SCE team would be capable of judging whether a perfectly adequate implementation was satisfactory or not. This is a larger question than specifically SQA, although that may be a particularly sensitive point. Hopefully, the Intro to CMM course will alleviate some of these concerns when we have SCE teams fully trained.

Q. What are the intended meanings of the terms "deviation" and "noncompliance item/issue" as used in the CMM? It seems in the CMM these two terms do not mean the same thing since both are reported to a project but only noncompliance items are escalated to senior management. A brief survey here of CMM experience people came up with 4 different sets of definitions.

A. The CMM defines

> deviation - A noticeable or marked departure from the appropriate norm, plan, standard, procedure, or variable being reviewed.

in the glossary. Noncompliance item is not defined in the glossary; it is discussed only under the key practice QA.AC.7 on documenting and handling deviations.

Frankly, we didn't intend any subtle distinction. As QA.AC.7.2 suggests (but does not explicitly state), a deviation may result in a noncompliance item. It's like the difference between a failure and a problem report. Deviations occur, even if they're never identified, documented, and tracked to closure. The noncompliance item is a form of documented deviation (although there may be other forms as is perhaps suggested by QA.AC.7.1).

When to officially classify a deviation as a noncompliance item is fuzzy, and I infer that's the potential source of confusion. There is an implication that if a deviation is handled at the project level, it doesn't (necessarily) enter the reporting chain as a noncompliance item. We did not intend to say that this is the right way to track deviations/noncompliance items. It MAY be useful to separate deviations into items handled at the project level versus those that need to be escalated; it MAY be useful to not even track deviations handled at the project level.

All that we intended in the CMM was to delimit the end points: deviations occur; noncompliance reports are escalated to senior management as necessary. The middle territory is left "flexible" but our terminology may have made it actively ambiguous. If so, this may be an error in the CMM that should have a change request written for correction in the next release of the CMM. If you feel this is something that we need to do, please send in a CR.

H3>Organization Process Definition

Q. OPD Activity 1 - The organization's standard software process is developed and maintained according to a documented procedure. Does this paragraph imply that the "procedure" is a statement that indicates or acknowledges the organization's policies and standards will be satisfied and that state-of the practice tools and methods will be used?

A. Both of these are motherhood kinds of statements. For the first, if the policies, procedures, and standards are inconsistent or incompatible, it seems clear that you have a problem. For the second, we don't say anywhere in the CMM that you have to use state-of-the-practice tools and methods - just that you use appropriate tools for what you're doing. There may be reasons for not using state-of-the-practice tools (some reports indicate most organizations don't use structured programming yet, which is 70s vintage), but the efficiency of your process in a competitive situation may be so low that you can't survive.

Q. OPD Activity 2 - The organization's standard software process is documented according to established standards. Do established standards simply specify that elements are decomposed "to the granularity needed to understand and describe the process", Do established standards simply state that "Each process element is described and addresses: . . .", and that "relationships of the process elements are described and address: . . . "? Or do established standards provide how to document the process.

These standards would tell you, for example, what a process element looks like. Compare it to a design standard. Do design standards discuss granularity of the design? Do they tell you how to document a design? Do they tell you how to describe a design? Do they tell you how parts of the design can/should relate to one another? I think the answers to ALL these questions are yes, and the equivalent would hold true for process standards.

Q. Do you have a sample or a template of a standard?

A. Some references:

Bill Curtis, Marc I. Kellner, and Jim Over, "Process Modeling," Communications of the ACM, Vol. 35, No. 9, September 1992, pp. 75-90.

David Harel, "On Visual Formalisms," Communications of the ACM , Vol.31, No. 5, May 1988, pp. 514-530.

David Harel, H. Lachover, A. Naamad, A. Pnueli, M. Politi, R. Sherman, and A. Shtul-Trauring, "STATEMATE: A Working Environment for the Development of Complex Reactive Systems," Proceeding of the 10th IEEE International Conference on Software Engineering, Singapore, IEEE Press, 13-15 April 1988.

G.F. Hoffnagel and W. Bergei, "Automating the Software Development Process," IBM Systems Journal, Vol. 24, No. 2, 1985, pp. 102-120.

Watts S. Humphrey and Marc I. Kellner, "Software Process Modeling: Principles of Entity Process Models," Proceedings of the Eleventh International Conference on Software Engineering, IEEE Computer Society Press, 1989, pp. 331-342.

Marc I. Kellner and Gregory A. Hansen, "Software Process Modeling: A Case Study," Proceedings of the Twenty-Second Annual Hawaii International Conference on Systems Sciences, Vol. II - Software Track, IEEE Press, 1989, pp. 175-188.

Marc I. Kellner, "Software Process Modeling: Value and Experience," SEI Annual Technical Review, Carnegie Mellon University, Pittsburgh, PA, 1989, pp. 23-54.

Marc I. Kellner, "Software Process Modeling Support for Management Planning and Control," Proceedings of the First International Conference on Software Process, IEEE Computer Society Press, 1991, pp. 8-28.

R.A. Radice, N. K. Roth, A. C. O'Hara, Jr., and W. A. Ciarfella, "A Programming Process Architecture," IBM Systems Journal, vol. 24, no. 2, 1985.

Ronald A. Radice and Richard W. Phillips, Software Engineering: An Industrial

Approach, Volume 1, Simon & Schuster, Englewood Cliffs, NJ, 1988.

Training Program

Q. The contractor had a terrific training plan but did not implement it because "the customer would not pay for it." None the less, the SCE team assessed them as inadequate since there was no implementation. Should the customer pay for training?

A. If the customer has an interest in an on-going customer/supplier relationship, then it behooves the customer to be proactive in supporting the development of skills pertinent the area of the relationship.

If the customer refuses to pay for training, however, the contractor should bite the bullet and train their people out of their own pocket. This is a basic cost of quality issue. If you believe that:

- people who are happy about their continuing professional development are more likely to stay with a company
- high turnover leads to both low productivity and low quality, with a corresponding impact of high cost
- training is value-added because it increases the skill of the people doing the work then I don't see how a contractor can legitimately claim that "the customer won't pay for training" makes training Not Applicable. It just doesn't hold for me.

The flip side, of course, is that:

- training costs money
- people being trained are not working on a product
- key people may be needed on the project at critical times

It's long-term vs short-term views that we're looking at. If people are viewed as a resource to be used up and discarded when the contract is over, then I'm not sure that contractor is one that I would want to establish a long-term customer/supplier relationship with.

Of course there's also the possibility that the "terrific training plan" was really pretty lousy and not worth implementing if the customer wouldn't pay for it :-)

A. The question concerns the apparent absence in the Software Project Planning KPA of the need to define project-specific training in the project plans. Is this an oversight or an omission that will be corrected in the next version of the CMM? I would think, as a matter of good project planning, that plans should specify the training that team members will need to receive, the timing of same, the potential impacts to schedule of off-site training, the cost of training billed to the project, etc. But the CMM seems to omit these notions.

A. Project training plans are specifically mentioned in Level 3 in Training Program, but you're right that they aren't mentioned at Level 2. It is certainly an omission that should be considered. If anyone thinks it's important enough to write a change request (hint, hint), then it would probably be added in the next version of the CMM.

Integrated Software Management

Q. ISM Activity 1 - The project's defined software process is developed by tailoring the organization's standard software process according to a documented procedure.. How detailed is a "documented procedure?"

A. Detailed enough to be useful; not too detailed to be usable.

Q. Do you have a sample or a template of this procedure?

A. The bottom line is that the CMM leaves a lot of flexibility in terms of level of detail, etc., that's needed to document processes. There are a number of factors that influence how you document processes, including:

- size of organization
- organizational culture
- size of project
- determinism of process
- process description tools
- degree of automated process support
- technical sophistication of management
- budget available for process definition
- degree of organizational support for process definition
- etc.

There are no simple answers to this question.

Peer Reviews

Q. The contractor had great procedures for walkthroughs in their "generic SDP". But EVERY project tailored out all references to requirements, code, and test case walkthroughs and did not use any forms for recording preparation effort or summarizing findings (they did however have adequate Action Item Logs). The project level SDPs which tailored out these walkthrough procedures were approved by the company's Software Review Board which oversees implementation of the "procedures in the generic SDP." But the team assessed this KPA as inadequate. Should a KPA be acceptable if a company has good walkthrough procedures but tailors (most) all of them out with the approval of a higher review authority?

A. I consider this a judgement call on the part of the SCE team, in light of the needs of the acquisition. As described, this is a case of trying to score well, rather than implementing a process. A company-level process should be applied to the majority of cases (although when there are alternatives, picking one of the alternatives is the reasonable interpretation of majority rule). Consistently tailoring out practices violates the intent of the CMM.

The judgement might be that Peer Reviews is satisfied (if the procedures are good), but Software Product Engineering is not, since SPE calls out peer reviews for requirements (PE.AC.2.8), code (PE.AC.4.4), and test cases (PE.AC.5.6). I could infer from your description that adequate peer reviews are implemented in some areas, such as design. The question, if I wanted to probe more deeply, would then be what alternative quality control mechanisms is used for these particular cases.

QUESTIONS

top | CMM main page

Questions and Answers on the CMM

Mark C. Paulk
Issue #2
2 August 1994

I am frequently asked questions on how to interpret the CMM in various contexts. The following “newsletter” may be of general use. It reproduces the 2 August 1994 mailing to the CMM Correspondence Group. I have to caveat these answers by saying that they represent my opinion and do not represent official SEI positions. They have typically not been internally reviewed, but they do represent a well-informed opinion on what the CMM means.

You are welcome to ask questions on the CMM. I do not guarantee quick answers, although I’ll try to get back to you as soon as possible. Sometimes that’s the same day I receive a question, sometimes it may be a month, depending on my travel schedule and work load. I prefer to receive questions in writing by e-mail. I will lightly edit and sanitize question and answer before distribution in this newsletter.

Mark Paulk
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
Fax #: (412) 268-5758
Internet: mcp@sei.cmu.edu

General Questions and Answers on the CMM

In our thinking

> *A policy specifies what will happen,*
> *A procedure specifies how it will happen, and*
> *A standard specifies the quality of what will happen.*

As you developed the KPA of the CMM document were there rules such as these for interpretation of these terms:

Our general guidelines were similar:

- policy sets the cultural expectations of the organization
  - “that's how we do things around here”
- procedure is process-oriented (how)
  - a set of steps for doing something
- standard is product-oriented (both functional and quality)
  - what the resulting work product will look like

Requirements Management

In the CMM, each KPA has a measurements and analysis section. However, unlike the other KPAs, under Requirements Management, there does not appear to be an example measurement of "work completed, effort expended in Requirements Management compared to the plan." Understanding that all of KPA measurements are only "examples" was this merely an omission, or is there some other reason why requirements costs are not relevant?

It was merely an omission. A complicating factor may be that much of the work in RM may be done in proposal phase and early in the life cycle. Depending on your project management process, you may only have "draft plans" in place early in the life cycle, so you have some complications in tracking against the plan. This is a particular case, however, of tracking in a rework / revised plan environment. Also, much of this work may be done by Systems Engineering, so the SDP part may (or may not) be separately managed from the Project Management Plan part - depending on the degree of concurrency / integration between your engineering groups.

As I understand, the CMM is directed to mission-critical, development efforts. For (new) development projects, one can determine a baseline of software requirements. In measuring the "change activity" for allocated requirements, this would suggest to me evaluating the volatility of requirements by measuring the number of changes to the baseline. In your experience, is there value to comparing the number of changes to the number of requirements in the baseline? This could be expressed as a % of change, as opposed to a count only.

In a development environment, yes, because it helps you understand the relationship of volatility to eventual delivery. For maintenance, you may be more interested in trends as related to block updates.

In our environment, most of our work is in the arena of MAINTENANCE, and as such we do not have the baseline requirements for the original production software on legacy systems. What we could do is as follows: Using the medium of the System Service Requests (SSRs) (a form to report problems, request enhancements to production systems, etc.), we could define, itemize and enumerate the requirements related to each SSR, thereby determining a maintenance baseline. If changes ensue during the course of working to satisfy these requirements, then one could monitor change to this baseline. (Note that (1) costs are tracked via estimates and actuals to SSRs and (2) time cards have the capability of tracking time spent in Requirements Management activities.) Over time, one could (1) collect costs to manage requirements and (2) determine % of requirements changed over the entire SSR implementation and/or the phases/sub-phases. Both of these would be indicators of how much time to estimate for requirements *management relative to number of requirements and how much time should be expected to accommodate changes to requirements.*

I would assume that you do block updates and have multiple releases based on different maintenance baselines per update. SSRs would be tracked to a particular baseline ... but it looks like you want/need to track changes to SSR changes, so your SSRs may be the equivalent of a functional release (operational increment, etc.). In that case, I'd treat the SSR as a baseline and measure volatility in terms of change requests to that baseline.

What experience if any, have you had with enterprises dealing with maintenance software vs. development software, with regard to requirements management?

Look at some of the experience papers coming out of the IBM Houston (now Loral Space Information Systems) Space Shuttle software project. They're in an environment that would seem to be somewhat similar to what you're dealing with. Also telecom companies have a similar set of problems and have published on their update/release process. Check some of the recent international conference proceedings for up-to-date citations.

The first question arose out of an assessment of a set of maintenance projects involving legacy systems. The legacy systems are poorly documented in general, and lack documentation of the original requirements, in particular. The organization handles software problem reports (SPRs) and baseline change requests (BCRs), which are documented explicitly to ensure clarity, completeness, feasibility and testability. Testing of the changes is made somewhat awkward because testers lack documentation of requirements for the entire system (as opposed to just the specific SPRs and BCRs, whose changes they can verify directly). The question is whether the organization meets the spirit and letter of the Requirements Management KPA if they do not back-document all the system's requirements (an activity that might need several man-years' effort to complete). Does the SEI have an official position on this situation? Your opinion?

I can't give you an "official" position, but I'm happy to give you my opinion.

Requiring extensive back documentation would be unreasonable, although it might be desirable to maintain an architectural view of what's going on in the system. In this environment there would seem to be two critical needs:

* a good regression test suite, to make sure you aren't accidentally blowing away functionality that was there, but which may be undocumented (this may just be an add-on to the undocumented requirements problem, but it's certainly another aspect to worry about), although I suppose you could use the "if it goes away and nobody complains, we really didn't need it anyway" argument
* good documentation and tracking of changes and change requests/problem reports, which you seem to have

The second need would address the spirit of RM, although I think you'd agree that there's a noticeable vulnerability here. It's characteristic of many legacy systems, however, and we're just going to have to live with them (at least until the year 2000 when most of them will turn belly up on two-digit years).

I would classify this KPA as satisfactorily addressed, but document a finding that there is a risk here that needs to be tracked over time. If the legacy system is to be kept, and it's critical, it might be worthwhile to consider re-engineering it.

Software Project Planning

A question came up concerning the interpretation of the Project Planning KPA practice that says "Size, cost, and schedule estimates must have a BASIS IN REALITY" (in SEI training materials). When contractors said they use informal personal experience that is undocumented or tracked, many team members said that their estimates satisfied this criteria since developers were basing their estimates on their personal experience. I however said that this is not sufficient for a Defined or Repeatable process. I was looking for historical logs or data bases, estimation procedures, and a formal comparison of actuals vs. planned to improve the estimation process. What does "basis in reality" mean?

PP.AC.9.3 says "historical data are used where available." PP.AC.9.4 and 9.5 go on to talk about documenting assumptions and estimates. There is no explicit requirement that the historical data be documented initially, and initially I would doubt that it would be. One purpose of documenting, however, is to institutionalize corporate knowledge. If the organization is documenting and plans to use this historical data more formally in the future, I would have no concerns. If they aren't, then I would be concerned they're looking at the letter of the law rather than the spirit. Personal experience is a beginning of "a realistic basis" and the major concern of realism could be (at least partially) addressed by estimating procedures that prevented/controlled a proposal manager who said "we have to cut this by 25% to win the contract."

In a similar vein, Measurement & Analysis is the beginning of improvement for the planning process, but note that we do not close the loop (in the PDCA cycle sense, where this is the Check aspect) at Level 2. Improvement of the planning process is not required at Level 2, although we do expect to see the infrastructure being put into place (i.e., M&A).

Level 3 is supposed to be using the organization's software process database, which contains this kind of historical data, so I infer the question applies only to the Level 2 instantiation.

- Humphrey's Book: To clarify the above criteria concerning "basis of reality" I wanted the team to refer to Humphrey's book for clarification. But some team members said that Humphrey's book is NOT a reference book for appraisal teams and should not be used to clarify the intent of some of the bullets presented in the training . Should teams refer to Humphrey's book for clarification of KPAs?

I would infer that this is a very audit oriented team. To use the CMM correctly, one must apply professional judgement. (As many of the team members seemed willing to do in the case of personal experience.) Any tool, whether it's Humphrey's book or someone else's, that helps the team understand the technical and management issues underlying a point of discussion is of value. No book, and I include the CMM here, should be used to replace thinking and consideration on the part of the team. Appropriate implementations of the CMM practices must be judged in a thoughtful and professional way, and the team should come to consensus on issues. If there are strong minority opinions on the team, I would expect those to come out as risks in the teams report, even though the "score" might be satisfactory in that area (I believe you should be able to have "fully satisfied with findings" for any KPA).

The issue that's being struggled with is whether this is a "good enough" practice. I believe the CMM provides significant guidance for making these kinds of judgement, but there are times when flexibility/ambiguity clouds an issue. Do not remove the possibility of reporting a concern, even if the conservative decision is that the practice is okay.

There's another underlying issue here, which is the training as it relates to the CMM. My perhaps biased judgement is that the CMM overrides any particular set of training materials, which of necessity has to summarize and abstract the concerns of the CMM. In its turn, the CMM summarizes and abstracts the software engineering and management practices that a team must judge - professionally - when doing an assessment or evaluation. The CMM is not, and cannot be, an absolute objective set of criteria that applies equally in all environments.

Software Quality Assurance

I am working to tailor the CMM for small-medium IS organizations. The team has agreed that the CMM, with [minimal] tailoring of terminology, does apply to IS organizations. One area of concern though is Software Quality Assurance. I think everyone agrees with the intent behind the SQA KPA, but there is an issue with the "objectivity" of the SQA group. [It is understood that there does not have to be a dedicated "group" of people to do these activities.] IS projects/organizations are typically small (some as small as 1 or 2 people, for short durations [< 3 weeks]). It seems that a common practice in this *company's IS organizations is for project managers to perform the role of SQA. Herein lies the gotcha with the SQA KPA - the objectivity of the SQA group. I consider this implementation of SQA to be in violation of the intent of the KPA. Can you provide any insight into how an organization (with many small projects) can implement SQA without incurring too much cost, but also comply with the intent of the CMM?*

I agree that this would be in violation of the intent. Although SQA acts as management's window on the project, the manager may not be objective either - and is usually juggling several priorities. That's the reason for bucking noncompliance issues up to senior management.

I might assign someone outside the project just to run through the SQA function on a part-time basis. Without an SQA group, the risk you run is a "you scratch my back, I'll scratch yours" situation developing. Clearly, organization culture will be a major determinant of whether an ad hoc SQA function is viable. I would have to know the organization before I could determine whether this would work or not.

What seems to have been very successful in small projects is setting up a culture where using the standards and procedures is "just the way we do things around here," re-inforcing the culture with process definitions that are clearly stated (and trained where possible, maybe through mentor), and using peer pressure as the enforcement mechanism. Then an ad hoc function with part time SQA people can be pretty effective. You can even embed it into your peer review process.

Do you feel there is a lower bound to the size (head count) and duration of projects that is outside the domain of CMM applicability?

In terms of the goals, no. In terms of implementation, definitely. I've worked in the two-person project environment before, and our SQA function was definitely driven by peer pressure - and professional pride. Ultimately that's what is need in any organization to make Quality work.

Added comment from Judah Mogilensky:

Let me add just a couple of comments to what Mark has already said.

The implicit assumption of the CMM's SQA KPA, founded upon a great deal of experience with Level 1 and Level 2 organizations, is that the Software Project Manager role is more concerned with schedule and budget than with process integrity. I might be willing to entertain an argument that, in a particular organization, the Software Project Manager is expected to sacrifice budget and schedule considerations in favor of process integrity, and that person is rewarded by the organization when he/she does so, and therefore that person is "objective" about compliance with policies, procedures, and standards. But I would be very skeptical, and I'd need a lot of convincing. And I would not consider having the Software Project Manager do SQA acceptable under any other circumstances.

If an organization is one in which most work is done by very small teams over very short durations, then I would think that kind of work would be very important to the organization. I would think that the quality of the products, and the predictability of the projects, would be crucial to the health of the organization. And I would think that having standard approaches to planning and tracking such short, small projects would be a top priority, to save projects from wasting time inventing their own or not following proven successful approaches. And therefore someone should be spending at least part of their time verifying that these small projects are being managed according to organizational standards, someone other than the Project Manager of each project. Someone who is in a position to raise a flag with the organization's management when a project is not being managed according to organizational standards. This, in a nutshell, is what the SQA KPA calls for.

As Mark knows, I am a strong advocate of the idea that organizations can evolve to the point where process compliance is part of the culture, where failure to comply with process is viewed as "socially aberrant behavior," and therefore is exceptionally rare or totally non-existent. In such a case, I believe that the formal organizational mechanisms of SQA may no longer be needed, like modern sutures that are absorbed into the skin and disappear when no longer needed to hold an incision together. However, to make this case, the organization needs to show both what mechanisms they have for sustaining this culture and indoctrinating new individuals, and they need to show that process compliance can be verified, and has been sustained at very high levels.

Small teams and short duration projects are not exemptions from the concept of process maturity (although they can typically make effective use of simpler, lighter forms). Watts Humphrey's next book advocates a Personal Software Process, where beginning programming students working alone on toy problems learn to apply a personal process from the start, and collect data, so that they can scale up their process as their software development skills grow and project scope expands.

Do you know what Activity 5 in Software Quality Assurance is looking for?
"SQA group audits" -
Is this looking for an audit of every designated work product or could it be a spot audit?

Designation can cover a variety of possibilities. If the Software Requirements Spec is simply designated to be audited, then it should be fully audited. However, the designation could be to spot-audit by some criteria that are specified as part of the designation. The key point is to decide in advance what the criteria are for auditing (random sampling, 100%, these products only, etc.) lest schedule pressures "crunch" the SQA process.

"products are evaluated"

Is this an evaluation by the QA group or could it be an evaluation by another group which the QA group verifies happens?

Either is valid. For example, evaluation could include the conduct of system testing performed by a testing group and the SQA group verifies that the system tests were performed according to the specified test process/criteria.

"customer"

There are internal and external customers, in fact every task produces a product which is then delivered to a customer, what sort of customer is meant here?

Both. For example, SRS may be evaluated via peer review before handover to the design "customers" and SQA reviews conduct of the peer review. Previous example was handover to external customer.

"designated"

Can designated be something that QA and the project agree to or is there some other criteria that should be used to determine designated.

Depends on what the corporate standards/procedures and project tailorings are. If the corporate standard cannot be tailored to remove some criteria, then project cannot prevent QA from dinging them. If project has validly tailored something, with the concurrence of the affected parties - as specified in the tailoring guidelines (I'm hitting this from a Level 3 perspective, sorry about that), then that's what QA audits.

Short answer: if organization has criteria, use them; if not, affected parties should agree on what designated means - early in process rather than when crunch is on.

Can you or the SEI provide any clarification or guidance to CMM users for SQA Commitment 1 on page L2-61 of 93-TR-25. We are concerned about whether the recommendation about "performance appraisal by the management of the software project they are reviewing." We found this recommendation in two places, but one of them is bracketed supplementary information which can be construed as either guidance or elaboration in this context. (The quotes follow below.) Could the appraisals be done by a manager in the same project, but above or independent of the software manager?

You're talking about specific implementations here. This concept maps up to Goal 2, which states that "Adherence of software products and activities to the applicable standards, procedures, and requirements is verified objectively" If an organization can demonstrate objective verification, then independence is not required. That demonstration is the responsibility of the organization however; an SCE team, for example, might have a different judgement, depending on how convincing the argument was.

To answer the questions specifically as asked, however, no, the appraisal would have to be done outside the project UNLESS you can DEMONSTRATE that there is an effective escalation mechanism in place that is not affected by the fact that the person (potentially) halting the project receives raises and promotions from the individual responsible for meeting schedules. That may not be easy to do. A close examination of noncompliances and their disposition would be in order.

Does the appraisal have to be done by a manager to whom the program manager of the whole project (including software) reports?

No. Could be matrixed in from a separate QA organization, for example.

Is there are general issue here that we are missing (indicated by the appearance of the requirement in two places?

Well, you're quoting from the Overview and the Key Practices. Realizing that there would be interpretation concerns about organizational structure, section 4.4.3 was written to try and clarify potential concerns and issues in thinking about different implementations. In going back and looking at it, I think I've just re-stated what it said. Similarly for the supplementary boxes.

We recognize how and why this independence is useful, so we have several solutions in mind, but we are also concerned that if our customers (users of the CMM and SCE) have a different interpretation, then our solution may not be satisfactory. Any clarification you may provide will be gratefully received.

I infer that your real concern is whether an SCE team would be capable of judging whether a perfectly adequate implementation was satisfactory or not. This is a larger question than specifically SQA, although that may be a particularly sensitive point. Hopefully, the Intro to CMM course will alleviate some of these concerns when we have SCE teams fully trained.

What are the intended meanings of the terms "deviation" and "noncompliance item/issue" as used in the CMM? It seems in the CMM these two terms do not mean the same thing since both are reported to a project but only noncompliance items are escalated to senior management. A brief survey here of CMM experience people came up with 4 different sets of definitions.

The CMM defines

> deviation - A noticeable or marked departure from the appropriate norm, plan, standard, procedure, or variable being reviewed.

in the glossary. Noncompliance item is not defined in the glossary; it is discussed only under the key practice QA.AC.7 on documenting and handling deviations.

Frankly, we didn't intend any subtle distinction. As QA.AC.7.2 suggests (but does not explicitly state), a deviation may result in a noncompliance item. It's like the difference between a failure and a problem report. Deviations occur, even if they're never identified, documented, and tracked to closure. The noncompliance item is a form of documented deviation (although there may be other forms as is perhaps suggested by QA.AC.7.1).

When to officially classify a deviation as a noncompliance item is fuzzy, and I infer that's the potential source of confusion. There is an implication that if a deviation is handled at the project level, it doesn't (necessarily) enter the reporting chain as a noncompliance item. We did not intend to say that this is the right way to track deviations / noncompliance items. It MAY be useful to separate deviations into items handled at the project level versus those that need to be escalated; it MAY be useful to not even track deviations handled at the project level.

All that we intended in the CMM was to delimit the end points: deviations occur; noncompliance reports are escalated to senior management as necessary. The middle territory is left "flexible" but our terminology may have made it actively ambiguous. If so, this may be an error in the CMM that should have a change request written for correction in the next release of the CMM. If you feel this is something that we need to do, please send in a CR.

Organization Process Definition

OPD Activity 1 - The organization's standard software process is developed and maintained according to a documented procedure. Does this paragraph imply that the "procedure" is a statement that indicates or acknowledges the organization's policies and standards will be satisfied and that state-of the practice tools and methods will be used?

Both of these are motherhood kinds of statements. For the first, if the policies, procedures, and standards are inconsistent or incompatible, it seems clear that you have a problem. For the second, we don't say anywhere in the CMM that you have to use state-of-the-practice tools and methods - just that you use appropriate tools for what you're doing. There may be reasons for not using state-of-the-practice tools (some reports indicate most organizations don't use structured programming yet, which is 70s vintage), but the efficiency of your process in a competitive situation may be so low that you can't survive.

OPD Activity 2 - The organization's standard software process is documented according to established standards. Do established standards simply specify that elements are decomposed "to the granularity needed to understand and describe the process", Do established standards simply state that "Each process element is described and addresses:

. . .", and that "relationships of the process elements are described and address: . . . "? Or do established standards provide how to document the process.

These standards would tell you, for example, what a process element looks like. Compare it to a design standard. Do design standards discuss granularity of the design? Do they tell you how to document a design? Do they tell you how to describe a design? Do they tell you how parts of the design can/should relate to one another? I think the answers to ALL these questions are yes, and the equivalent would hold true for process standards.

Do you have a sample or a template of a standard?

Some references:

Bill Curtis, Marc I. Kellner, and Jim Over, "Process Modeling," Communications of the ACM, Vol. 35, No. 9, September 1992, pp. 75-90.

David Harel, "On Visual Formalisms," Communications of the ACM , Vol.31, No. 5, May 1988, pp. 514-530.

David Harel, H. Lachover, A. Naamad, A. Pnueli, M. Politi, R. Sherman, and A. Shtul-Trauring, "STATEMATE: A Working Environment for the Development of Complex Reactive Systems," Proceeding of the 10th IEEE International Conference on Software Engineering, Singapore, IEEE Press, 13-15 April 1988.

G.F. Hoffnagel and W. Bergei, "Automating the Software Development Process," IBM Systems Journal, Vol. 24, No. 2, 1985, pp. 102-120.

Watts S. Humphrey and Marc I. Kellner, "Software Process Modeling: Principles of Entity Process Models," Proceedings of the Eleventh International Conference on Software Engineering, IEEE Computer Society Press, 1989, pp. 331-342.

Marc I. Kellner and Gregory A. Hansen, "Software Process Modeling: A Case Study," Proceedings of the Twenty-Second Annual Hawaii International Conference on Systems Sciences, Vol. II - Software Track, IEEE Press, 1989, pp. 175-188.

Marc I. Kellner, "Software Process Modeling: Value and Experience," SEI Annual Technical Review, Carnegie Mellon University, Pittsburgh, PA, 1989, pp. 23-54.

Marc I. Kellner, "Software Process Modeling Support for Management Planning and Control," Proceedings of the First International Conference on Software Process, IEEE Computer Society Press, 1991, pp. 8-28.

R. A. Radice, N. K. Roth, A. C. O'Hara, Jr., and W. A. Ciarfella, "A Programming Process Architecture," IBM Systems Journal, vol. 24, no. 2, 1985.

Ronald A. Radice and Richard W. Phillips, Software Engineering: An Industrial Approach, Volume 1, Simon & Schuster, Englewood Cliffs, NJ, 1988.

Training Program

The contractor had a terrific training plan but did not implement it because "the customer would not pay for it." None the less, the SCE team assessed them as inadequate since there was no implementation. Should the customer pay for training?

If the customer has an interest in an on-going customer/supplier relationship, then it behooves the customer to be proactive in supporting the development of skills pertinent the area of the relationship.

If the customer refuses to pay for training, however, the contractor should bite the bullet and train their people out of their own pocket. This is a basic cost of quality issue. If you believe that:
1) people who are happy about their continuing professional development are more likely to stay with a company
2) high turnover leads to both low productivity and low quality, with a corresponding impact of high cost
3) training is value-added because it increases the skill of the people doing the work then I don't see how a contractor can legitimately claim that "the customer won't pay for training" makes training Not Applicable. It just doesn't hold for me.

The flip side, of course, is that:
1) training costs money
2) people being trained are not working on a product
3) key people may be needed on the project at critical times

It's long-term vs short-term views that we're looking at. If people are viewed as a resource to be used up and discarded when the contract is over, then I'm not sure that contractor is one that I would want to establish a long-term customer/supplier relationship with.

Of course there's also the possibility that the "terrific training plan" was really pretty lousy and not worth implementing if the customer wouldn't pay for it :-)

The question concerns the apparent absence in the Software Project Planning KPA of the need to define project-specific training in the project plans. Is this an oversight or an omission that will be corrected in the next version of the CMM? I would think, as a matter of good project planning, that plans should specify the training that team *members will need to receive, the timing of same, the potential impacts to schedule of off-site training, the cost of training billed to the project, etc. But the CMM seems to omit these notions.*

Project training plans are specifically mentioned in Level 3 in Training Program, but you're right that they aren't mentioned at Level 2. It is certainly an omission that should be considered. If anyone thinks it's important enough to write a change request (hint, hint), then it would probably be added in the next version of the CMM.

Integrated Software Management

ISM Activity 1 - The project's defined software process is developed by tailoring the organization's standard software process according to a documented procedure.. How detailed is a "documented procedure?"

Detailed enough to be useful; not too detailed to be usable.

Do you have a sample or a template of this procedure?

The bottom line is that the CMM leaves a lot of flexibility in terms of level of detail, etc., that's needed to document processes. There are a number of factors that influence how you document processes, including:

size of organization
organizational culture
size of project
determinism of process
process description tools
degree of automated process support
technical sophistication of management
budget available for process definition
degree of organizational support for process definition
etc.

There are no simple answers to this question.

Peer Reviews

The contractor had great procedures for walkthroughs in their "generic SDP". But EVERY project tailored out all references to requirements, code, and test case walkthroughs and did not use any forms for recording preparation effort or summarizing *findings (they did however have adequate Action Item Logs). The project level SDPs which tailored out these walkthrough procedures were approved by the company's Software Review Board which oversees implementation of the "procedures in the generic SDP." But the team assessed this KPA as inadequate. Should a KPA be acceptable if a company has good walkthrough procedures but tailors (most) all of them out with the approval of a higher review authority?*

I consider this a judgement call on the part of the SCE team, in light of the needs of the acquisition. As described, this is a case of trying to score well, rather than implementing a process. A company-level process should be applied to the majority of cases (although when there are alternatives, picking one of the alternatives is the reasonable interpretation of majority rule). Consistently tailoring out practices violates the intent of the CMM.

The judgement might be that Peer Reviews is satisfied (if the procedures are good), but Software Product Engineering is not, since SPE calls out peer reviews for requirements (PE.AC.2.8), code (PE.AC.4.4), and test cases (PE.AC.5.6). I could infer from your description that adequate peer reviews are implemented in some areas, such as design. The question, if I wanted to probe more deeply, would then be what alternative quality control mechanisms is used for these particular cases.

MANAGEMENT

- Welcome
- Capability Maturity Modeling
- Team & Personal Software Process
- IDEAL Model
- Risk Management
- Software Engineering Measurement & Analysis (SEMA)
- Software Engineering Information Repository (SEIR)
- Software Process Improvement Networks (SPINs)
- Appraiser Program
- Acronyms
- SEI Initiatives
- Conferences
- Education & Training

Questions and Answers about the Software Capability Maturity Model® (SW-CMM® (Q&A #3)

Questions and Answers on the CMM

Mark C. Paulk
Issue #3 - 29 March 1996

I am frequently asked questions on how to interpret the CMM in various contexts. The following "newsletter" may be of general use. I have to caveat these answers by saying that they represent my opinion and do not represent official SEI positions. They have typically not been internally reviewed, but they do represent a well-informed opinion on what the CMM means.

You are welcome to ask questions on the CMM. I do not guarantee quick answers, although I'll try to get back to you as soon as possible. Sometimes that's the same day I receive a question, sometimes it may be a month, depending on my travel schedule and work load. I prefer to receive questions by e-mail. I will lightly edit and sanitize question and answer before distribution in this newsletter.

Mark Paulk
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
FAX: 412-268-5758
E-mail: mcp@sei.cmu.edu

The questions in this issue relate primarily to Intergroup Coordination and Training Program.

Intergroup Coordination

Encouraging integrated product development

Q. How does the SEI through the CMM process, address the growing project organizational standard regarding Integrated Product Teams (IPTs). In the IPT concept the IPT is fully empowered to have, in essence, "womb-to-tomb" responsibility for its product. Where there is a significant software content this can be at odds with the CMM in the area of a software manager reporting to the project manager. The CMM speaks to the need to have a software manager responsible for the software and its development where there is meaningful software development on a project. In concurrent engineering and the employment of IPTs this responsibility is empowered to the IPT leads. In some cases a major project will have multiple CSCIs which all contribute to the ultimate system. In this case the CSCIs are each lead by an IPT leader and his/her team. It is in this context that the position of a software manager, who has responsibility for all the software on the project, becomes a concern. My question to you is, what is the SEI's thinking in the area of addressing Concurrent Engineering/IPTs and the desirability of empowered teams while at the same time addressing the issue of a software manager who has responsibility for the software? The two seem to be at odds with each other.

A. I would certainly encourage the IPT approach. The concern seems based on a misunderstanding of how to use the CMM properly. If you view the practices as a set of "shall" statements that have to be slavishly followed, then there's a clash. The CMM, however, does not have a single shall statement in it. Focus on the KPAs and their goals. Apply professional judgment whether you are achieving those goals.

Having a software manager who reports to the project manager supports all of the goals for PTO, but there is a focus in PT.CO.1 on dealing with Goal 2 of PTO on taking corrective action. Making software issues appropriately visible is the concern; in the IPT environment that may be less of an issue than in a more traditional environment because of the empowerment. I would want to confirm that software issues don't get submerged so that there are surprises near the end of a project, but if the IPT philosophy holds, that should not be a problem.

In fact, there is a proposal to do a variant of the CMM for IPT next year, which may clarify some of these concerns. Personally, I am a strong proponent of IPT and concurrent engineering. I see this as simply an alternate implementation that fully satisfies (potentially exceeds, in terms of process effectiveness) the goals of the CMM. As an organization matures, I suspect it would move towards an IPT approach. See Loral Houston's OnBoard Shuttle project as an example of such a maturation through their process control boards.

IC policy and scope

Q. Commitment 1, Intergroup Coordination. Does this statement mean that engineering teams are established across disciplines or organizations or does this statement refer to the establishment of interdisciplinary teams on a project, i.e. all disciplines are represented on the project?

A. The scope of IC is set in the introductory paragraphs, where it says: "The purpose of Intergroup Coordination is to establish a means for the software engineering group to participate actively with the other engineering groups so the project is better able to satisfy the customer's needs effectively and efficiently.

Intergroup Coordination involves the software engineering group's participation with other project engineering groups to address system-level requirements, objectives, and issues."

So it is within the project that we're talking about in IC.CO.1. The implementation could be matrixed, concurrent engineering, integrated product teams, etc., which might well look interdisciplinary across the organization as projects come and go, but IC is project-specific.

IC and project size

Q. We have a big software project with three subgroups producing 3 different software packages and one integration subgroup for the seamless integration of the three. Is KPA Intergroup Coordination applicable for this project?

A. Yes.

Q. Is Intergroup Coordination only applicable for a big system which involves both hardware and software groups?

A. No. All engineering and project groups are included under IC.

IC and senior management

Q. How is "senior management" defined in the CAO/IPT concept ?

A. Senior management would be the same as for non-IPT: A management role at a high enough level in an organization that the primary focus is the long-term vitality of the organization, rather than short-term project and contractual concerns and pressures. In general, a senior manager for engineering would have responsibility for multiple projects.

Or in this case, multiple teams.

Training

Formal training

Q. CMM formally Documented Training Plans for SEI Level 2: Regarding training for SEI L2 qualification, it seems to me that every KPA has "Abilities" defined for training. Here are examples: Req Mng Ability 4, Proj Plan Ability 4, Proj Track Ability 4, SQA Ability 3&4, SCM Ability 4&5. From your experience, do these abilities require a formal training plan and formal practice to achieve SEI Level 2?

A. What do you mean by formal? The short answer is NO, but some organizational infrastructure is desirable in terms of identifying common training needs, deciding what courses answer your specific training requirements, coordinating attendance, etc.

Q. There is obviously more emphasis on formal training in the KPCMM Level 3 KPA "Training Program".

A. Yes. At Level 3, you are expected to have identified roles, defined training needs, and specified required training.

Q. I know that in the SEI Technical Report CMU/SEI-87-186, "A Method for Assessing the Software Engineering Capability of Contractors", page 75, question 1.2.2, it asks if there is a required training program for all newly appointed managers. I assume this means a formally documented training plan, right? But it doesn't cover most of the training mentioned in the Level 2 abilities I have listed above, so I am not sure. What do you think?

A. Documented training plan and records, yes. There are only 85 questions in the 1987 questionnaire. There is a sampling issue. It's better addressed in the 1994 questionnaire.

Training deployment

Q. I would prefer video courses as much as possible for cost reasons. Also, in general, during a SEI evaluation, how many personnel (all, 80%, 50%) have to be trained before you are considered to have met the KPA objective?

A. That's a judgment call, but in general "all". You may have folks waived from training if they don't need it, and a few folks (like new hires) may be in the queue, but in general everyone who needs training should have had it to satisfy the pertinent KPA. That's one of the reasons deployment is so hard and it takes a while to get to the next level -- it's not just saying you're going to offer it, but to have actually done so.

QUESTIONS

Return to top of the page

Return to main page

Questions and Answers on the CMM

Mark C. Paulk
Issue #3
29 March 1996

I am frequently asked questions on how to interpret the CMM in various contexts. The following "newsletter" may be of general use. I have to caveat these answers by saying that they represent my opinion and do not represent official SEI positions. They have typically not been internally reviewed, but they do represent a well-informed opinion on what the CMM means.

You are welcome to ask questions on the CMM. I do not guarantee quick answers, although I'll try to get back to you as soon as possible. Sometimes that's the same day I receive a question, sometimes it may be a month, depending on my travel schedule and work load. I prefer to receive questions by e-mail. I will lightly edit and sanitize question and answer before distribution in this newsletter.

Mark Paulk
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
Fax #: (412) 268-5758
Internet: mcp@sei.cmu.edu

The questions in this issue relate primarily to Intergroup Coordination and Training Program.

Intergroup Coordination

Encouraging integrated product development

How does the SEI through the CMM process, address the growing project organizational standard regarding Integrated Product Teams (IPTs). In the IPT concept the IPT is fully empowered to have, in essence, "womb-to-tomb" responsibility for its product. Where there is a significant software content this can be at odds with the CMM in the area of a software manager reporting to the project manager. The CMM speaks to the need to have a software manager responsible for the software and its development where there is meaningful software development on a project. In concurrent engineering and the employment of IPTs this responsibility is empowered to the IPT leads. In some cases a major project will have multiple CSCIs which all contribute to the ultimate system. In this case the CSCIs are each lead by an IPT leader and his/her team. It is in this context that the position of a software manager, who has responsibility for all the software on the project, becomes a concern. My question to you is, what is the SEI's thinking in the area of addressing Concurrent Engineering/IPTs and the desirability of empowered teams *while at the same time addressing the issue of a software manager who has responsibility for the software? The two seem to be at odds with each other.*

I would certainly encourage the IPT approach. The concern seems based on a misunderstanding of how to use the CMM properly. If you view the practices as a set of "shall" statements that have to be slavishly followed, then there's a clash. The CMM, however, does not have a single shall statement in it. Focus on the KPAs and their goals. Apply professional judgment whether you are achieving those goals.

Having a software manager who reports to the project manager supports all of the goals for PTO, but there is a focus in PT.CO.1 on dealing with Goal 2 of PTO on taking corrective action. Making software issues appropriately visible is the concern; in the IPT environment that may be less of an issue than in a more traditional environment because of the empowerment. I would want to confirm that software issues don't get submerged so that there are surprises near the end of a project, but if the IPT philosophy holds, that should not be a problem.

In fact, there is a proposal to do a variant of the CMM for IPT next year, which may clarify some of these concerns. Personally, I am a strong proponent of IPT and concurrent engineering. I see this as simply an alternate implementation that fully satisfies (potentially exceeds, in terms of process effectiveness) the goals of the CMM. As an organization matures, I suspect it would move towards an IPT approach. See Loral Houston's OnBoard Shuttle project as an example of such a maturation through their process control boards.

IC policy and scope

Commitment 1, Intergroup Coordination. Does this statement mean that engineering teams are established across disciplines or organizations or does this statement refer to the establishment of interdisciplinary teams on a project, i.e. all disciplines are represented on the project?

The scope of IC is set in the introductory paragraphs, where it says: "The purpose of Intergroup Coordination is to establish a means for the software engineering group to participate actively with the other engineering groups so the project is better able to satisfy the customer's needs effectively and efficiently.

Intergroup Coordination involves the software engineering group's participation with other project engineering groups to address system-level requirements, objectives, and issues."

So it is within the project that we're talking about in IC.CO.1. The implementation could be matrixed, concurrent engineering, integrated product teams, etc., which might well look interdisciplinary across the organization as projects come and go, but IC is project-specific.

IC and project size

We have a big software project with three subgroups producing 3 different software packages and one integration subgroup for the seamless integration of the three. Is KPA Intergroup Coordination applicable for this project?

Yes.

Is Intergroup Coordination only applicable for a big system which involves both hardware and software groups?

No. All engineering and project groups are included under IC.

IC and senior management

How is "senior management" defined in the CAO/IPT concept ?

Senior management would be the same as for non-IPT: A management role at a high enough level in an organization that the primary focus is the long-term vitality of the organization, rather than short-term project and contractual concerns and pressures. In general, a senior manager for engineering would have responsibility for multiple projects.

Or in this case, multiple teams.

Training

Formal training

CMM formally Documented Training Plans for SEI Level 2: Regarding training for SEI L2 qualification, it seems to me that every KPA has "Abilities" defined for training. Here are examples: Req Mng Ability 4, Proj Plan Ability 4, Proj Track Ability 4, SQA Ability 3&4, SCM Ability 4&5.

From your experience, do these abilities require a formal training plan and formal practice to achieve SEI Level 2?

What do you mean by formal? The short answer is NO, but some organizational infrastructure is desirable in terms of identifying common training needs, deciding what courses answer your specific training requirements, coordinating attendance, etc.

There is obviously more emphasis on formal training in the KPCMM Level 3 KPA "Training Program".

Yes. At Level 3, you are expected to have identified roles, defined training needs, and specified required training.

I know that in the SEI Technical Report CMU/SEI-87-186, "A Method for Assessing the Software Engineering Capability of Contractors", page 75, question 1.2.2, it asks if there is a required training program for all newly appointed managers. I assume this means a formally documented training plan, right? But it doesn't cover most of the training mentioned in the Level 2 abilities I have listed above, so I am not sure. What do you think?

Documented training plan and records, yes. There are only 85 questions in the 1987 questionnaire. There is a sampling issue. It's better addressed in the 1994 questionnaire.

Training deployment

I would prefer video courses as much as possible for cost reasons. Also, in general, during a SEI evaluation, how many personnel (all, 80%, 50%) have to be trained before you are considered to have met the KPA objective?

That's a judgment call, but in general "all". You may have folks waived from training if they don't need it, and a few folks (like new hires) may be in the queue, but in general everyone who needs training should have had it to satisfy the pertinent KPA. That's one of the reasons deployment is so hard and it takes a while to get to the next level -- it's not just saying you're going to offer it, but to have actually done so.

Questions and Answers about the Software Capability Maturity Model® (SW-CMM® (Q&A #4)

Mark C. Paulk
Issue #4 - 7 April 1997

I am frequently asked questions on how to interpret the CMM in various contexts. The following "newsletter" may be of general use. I have to caveat these answers by saying that they represent my opinion and do not represent official SEI positions. They have typically not been internally reviewed, but they do represent a well-informed opinion on what the CMM means. The Q&A has been sanitized for proper names (where appropriate), but it has not been cleaned up for publication -- it's pretty much the off-the-cuff reply that folks have gotten to their questions.

You are welcome to ask questions on the CMM. I do not guarantee quick answers, although I'll try to get back to you as soon as possible. Sometimes that's the same day I receive a question, sometimes it may be a month, depending on my travel schedule and work load. I prefer to receive questions by e-mail. I will lightly edit and sanitize question and answer before distribution in this newsletter.

Mark Paulk
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
FAX: 412-268-5758
E-mail: mcp@sei.cmu.edu

The questions in this issue are fairly general. Topics covered include:

- TQM and CMM
- Organizational analysis
- If the customer won't pay
- Tailoring
- Small projects
- Common threads in the CMM
- Incremental development
- Incremental process improvement
- Legacy systems and maintenance documentation
- Software project dynamics
- Not Applicable versus risk
- Discipline versus bureaucracy
- COTS
- Required overtime
- Methodologies
- Regular versus periodic

TQM and CMM

Q. I would like to know what you think the relationships are between TQM and its philosophies and the CMM. ... I have been reading more and more articles lately stating the philosophies of Deming, Juran and Crosby are the basis for the CMM. I also am seeing more articles, similar to yours, merging ISO 9000 and the CMM. I do not see the connection to the TQM founders.

It seems to me a company could be ISO 9000 and CMM level 4 and be creating cement rain coats. They may be the highest quality cement rain coats ever manufactured. They may be manufactured with very mature processes. The company may be following all their processes definitions right from the ISO 9000 guidelines. But what good is it if no customers want cement rain coats.

TQM is customer focused. It has as its foundation the needs of society. Deming's point is that healthy companies with healthy employees are necessary for the survival of a society. What he documented with his 14 points and his 7 deadly diseases has little to do with the CMM. Maybe one is about the why and the other is about the how.

The precepts and practices of TQM are essential.
The precepts and practices of the CMM are essential.
ISO 9000 certification seems to be becoming another essential.

These are all complimentary. They are also synergistic. They have some common points, like continuous improvement, but that does not mean they have come from the same place or are meant to do the same things. Good, logical, and practical solutions to problems such as what TQM and the CMM are attempting to solve will overlap.

I am not very far along in this journey. I am very likely dead wrong.

What do you think the relationships are between TQM and its philosophies and the CMM?

A. CMM only attacks the process side of TQM, and specifically for software engineering, although the principles can be applied to any (engineering) discipline.

There are some very important aspects of TQM that are deliberately not addressed in the CMM. What value, for example, would the SEI add to a discussion of the "people" issues? Is there anything that's software-specific about "people"?

The obvious answer is no, yet we can add value in the software process arena, while at the same time we should recognize the importance of the people issues in enabling software process improvement.

As I said, we made a deliberate decision to focus on the area where we could make a major contribution. Things change over time...

The desire of some organizations to have a "People CMM" to help them build their human resource has led to the People CMM. The word software isn't mentioned a whole lot... and it was written by HR-knowledgeable folks.

We are drafting a chapter for CMM v2 on interpersonal skills and organizational change

- not as key practices that describe maturity levels, but in the context of enabling process improvement and pointing to better sources of information, such as the P-CMM, Deming, etc. A lot of excellent work has been done by folks much more knowledgeable than we about interpersonal skills!

We also don't (currently) talk about strategic business planning or customer satisfaction explicitly, which are some of the other important TQM topics. We may do something about those in CMM v2 also, but we are the _software_ engineering institute and need to stay focused on our primary mission, while not ignoring the critical interdependencies - a difficult balance to maintain.

The CMM was inspired by Crosby's maturity grid, but it has evolved a lot over the past several years. If you'd like to track it's evolution, you could read:

Mark C. Paulk, "The Evolution of the SEI's Capability Maturity Model for Software," Software Process: Improvement and Practice, Vol. 1, Pilot Issue, Spring 1995, pp. 3-15.

Q. Will it cost as much to produce a product when on level 1/2/3/4 as it will producing the equivalent product when on the level above? (Has any company really performed this experiment? A 2 by 2 matrix?)

A. The ones who have data have demonstrated to their satisfaction that it's cheaper and quicker to build products at a higher maturity level.

Q. Or does the product automatically change when you produce it being on a different/higher level?

A. That's a business decision. You have added opportunities on the cost/functionality tradeoff curve.

Q. Of course, the workers will give the product a higher subjective value. So does the management, hence the selling price is likely to be higher.

Is the cost (as perceived by the manager) to produce a product knowing how the total cost is distributed over all activities involved less than the perceived cost when nothing is known about the cost distribution?

Will you charge the same for a s/w product when you're on level 2/3/4/5 as you charged for an equivalent product (functionality perceived by the customer) when you were on the level below?

A. That's a business decision.

Q. If the cost of developing s/w products really decreases as you move up the CMM ladder, how will you be looked upon by potential customers if you claim to sell/produce products with the same quality as your competitors notably cheaper?

A. Almost by definition, it won't be the same quality and cheaper. It will have to be higher quality and cheaper. You're forgetting rework costs. About 40% of software project costs are in fixing bugs to get something shippable. If appraisal/prevention permits you to build stuff for 30% less, by definition it's going to be a higher quality product - see TQM literature.

Q. I've been told that most companies decide to produce cheap products or high quality products (read expensive). But there are some companies that try to sell/produce cheap high quality products (called in-betweens?). How do these in-betweens perform on the market compared to the other two?

A. Don't assume that all business decisions are consciously made ;-)

Also, don't assume that a Civic is a low-quality car just because it's cheap. It's a high quality car (remember fitness for use!) targeted to an inexpensive market. Quality /= luxury.

This is the classic misunderstanding of the quality movement. Before studying the software process improvement world, study the TQM world. Read Deming and Juran. SPI is an application of those more general concepts in the software world.

Q. I think that levels 2 and 3 are quite different/separate from levels 4 and 5. My view is that levels 2 and 3 are parts of a larger goal, namely to stabilize the process(es), i.e.,, to minimize the variance/deviation/fluctuations.

Levels 4 and 5, on the other hand, deal more with adjusting "the level/height of the curve". This is just my opinion. Is it totally wrong? (Doesn't this imply that defect prevention is misplaced at level 5? I don't know enough of the theory. I have lots and lots of reading to do...)

A. Think of it as qualitative vs. quantitative process control then process improvement - which gives you 4 levels above level 1.

Q. This means that you disagree with the basic principle of TQM. While I don't particularly mind, it's a very old argument - that quality costs - that's been debated by far abler folks than I.

I don't disagree. But, I think (it's just an uneducated opinion) it's reasonable to assume that the levels 2 and 3 block will give a better "ROI" than t he upper block. Why? Because it is probably easier and cheaper to remove the first errors.

A. Pick the low hanging fruit, as we say in the States.

Q. Again, this goes back to the old "quality costs" debate. We're seeing so me of this resurface under "good enough software". I have no problem with this concept so long as you realize this is a "fitness for use" debate; when i t becomes operationalized as "whatever junk we can get a customer to buy", I think the market will determine pretty effectively.

I think I'm getting your point, but I'm not sure. Please, could you elaborate e this a bit? I am having trouble remembering all the stuff all the time.

A. Analogy: GM said they built good cars and could point to domination of market in 50's through early 70's as proof that they understood what was good enough. The environment changed... and the companies that had focused on "quality" made enormous inroads...

Q. Do you think that all SE developers should try to achieve level 5? And, do you think that they all have go "all the way", or do you think that companies like Microsoft necessarily don't have to go as far as companies like LORAL Houston, i.e.,, can improvement/changes be overdone?

A. Go as far as necessary. If you're level 3, you're world class today. What the competition may inspire in the future, in terms of business reasons for continuing to improve and how you improve, will determine your strategy later.

Q. The ones who have data have demonstrated to their satisfaction that it's cheaper and quicker to build products at a higher maturity level.

It is my opinion that there isn't conclusive proof, yet (At least I think I haven't seen any proof). Intuitively, the CMM has to work! That is, I hope and think that quality is free (maybe I should have said this in my first e-mail...). But, what I'm not sure about is that "state-of-practice" quality is free. [You have to put this into my context: in Sweden people cannot sue for astronomical amounts of money :-)]

The main problems are that I don't know how they came to the results. How do they define/measure/calculate productivity for example? And, how do they define/measure/calculate the cost and the return on changes that supposedly affect the productivity (reprise)?

A. They defined it based on what they considered important in their environment - which is always good advice. Their papers provide some details, but they're all different.

Organizational analysis

Q. Responding to a customer's request, here is the situation:

Large corporation with 6 sites around the country. The corporation has 8 major systems; one at each site plus two sites with two each. The major systems make up ~60% of the corporation's business. The corporation has a SPI effort that encompasses all 6 sites.

Their assessment plan is to assess a major system at each site (i.e, assess 6 representative systems of the corporation). Assume: the assessment result is that the 6 systems assessed are level 2. What can the corporation say:
a) That the corporation is a level 2 organization.

A. Possible, but I'm not really comfortable with it as stated. This isn't just a sampling issue, they're really looking at six different sites with what is presumably their best and brightest.

Are these development efforts? Maintenance projects? Operations?

What about the 2 missing major systems? Will this skew the perception of the process? You can do ML2 diagnosis on individual projects without worrying about organizational infrastructure, but if you're skipping 2 major systems, why?

Q. b) That the major systems of the corporation operate at level 2.

A. It isn't the systems, it's the projects or sites that operate at ML2.

Q. c) Nothing about being level 2 [if you say c), please explain your reasoning some].

A. They're violating a bunch of the organizational analysis heuristics, I think. They're basically claiming the organization is the whole company, in spite of geographical dispersion, possible major cultural differences, possible deployment issues, etc. Who is the sponsoring manager of the SPI effort? Are they directly funding SPI efforts at each site? Is there sponsorship physically located at each site?

Major qualms about saying "we're ML2" in this kind of context. Certainly if we "endorse" their statement, we should know a lot more.

Q. Would you say something different if you were the lead assessor? That is, what would you say about the results when reporting them (assume: you can't change the assessment scenario, but feel free to add comments on how you would change it).

A. Tough question. This is also sampling issue, similar to 1 big project + bunch of small ones. Issues of consistency and predictability can be significantly skewed by outliers, whether more or less mature. What is the definition of "typical project" for company?

I'd go for option C unless I knew a lot more about the assessment. For political reasons, that may not be viable, but the existence of the political reasons influences me to believe the organization is likely to be ML1 with ML2 on some critical projects that get a lot of management attention.

Q. Most of the CMM seems to deal with a single development organization, but many companies have dozens of development organizations at a site, and try to get an SE I rating for the entire site. This leaves a lot of questions unanswered.

A. My general comment on this is that it's a bad idea to overload organizations like you're suggesting happens. I know that it happens also. Depending on why you're going for an SEI rating, this can have very bad consequences: 1) ineffective improvement, 2) an SCE team may not make the same mistake (it's one of the reasons determined for some of the maturity level mismatches between SCEs and SPAs several years ago), 3) going for a score rather than better business value. Problems, any way you look at it.

However, I would comment that "dozens" seems like an exaggeration. "Several" seems a better term and "few" might be even more appropriate. "One" is, I agree, rare for sizable companies. Organizational analysis is one of the least well-defined aspects of doing appraisals.

Q. For example, During the analysis of data (step 3, Response Analysis, in your figure 4.1) how do you handle an organization that "falls down" on key process areas in one group, and scores well on those same KPAs in another group?

A. The organization "fails." The weakest link determines organizational maturity. There are exceptions, e.g., legacy systems, but inconsistency of deployment is one of the hallmarks of a level 1 organization. Every company (that survives) has pockets of excellence. Organizational learning implies a lot more! In particular, predictable implementation of a consistent process unless there are "external drivers" otherwise.

Q. In some organizations that I have worked for, they gave the full score because the y felt that the capability existed somewhere in the organization.

A. There's an optimist in every crowd... rating processes is hard, but this is clearly over the line of acceptable judgment.

Q. In other places, I've seen them apply a weighting based on the amount of software produced by each group.

A. I don't think that would work very well either. Is a small amount of life-critical software less important than a large amount of MIS software? I don't think so... that's one of the problems with algorithmic approaches -- and why I prefer heuristics that warn an assessment team of the kinds of issues to be sensitive too, but leaves the responsibility to the team to apply reasonable professional judgment.

Q. Personally, I feel that the purpose of the CMM/SCE is to identify and prioritize areas needing improvement, and therefore the areas needing improvement need to be pointed out as having geographical or other inconsistencies, and the scoring needs to be based on the WORST rather than the best groups to place the proper management influence on the problem areas.

A. Seems reasonable to me ;-)

Q. However, no one (in Intel or Motorola for examples) would ever do it that way because it reflects badly on them. I believe that this is because of a violation o f the spirit of the SEI, and of Deming's beliefs (that metrics should never be used to rate people, just the process).

A. I'll pass this along to some of our CBA IPI and SCE folks. I agree with you in concept, although I don't know enough about your specific examples to judge. Some of the toughest issues in assessment include:

- organizational analysis (what is the scope of an appraisal?)
- project sampling (what is a representative project? how many are needed in appraisal?
- what are the "exceptions to the rule" that need to be thoughtfully considered re their impact on process and organizational capability? E.g., legacy systems, very small projects (tasks), rapid prototyping projects,...

From James Hart:

There are (being somewhat realistic) a few requirements that must be met before an organization of that size and geographical separation can say they are at ML 2. For them to say they are at level 2, they must say this is true for ALL projects in the organization, with only the exclusion of outliers (e.g., short duration projects, 2-3 man efforts, those outside the normal business direction). In order for them to make this claim, then the projects must really be selected in a somewhat random way. Preselection of 4-5 projects based on criteria OTHER THAN PROCESS-RELATED will skew their results.

Second, you can only answer such questions as you ask in the context of why they want to do an assessment. If it is to validate their level and make public the results, they have potential problems with getting a REALITY-check. People will say and report what they think management wants (or needs) to hear. Since they wish to do a consolidated assessment (in essence, combining six assessments into one), then it does not sound like they are interested primarily in identifying key weaknesses.

Finally, projects selected for assessments only make up a major part of who is talked to on an assessment. During interview sessions with personnel, you will want to select those OUTSIDE of the projects selected. Their inputs will effect the results, so you can't really say the assessments will only cover their major systems. 60% of the business may not be 60% of the people or 60% coverage of project processes.

In summary, I would personally not hold much faith in results of the kind suggested. From experience, I have one data point: an organization wanted to do an assessment over a very broad range of software developers. We did as they wanted; but they were not happy with the results. The reason was due to diversity in processes across the groups; each of the resulting findings were issues for a PORTION of the assessment's scope, and NONE were REPRESENTATIVE of issues the ORGANIZATION faced as a whole. They subsequently sent 4-6 months following the assessment weeding through each of the findings to determine what portions of the organization needed to address what findings. Consolidating results across diverse areas leads to LESS meaningful data.

A related point raised by Balzer: at L4-5, it may not be appropriate to focus on specific practices these high-maturity companies use, but rather at the meta-level, what it is they do with process capability. Bob's thesis (which I believe has merit and which is consistent with a conversation we had with Curtis 3 years ago) is that as organizations mature, their processes more-and-more reflect their product and product family architectures and their important quality features - and these in turn drive the specific features of interest in measuring the capability (my paraphrase), so there may not be a lot of commonality in what is done at the detailed practice level, but more likely at the level of what they do with capability. This suggests the real enhancements at L4-5 may not be so much in additional KPAs, but rather:

1. achieving the right level of detail on practices determining, measuring, and manipulating process capability,
2. better examples (reflective of different product domains, etc.), and
3. making stronger meta-connects with business goals and needs.

If the customer won't pay

Q. Here's a new one. Can an organization be rated Level 3 if a project does not comply with Level 2 or 3 KPAs if their customer has specifically said he doesn't want to pay for them? This is in a defense, contract based environment.

A. Yes, if it's a rare occurrence, but I would want to see a directive from the customer that they refuse this functionality and accept the risk that it entails. More generally, I would expect a company to "do the right thing" even if the customer says don't do this -- to the extent that a supplier would charge MORE for not doing SCM than for doing it (after all, the reasons for the KPAs are driven by business concerns).

Tailoring

Q. We have been trying to understand the applicability of the CMM to short cycle time software support activities. We concluded that applying the CMM in all its rigor will add to the cycle time of the support activity and possible customer dissatisfaction.

I am now thinking of formulating a support maturity model with the same five levels as the CMM but with different key practices and KPAs.

Are there some guidelines to selecting KPAs and KPs for a maturity model? Is there a generic process for this? Are there any references available? Any light you can throw on this will be deeply appreciated.

Q. A report on tailoring the CMM is going through the approval cycle now. It's by Mark Ginsberg. Check with our customer relations people in January (customer-relations@sei.cmu.edu) to see if it's been released and/or check the anonymous ftp site for it (ftp.sei.cmu.edu, pub/cmm or pub/documents).

I would disagree with the idea of having different KPAs and claiming it's the same maturity levels. I would strongly argue that all of the KPAs, with the exception of subcontract management, apply in any software development or maintenance environment. The implementation may differ dramatically, and so arguing that different key practices may needed may be appropriate, but organizations that have tailored the CMM for small projects observed that 90%+ of the key practices carried forward into that environment. Where you really get large project/organization specific is in the subpractices. What is important is mapping the concepts and roles of your environment to the concepts and roles in the CMM. Once you establish that relationship, then analyze the applicability of the practices.

I would also refer you to the first newsletter on tailoring the CMM.

Q. I am an SQA member of the SEPG at ZZZ. Recently we have been attempting to institute Level 2 process on smaller projects. I was reading through the CMM v1.1 (Feb '93) document and came across the following paragraph:

Has a tailored version of the CMM addressing smaller projects been developed? If not, has this issue been addressed to some degree in other publications? How can we obtain such information?

A. We held a workshop on this topic a couple of years ago and concluded that there was too much variation between small projects and organizations to justify a CMM tailored specifically for the "small" environment. The challenges were shared, in general, by "non-small" environments. A resident affiliate, Mark Ginsberg, had come to the SEI to work on a CMM/small, but this changed the direction more to tailoring in general. The ultimate result was

Mark Ginsberg and Lauren Quinn, "Process Tailoring and the Software Capability Maturity Model," Software Engineering Institute, CMU/SEI-94-TR-024, November 1995.

The Software Capability Maturity Model(®) (SW-CMM[®]) is serving as the foundation for a major portion of the process improvement being undertaken in the software industry. It is composed of two volumes: the Capability Maturity Model for Software and the Key Practices of the Capability Maturity Model. The key practices of the SW-CMM are expressed in terms that reflect normal practices of organizations that work on large, government contracts. There is, however, a significant population of software-producing and -acquiring organizations, operating in different environments, for which the key practices require significant interpretation and/or tailoring, prior to application. This report presents a tailoring framework that identifies process artifacts, tailoring processes, and their relationships to project artifacts, and explores the nature of various kinds of tailoring used in the definition and development of software process descriptions. Techniques appropriate to each type of tailoring are then discussed. The general approach utilizes and builds upon the Software Process Framework, whose purpose is to provide guidance for designing, analyzing, and reviewing software processes for consistency with the SW-CMM.

Small projects

Q. One of our major issues that continues to prevent us from reaching level 2: the degree of flexibility we may or may not allow for our very small projects. Small projects can be those that last one or two weeks and only have one or two people working on them.

Specifically, for the very small projects, how much of the level 2 key practices within a key process area can be waived? Taking the question a step further, can any KPAs, such as SQA, be waived for our very small projects? We feel that it would be very near impossible to have our small projects deliver a full project plan, risk analysis, SQA plan, SCM plan, test plans for all levels of testing, and other similar such activities normally provided by the larger projects.

A. Rather than saying waived, I'd say have radically different implementations. See the first CMM newsletter on small projects. Certainly any key practice is subject to radically different implementation in very small projects, but the general concerns of each of the KPAs still need to be addressed, albeit at a different level. Look at Watts Humphrey's new book A DISCIPLINE FOR SOFTWARE ENGINEERING which applies the concepts at the level of the individual professional for some thoughts on what the CMM ideas mean in the micro-project environment.

Q. Our objective of course is to fulfill the intent of a CMM level two organization. As a possible solution, if we were to outline minimized guidelines for our small projects that meet the intent of the level 2 KPAs, AND all small projects were required to follow those guidelines, would an SEI assessment accept such an approach as satisfying the intent of level 2? By minimized guidelines, for example, instead of a full project plan (using MS Project for example), would a simple ten item task list suffice? Could formal estimating be waived or replaced by a much simpler hours estimate provided by a programmer?

A. Sounds pretty reasonable to me. The intent is to have a process that is documented, consistently implemented, and a foundation for improvement. You're the best judge of what a reasonable process in your environment is.

Q. I am in the middle of a software improvement project for the information systems groups in our company. Our IS group would be considered small in the extreme by CMM standards (less than 35 people) so we are having to do some extensive interpretation of the key practices document.

What I am finding is the "Key Practices of the CMM" has the information that is needed but we are finding it very difficult to translate that into solid policies and procedures. This problem has been amplified by the fact that none of the members of the SEPG group including me are professional policy authors. This has put into motion two courses of action that I am responsible for. The first, I have started to search for books, white papers, or organizations that have for sale their version of CMM approved policies and/or procedures that could be tailored in their wording to meet our requirements and still meet CMM requirements. The second, I am looking to find information on CMM approved or at least CMM centric implementation methodologies that can be purchased. Some that we have and are still evaluating are "The Guide" by The Guide associates, "Perform" by Cap Gemini Sogeti, and "Navigator" by Ernst &

Young.

A. The CMM has been successfully used by organizations as small as 20 folks before, so size should not be a major problem - you just have to apply your judgment as to what is appropriate in your area, given the guidance supplied by the CMM. Note that the CMM has no "shall" statements. There is nothing that is mandatory in the sense of a formal standard.

Policy statements are issued to set expectations on how a process should be performed within the organization. You don't need to be a professional policy author to say "we expect to do CM on every project - this includes identifying configuration items, doing change control according to procedures X, Y, or Z, auditing baselines by procedure X, base-lining according to the criteria in standard Y, etc."

You may or may not call out specific standards and procedures - depends on how often you think they will be updated. You should identify training, procedures, and standards that will help folks implement the policies that you're establishing.

It's not really complex. It's just hard to do consistently.

As far as off-the-shelf policies and tools are concerned, look at the IEEE and ISO standards. They provide a lot of guidance. You can "buy policies", but in the long run you'll need to tailor them to the needs of your business. You may do better to write your own. There are several organizations selling such documents - rather than recommending any specifically, I suggest you try to get a copy of the exhibitors from the SEPG National Conference and the SEI Software Engineering Symposium (contact customer-relations@sei.cmu.edu) and check them out. Similarly for tools, although the case for automated support, so long as it's aligned with your existing processes, is a lot cleaner.

Q. What is the definition of a smaller project? Is it based on: - the planned duration of the project (i.e. less than 6 months)? - the estimated size of the software (i.e. less than 1000 L.O.C)? - the number of software development engineers assigned to the project (i.e. less than 5 engineers)? 2. Would there be any tailoring of the CMM requirements for software development projects of the following nature: - goal is proof of concept, experimentation, demonstration, or prototype; - no external customer, in-house use only.

Have the guidelines governing smaller projects been formalized in a tailored CMM or other publication? If no formal guidelines are available, what generally accepted means have been applied by other companies to deal with these kinds of situations?

A. Please note that in rating processes, the goals of the key process areas are the normative component of the CMM. We believe that they are sufficiently general to be applied in any size organization or project, although implementations may be radically different. The key practices, subpractices, examples, and elaboration indicate what we would typically expect to see in a large, contracting organization addressing those goals (although they are also useful suggestions in many other environments). The specifics of your questions are addressed in the tailoring report. See ftp://ftp.sei.cmu.edu/pub/documents/94.reports/tr24.94.ps

Q. The approach I have led in my organization is to use the key practices for each level 2 KPA as a guideline. Our organization has evaluated each key practice and made a conscious choice as to whether we will make certain that our actual implemented practices satisfy that key practice. In almost all cases, we felt that including the intent of each key practice in our practices was beneficial. We have chosen to reject some key practices, but in each case it was due to a belief that it was inappropriate for our specific business practices. I primarily based this effort on a quote you forwarded once before:

The relevant point, as Charlie Weber has pointed out, is whether you are tailoring a top-level key practice or a sub-practice. To tailor a goal, you should sweat blood (and then be conservative in mapping the relationship to "the" CMM). To tailor a key practice, you should just sweat. To tailor a sub-practice, your conscience should hardly bother you.

Are we doing the right thing, or are we being far too rigorous? Are we taking too strict of a view of the CMM?

A. I think you're doing absolutely the right thing. You're applying intelligence! Finding the value! And I certainly want to encourage that!

At the same time, you cannot rate at the practice level without risking some severe usability issues. CMM ratings are very conservative in terms of what is required, and the application of professional judgement (as you are evidencing here) is critical to interpreting the "right process" for your business environment. When we release v2, I hope we'll fix this problem.

Common threads in the CMM

Q. I'm doing some work on an integration strategy/mapping of the CMM and Malcolm Baldrige evaluation criteria. I am interested in knowing if there are any "officially" or "unofficially" recognized common threads in the CMM (e.g., Measurement, Senior Leadership Commitment & Sponsorship, Education & Training, etc.). I have reviewed the Bell Canada TRILLIUM model, which does a good job of merging the concepts of "maturity levels" with what they call "roadmaps". They have identified 28 common threads, which is more than the average person can deal with. I'm looking for approximately 7 +/- 2 of the major threads in the CMM. I have my own opinions, but am interested in your thoughts (as the CMM authors) and the opinions of other experts from the CMM community.

P.S. Adding the concept of threads to the CMM might be worth considering for Version 2.0. It would significantly enhance the usability of the model (especially for strategic planning of SPI efforts). Just a thought.

From Mary Beth Chrissis:

In fact there are several themes that exist in the CMM. We have a module in the Intro course that discusses each of them. This is not an exclusive list of themes, but it does cover the major ones.

1. Continuous improvement -- The CMM focuses on defining processes that are mature. An attribute of a mature process is that it is improvable. If you think about the way the CMM and key process areas are defined, you will see a focus on continuous improvement. Maturity levels build upon each other and key process areas have a set of common features that help to insure that mature processes are defined.

2. Defined, documented, and used processes - You will see many practices that

address defined, documented, and used. These practices are especially prevalent in the Activities Performed common feature.

3. Commitment by senior management - This theme is contained primarily in the Commitment to Perform common feature. Policy statements are included in each key process area to address senior management sponsorship and commitment.

4. Stable processes - Processes must be stable and understood. Training helps to stabilize a process. Training practices are contained in the Ability to Perform common feature.

5. Measured processes - To objectively improve a process, it must be measured. Product measures are usually contained in the Activities Performed common feature. Measures of the process are contained in the Measurement and Analysis common feature.

6. Controlled processes - Processes need to be controlled to insure that they are being practiced as they are defined and documented. The Verifying Implementation common feature and the Software Quality Assurance key process area makes ensure that products and processes are verified and validated.

7. Process evolution - Processes evolve in the CMM. As processes improve, they will change over time. An example of this is the project management process. Level 1 organizations really depend upon the project manager as the primary means of project management. At level 2, Software Project Planning and Software Project Tracking and Oversight key process areas are put into place to provide basic project management processes for the projects. At level 3, Integrated Software Management takes the project management processes from level 2 and develops the project's defined software process based on the organization's standard software process. At level 4, now that the project has a defined software process, data can be used to manage the project. This is addressed by the Quantitative Process Management key process area. The project manager has an expectation of how the project should perform prior to the start of the project. At level 5, a project now has the proper foundation in place to fine tune the project management process. This is addressed by the Process Change Management key process area.

As you can see the common features provide the primary themes in the CMM. I hope this addresses your question or at least points you in the right direction.

Incremental development

Q. I think I have found a major weakness in the CMM. I'd like to hear your viewpoint on the matter. My CMM references are to SEI-91-TR-24.

The CMM does not emphasize incremental development of software systems. The term is not used in the CMM documentation as far as I know. The closest thing to "incremental development" that I find in the CMM is the mention of "serial build" in Level 2 Activity 5. The CMM seems to be neutral on the selection of the software life cycle. It requires "predefined stages of manageable size". But "stage" is ambiguous in this context. It could be taken to refer to a step in a process cycle rather than a complete increment that ends in test.<> Incremental development has been identified as the most important component of the Cleanroom method by the leader of (to my knowledge) the largest Cleanroom project ever completed. ("OS32 and the Cleanroom", Proc. 1st European Industrial Symp. Cleanroom Software Engineering, 1993).

Terry Baker identified incremental development as the most important component of structured programming for large systems. (I don't know where the widespread belief that hierarchical, "go-to-less" module construction is the most important component of structured programming got started.) ("Structured Programming in a Production Programming Environment", IEEE Transactions on Software, Vol. SE-1, No. 2 1975, p. 105) Baker used the term "top-down development" rather than "incremental development". In its historical context, "top-down development" is an ambiguous term. In the 1970s, Baker and Harlan Mills tended to use the term "top-down development" for what was, in reality, an evolving concept. Its meaning evolved from layered step-wise refinement to incremental development during the 1970s. But, the context provided in Baker's paper makes it clear that he was referring to the incremental development of new software systems.

Also, incremental development (under the name "the Milestone process") is apparently a key practice at Microsoft. (IEEE Software, Jan. 1995, p. 111).

Ironically, over a decade ago the proponents of incremental development identified two impediments to its widespread use: (1) it is often precluded by contract regulations and (2) it is often precluded by written development procedures, just the sort of regulations and procedures that the CMM is designed to influence. The Baker paper (cited earlier) describes ways to achieve incremental development in spite of government and commercial contracts regulations. Yourdon ("Top-Down Design and Testing" 1979, in "Software Design Strategies", Bergland, G. (editor), IEEE Computer Society Press, 1981, p. 60) found that in the organizations that had formal test procedures, those procedures commonly precluded incremental development.

CMM Activity 5 Level 2 calls for selecting an appropriate life cycle, but Yourdon and Baker have found that documents like the CMM have historically tended to obstruct the selection of the most appropriate life cycle model for many systems.

Incremental development potentiates process control by increasing the number of process cycles and the number of early opportunities to sample observed process capabilities. Incremental development potentiates defect prevention by allowing process defects to be identified in the early process cycles and eliminated.

The CMM calls for a incremental approach to process improvement, but it does not encourage an incremental approach to software development.

Overall, I am a supporter of the CMM. But the omission of a single key practice could greatly impact results.

What's your viewpoint on this matter?

A. I agree that software should be developed using some kind of evolutionary / incremental build / spiral life cycle. I don't think waterfall is a very good life cycle model for modern software projects.

The CMM does not, in general, prescribe how to solve specific problems. As you observe, all it says is "pick a life cycle and use it." This is one example of a general class of issues, where we have chosen to focus on what rather than how. There are cases were waterfall is the best life cycle. We do not want the CMM to be overly prescriptive.

If you write this up as a change request, it will be formally reviewed, tracked, and dispositioned. I would be interested in seeing the comments on such a proposed change, because my own natural tendency agrees with you. Once we start selecting how software projects should work, however, we're starting down a slippery path. What about replacing testing/peer reviews with formal methods? What about specifying inspections rather than the more general peer reviews? Why don't we recommend OOD? Should all projects use QFD?

We do rely, to large degree, on organizations and projects making reasonable decisions if they have a process that causes them to make the decision consciously. That is a risky assumption in many ways, but I'm more comfortable with under-specification than over-specification when it comes to an industry-wide "standard" such as the CMM.

Incremental process improvement

Q. In the book Managing the Software Process, page 87, Section 6.2.1, regarding goals and objectives, it states that one of the first rules of thumb is to implement the product in small incremental steps, and to select each increment to support succeeding increments. I have chosen SEI Level 2 as my first increment, and therefore I am currently trying to clearly understand the difference between SEI Level 2 and SEI Level 3.

A. Actually, moving from Level 1 to Level 2 is a pretty big step. You may want to set smaller goals. The CMM does not say what those should be; it may differ based on your business environment. Empirically, the last KPAs that are mastered are Requirements Management, Software Project Planning, and SQA. Emphasis on MASTERED. When you do an assessment, the problems facing your organization should fairly obvious. The challenge is in prioritizing which ones to tackle first; the maturity levels give some good guidance that will help.

Legacy systems and maintenance documentation

Q. How would you handle a situation where a large proportion of the organization's work load is legacy stuff and will continue to be so for the next couple of years? They have defined a life cycle model which essentially skips the high level design phase; thus they cannot answer yes to the PMM questions asking for traceability between top level design and requirements (2.4.8) or detailed design (2.4.11). They have other work which does follow "V" or Spiral SLCMs, but that is only about 35% of their work load and would be represented by at most three of the five PLs. [My opinion is that they should take the hit with No answers and hope that if 3 of 5 answers are Yes and they otherwise qualify for Level 3, the team would assess them at 3. In addition I'd produce a finding and a recommendation re creation of some re-engineered top level design document which could then be referenced in the traceability chains.]

A. This is almost the exact opposite of what I found useful when I was programming. I usually found that the detailed design was wrong and/or obsolete, but an accurate architecture, plus the code, let me understand what was going on in the system (after some study). I would agree with stating that, in this context, we don't do practices X, Y, or Z. I would probably argue (if I felt it was true), that rather than answering NO, they should be answered NOT APPLICABLE. You are definitely in a gray area here, where judgment is going to be crucial in scoring KPA satisfaction and level achievement. I would certainly ask the staff if they were having problems that were caused by the lack of high-level design info, concentrating especially on new people coming in (training issue). I would also write a finding that this was a potential problem and outline the pros and cons in doing something about it - both from a CMM scoring perspective and a maintaining the system perspective.

Software project dynamics

Q. Is there a COST ESTIMATION model for CMM ? I mean, if I am an organization at CMM level (x), how can I estimate the cost (in person months, not dollars) of moving to level (x+1), x less than 5.

I have read SEI-94-TR-12, and familiar with SEI-93-TR-24 and 25.

A. There is not a model, per se, but there has been some work in "software project dynamics" of modeling organizations at different maturity levels based on different assumptions. Herb Krasner and Stan Rifkin have done some work in this area. There's also a paper in American Programmer, Sept 1994, by Rubin, Johnson, and Yourdon on the topic.

Not Applicable versus risk

Q. Another question is about the assessment of KPAs. If we have developed a written policy and a documented process for a certain KPA, e.g., Subcontractor Management, but the projects to be assessed do not have subcontractors, can we be considered to achieving that KPA by assessment?

A. Yes, but it might be more appropriate to score SSM as Not Applicable or to explicitly identify the risk if a new project will be doing extensive subcontracting, since there the implementation has not been successfully demonstrated yet.

Discipline versus bureaucracy

Q. I have read CMM v1.1 carefully three times and discussed it with a group of my colleagues. I have the following observation. In an informal development environment such as the one here, the word "discipline" is easily and naturally read as "bureaucracy" or "rigidity". Personally, I believe that given a project's complexity and criticality, there is an appropriate level of formality which will optimize results.

The goal of CMM "Repeatable" Level seems to be to raise formality in order to conquer problems of complexity. In other words, it seems to concern itself solely with inappropriately LOW levels of formality. In "Managing the Software Process", Watts Humphrey says something like "No bureaucracy can ever produce truly great products.". Many of us have worked at one time or another in bureaucratic organizations which seem to accomplish little real work, due to what looks like an excess of process. What mechanisms exist in the CMM to reduce inappropriately high levels of formality in a process?

In particular, how does the Level 1 organization use the CMM to find the right degree of formality and not overshoot that goal? Is this a possible area for future development of the model?

A. The only mechanism in the CMM that addresses your concern is the hierarchy of the practices. There are no "shall" statements in the CMM, but to satisfy a key process area you have to satisfy each of the goals; to satisfy a maturity level you have to satisfy each of its KPAs.

The KPAs and goals are very abstract, and I would argue that they apply in any size or type of software project (with the exception of Software Subcontract Management if you aren't doing subcontracting). As you move down in detail, e.g., the subpractices and examples, you do move into a description of the normal behaviors we would expect to see in large-scale, government contracting kinds of projects.

We explicitly state that you have to apply professional judgment in appropriately applying the CMM as you move away from that specific context. Alternate implementations are quite possible, and their adequacy should always be considered with an open mind.

Because the CMM says what rather than how, you have to decide what the appropriate level of formality is in your business environment. We can't give generic advice that would apply to every conceivable user of the CMM beyond "apply intelligence." That may seem smug, but it's very hard to provide objective criteria for process standards in the software field (as Norm Fenton as pointed out).

When doing internal process improvement, this advice may be unsatisfactory. When you're concerned with external evaluations, a la SCEs, it can be even more difficult. The necessity is to be able to demonstrate to a neutral (in some cases, perhaps even adversarial) party that you have satisfactorily addressed a KPA's concerns. This leads to potential disagreements, i.e., reliability and consistency of evaluation issues, but in the last 7 years we have been unable to identify a better compromise position.

So the bottom line is, it's your responsibility to use the CMM as guidance for improving your process/evaluating contractors. You have to decide the appropriate degree of formality and rigor within your business environment. Your customers, who are part of your business environment, should share your "self-evaluation".

COTS

Q. How is the CMM applied to COTS software integration projects?

A. As appropriate :-) Certainly there are implementation concerns for COTS, but the fundamental principles of planning, managing, communicating, etc., as described in the goals of the KPAs would certainly apply - just ask yourself what is a reasonable implementation of this practice or process for this project, and you won't go far wrong.

Q. How is the CMM applied to Professional Services projects?

A. Ditto.

Required overtime Q. One of the tenets of CMM (as I understand it) is that as an organization moves up the levels things are going to work better and the organization will have "skilled, happy employees". With the downsizing we are seeing a very different

picture. The last major job to be awarded was to XXX. When the award was announced it included 8 hours of uncompensated overtime per week per employee. So folks were immediately working 6 days a week for the same pay they had been receiving for 5 days! Happy campers they aren't. If a contractor had included this in a bid several years ago the Source Selection Board would have thrown out the proposal. This time they not only took it they encouraged it. We are now in the same mode for a much larger effort to support YYY. There are a lot of very unsure software people here. It is obvious this procurement may go the same way as the last one. Some of these people are currently in CMM self assessments. They are having a tough time matching the CMM material with the world they live in......

We have met the enemy and it is all of us! (apologies to Pogo)

A. That's pretty terrible. I'm afraid that organizations following this "improvement" tack are ignoring the human side of process improvement that's critical to continuous improvement. This sounds like just another way of going to the lowest bidder, but using SPI as an excuse to justify the low-ball. If this is more than a temporary, one-shot deal then the best people will leave for greener pastures - and they really will be greener. If you want me to work round the clock, you better offer to make me a millionaire so I can retire in a few years - but I doubt that's your environment ;-)

I think I'd identify this as a finding in my next assessment. Also, I have some neat comics that describe this kind of "improvement" in a graphic way. If you're around the SEI anytime I'm here, drop by and see them.

Methodologies

Q. What impact/effect do methodologies such as Yourdon/DeMarco and TIs IEM have on the CMM effort?

A. Essentially none. In the CMM we basically say "pick a methodology that works well for you." The CMM does not recommend any specific methodology, and I don't believe that we have anything in the CMM inspired by a particular methodology that is generally valuable to all software efforts.

Q. How do "tools" such as TIs IEF, Finkelstein's IE Advantage and ADW (to name a few) affect this process? Do these products shorten the cycle, are they an important part of the process or does SEI recommend its own tools?

A. We do not recommend any specific tools. In general, tools make you more productive, increase quality, decrease cycle time, etc., when they are integrated into a well-defined process. If they aren't integrated, they become shelfware. Technology transition of tools can be tricky; one needs to be sensitive to process and cultural impacts of change. No one, not even the SEI :-), can say "this is the right tool/methodology that will solve all of the software community's problems." (Even if there was a technically superior answer, you'd still have to consider the organization's processes and culture!) SEI is not a tool vendor and we do not endorse any specific tools or methods.

Regular versus periodic

Q. We were going through compliance requirements attempting to make them more CMM-like in style and kept running into the word "regular" and

"periodically." So it occurred to us that perhaps the CMM used one term consistently to convey the idea. So we did a key word search and discovered that the word "regular" occurs 15 times in the CMM and the word "periodic" occurs 40 times.

A. There's no significant difference between regular and periodic, as far as I recall. In fact, we tried to go to "periodic" as the normal word; I don't remember if "regular" appears for good reason or just because we didn't catch it in the editing process. I did check that it's only in subpractices. The other significant term is "event-driven", so things can be either periodic or event-driven.

Q. So our question is what, if any, difference does it make which term is used?

A. Periodic is a little bit stronger in terms of "weekly" or "monthly" kinds of implication. I'd probably pick one that I was comfortable with (perhaps a bias to periodic) and use it.

QUESTIONS

Return to top of the page

Return to main page

Questions and Answers on the CMM

Mark C. Paulk
Issue #4
7 April 1997

I am frequently asked questions on how to interpret the CMM in various contexts. The following "newsletter" may be of general use. I have to caveat these answers by saying that they represent my opinion and do not represent official SEI positions. They have typically not been internally reviewed, but they do represent a well-informed opinion on what the CMM means. The Q&A has been sanitized for proper names (where appropriate), but it has not been cleaned up for publication -- it's pretty much the off-the-cuff reply that folks have gotten to their questions.

You are welcome to ask questions on the CMM. I do not guarantee quick answers, although I'll try to get back to you as soon as possible. Sometimes that's the same day I receive a question, sometimes it may be a month, depending on my travel schedule and work load. I prefer to receive questions by e-mail. I will lightly edit and sanitize question and answer before distribution in this newsletter.

Mark Paulk
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
Fax #: (412) 268-5758
Internet: mcp@sei.cmu.edu

The questions in this issue are fairly general. Topics covered include:

TQM and CMM

I would like to know what you think the relationships are between TQM and its philosophies and the CMM. ... I have been reading more and more articles lately stating the philosophies of Deming, Juran and Crosby are the basis for the CMM. I also am seeing more articles, similar to yours, merging ISO 9000 and the CMM. I do not see the connection to the TQM founders.

It seems to me a company could be ISO 9000 and CMM level 4 and be creating cement rain coats. They may be the highest quality cement rain coats ever manufactured. They may be manufactured with very mature processes. The company may be following all their processes definitions right from the ISO 9000 guidelines. But what good is it if no customers want cement rain coats.

TQM is customer focused. It has as its foundation the needs of society. Deming's point is that healthy companies with healthy employees are necessary for the survival of a society. What he documented with his 14 points and his 7 deadly diseases has little to do with the CMM. Maybe one is about the why and the other is about the how.

The precepts and practices of TQM are essential.

The precepts and practices of the CMM are essential.

ISO 9000 certification seems to be becoming another essential.

These are all complimentary. They are also synergistic. They have some common points, like continuous improvement, but that does not mean they have come from the same place or are meant to do the same things. Good, logical, and practical solutions to problems such as what TQM and the CMM are attempting to solve will overlap.

I am not very far along in this journey. I am very likely dead wrong.

What do you think the relationships are between TQM and its philosophies and the CMM?

CMM only attacks the process side of TQM, and specifically for software engineering, although the principles can be applied to any (engineering) discipline.

There are some very important aspects of TQM that are deliberately not addressed in the CMM. What value, for example, would the SEI add to a discussion of the "people" issues? Is there anything that's software-specific about "people"?

The obvious answer is no, yet we can add value in the software process arena, while at the same time we should recognize the importance of the people issues in enabling software process improvement.

As I said, we made a deliberate decision to focus on the area where we could make a major contribution. Things change over time...

The desire of some organizations to have a "People CMM" to help them build their human resource has led to the People CMM. The word software isn't mentioned a whole lot... and it was written by HR-knowledgeable folks.

We are drafting a chapter for CMM v2 on interpersonal skills and organizational change - not as key practices that describe maturity levels, but in the context of enabling process improvement and pointing to better sources of information, such as the P-CMM, Deming, etc. A lot of excellent work has been done by folks much more knowledgeable than we about interpersonal skills!

We also don't (currently) talk about strategic business planning or customer satisfaction explicitly, which are some of the other important TQM topics. We may do something about those in CMM v2 also, but we are the _software_ engineering institute and need to stay focused on our primary mission, while not ignoring the critical interdependencies - a difficult balance to maintain.

The CMM was inspired by Crosby's maturity grid, but it has evolved a lot over the past several years. If you'd like to track it's evolution, you could read:

Mark C. Paulk, "The Evolution of the SEI's Capability Maturity Model for Software," Software Process: Improvement and Practice, Vol. 1, Pilot Issue, Spring 1995, pp. 3-15.

Will it cost as much to produce a product when on level 1/2/3/4 as it will producing the equivalent product when on the level above? (Has any company really performed this experiment? A 2 by 2 matrix?)

The ones who have data have demonstrated to their satisfaction that it's cheaper and quicker to build products at a higher maturity level.

Or does the product automatically change when you produce it being on a different/higher level?

That's a business decision. You have added opportunities on the cost/functionality tradeoff curve.

Of course, the workers will give the product a higher subjective value. So does the management, hence the selling price is likely to be higher.

Is the cost (as perceived by the manager) to produce a product knowing how the total cost is distributed over all activities involved less than the perceived cost when nothing is known about the cost distribution?

Will you charge the same for a s/w product when you're on level 2/3/4/5 as you charged for an equivalent product (functionality perceived by the customer) when you were on the level below?

That's a business decision.

If the cost of developing s/w products really decreases as you move up the CMM ladder, how will you be looked upon by potential customers if you claim to sell/produce products with the same quality as your competitors notably cheaper?

Almost by definition, it won't be the same quality and cheaper. It will have to be higher quality and cheaper. You're forgetting rework costs. About 40% of software project costs are in fixing bugs to get something shippable. If appraisal/prevention permits you to build stuff for 30% less, by definition it's going to be a higher quality product - see TQM literature.

I've been told that most companies decide to produce cheap products or high quality products (read expensive). But there are some companies that try to sell/produce cheap high quality products (called in-betweens?). How do these in-betweens perform on the market compared to the other two?

Don't assume that all business decisions are consciously made ;-)

Also, don't assume that a Civic is a low-quality car just because it's cheap. It's a high quality car (remember fitness for use!) targeted to an inexpensive market. Quality /= luxury.

This is the classic misunderstanding of the quality movement. Before studying the software process improvement world, study the TQM world. Read Deming and Juran. SPI is an application of those more general concepts in the software world.

I think that levels 2 and 3 are quite different/separate from levels 4 and 5. My view is that levels 2 and 3 are parts of a larger goal, namely to stabilize the process(es). Ie, to minimize the variance/deviation/fluctuations.

Levels 4 and 5, on the other hand, deal more with adjusting "the level/height of the curve". This is just my opinion. Is it totally wrong? (Doesn't this imply that defect prevention is misplaced at level 5? I don't know enough of the theory. I have lots and lots of reading to do...)

Think of it as qualitative vs quantitative process control then process improvement - which gives you 4 levels above level 1.

This means that you disagree with the basic principle of TQM. While I don't particularly mind, it's a very old argument - that quality costs - that's been debated by far abler folks than I.

I don't disagree. But, I think (it's just an uneducated opinion) it's reasonable to assume that the levels 2 and 3 block will give a better "ROI" than t he upper block. Why? Because it is probably easier and cheaper to remove the first errors.

Pick the low hanging fruit, as we say in the States.

Again, this goes back to the old "quality costs" debate. We're seeing so me of this resurface under "good enough software". I have no problem with this concept so long as you realize this is a "fitness for use" debate; when i t becomes operationalized as "whatever junk we can get a customer to buy", I think the market will determine pretty effectively.

I think I'm getting your point, but I'm not sure. Please, could you elaborate e this a bit? I am having trouble remembering all the stuff all the time.

Analogy: GM said they built good cars and could point to domination of market in 50's through early 70's as proof that they understood what was good enough. The environment changed... and the companies that had focused on "quality" made enormous inroads...

Do you think that all SE developers should try to achieve level 5? And, do you think that they all have go "all the way", or do you think that companies like Microsoft necessarily don't have to go as far as companies like LORAL Houston. Ie, can improvement/changes be overdone?

Go as far as necessary. If you're level 3, you're world class today. What the competition may inspire in the future, in terms of business reasons for continuing to improve and how you improve, will determine your strategy later.

The ones who have data have demonstrated to their satisfaction that it's cheaper and quicker to build products at a higher maturity level.

It is my opinion that there isn't conclusive proof, yet (At least I think I haven't seen any proof). Intuitively, the CMM has to work! That is, I hope and think that quality is free (maybe I should have said this in my first e-mail...). But, what I'm not sure about is that "state-of-practice" quality is free. [You have to put this into my context: in Sweden people cannot sue for astronomical amounts of money :-)]

The main problems are that I don't know how they came to the results. How do they define/measure/calculate productivity for example? And, how do they define/measure/calculate the cost and the return on changes that supposedly affect the productivity (reprise)?

They defined it based on what they considered important in their environment - which is always good advice. Their papers provide some details, but they're all different.

Organizational analysis

Responding to a customer's request, here is the situation:

Large corporation with 6 sites around the country. The corporation has 8 major systems; one at each site plus two sites with two each. The major systems make up ~60% of the corporation's business. The corporation has a SPI effort that encompasses all 6 sites.

Their assessment plan is to assess a major system at each site (i.e, assess 6 representative systems of the corporation). Assume: the assessment result is that the 6 systems assessed are level 2. What can the corporation say:

a) That the corporation is a level 2 organization.

Possible, but I'm not really comfortable with it as stated. This isn't just a sampling issue, they're really looking at six different sites with what is presumably their best and brightest.

Are these development efforts? Maintenance projects? Operations?

What about the 2 missing major systems? Will this skew the perception of the process? You can do ML2 diagnosis on individual projects without worrying about organizational infrastructure, but if you're skipping 2 major systems, why?

b) That the major systems of the corporation operate at level 2.

It isn't the systems, it's the projects or sites that operate at ML2.

c) Nothing about being level 2 [if you say c), please explain your reasoning some].

They're violating a bunch of the organizational analysis heuristics, I think. They're basically claiming the organization is the whole company, in spite of geographical dispersion, possible major cultural differences, possible deployment issues, etc. Who is the sponsoring manager of the SPI effort? Are they directly funding SPI efforts at each site? Is there sponsorship physically located at each site?

Major qualms about saying "we're ML2" in this kind of context. Certainly if we "endorse" their statement, we should know a lot more.

Would you say something different if you were the lead assessor? i.e., what would say about the results when reporting them (assume: you can't change the assessment scenario, but feel free to add comments on how you would change it).

Tough question. This is also sampling issue, similar to 1 big project + bunch of small ones. Issues of consistency and predictability can be significantly skewed by outliers, whether more or less mature. What is the definition of "typical project" for company?

I'd go for option C unless I knew a lot more about the assessment. For political reasons, that may not be viable, but the existence of the political reasons influences me to believe the organization is likely to be ML1 with ML2 on some critical projects that get a lot of management attention.

Most of the CMM seems to deal with a single development organization, but many companies have dozens of development organizations at a site, and try to get an SE I rating for the entire site. This leaves a lot of questions unanswered.

My general comment on this is that it's a bad idea to overload organizations like you're suggesting happens. I know that it happens also. Depending on why you're going for an SEI rating, this can have very bad consequences: 1) ineffective improvement, 2) an SCE team may not make the same mistake (it's one of the reasons determined for some of the maturity level mismatches between SCEs and SPAs several years ago), 3) going for a score rather than better business value. Problems, any way you look at it.

However, I would comment that "dozens" seems like an exaggeration. "Several" seems a better term and "few" might be even more appropriate. "One" is, I agree, rare for sizable companies. Organizational analysis is one of the least well-defined aspects of doing appraisals.

For example, During the analysis of data (step 3, Response Analysis, in your figure 4.1) how do you handle an organization that "falls down" on key process areas in one group, and scores well on those same KPAs in another group?

The organization "fails." The weakest link determines organizational maturity. There are exceptions, e.g., legacy systems, but inconsistency of deployment is one of the hallmarks of a level 1 organization. Every company (that survives) has pockets of excellence. Organizational learning implies a lot more! In particular, predictable implementation of a consistent process unless there are "external drivers" otherwise.

In some organizations that I have worked for, they gave the full score because the y felt that the capability existed somewhere in the organization.

There's an optimist in every crowd... rating processes is hard, but this is clearly over the line of acceptable judgment.

In other places, I've seen them apply a weighting based on the amount of software produced by each group.

I don't think that would work very well either. Is a small amount of life-critical software less important than a large amount of MIS software? I don't think so... that's one of the problems with algorithmic approaches -- and why I prefer heuristics that warn an assessment team of the kinds of issues to be sensitive too, but leaves the responsibility to the team to apply reasonable professional judgment.

Personally, I feel that the purpose of the CMM/SCE is to identify and prioritize areas needing improvement, and therefore the areas needing improvement need to be pointed out as having geographical or other inconsistencies, and the scoring needs to be based on the WORST rather than the best groups to place the proper management influence on the problem areas.

Seems reasonable to me ;-)

However, no one (in Intel or Motorola for examples) would ever do it that way because it reflects badly on them. I believe that this is because of a violation o f the spirit of the SEI, and of Deming's beliefs (that metrics should never be used to rate people, just the process).

I'll pass this along to some of our CBA IPI and SCE folks. I agree with you in concept, although I don't know enough about your specific examples to judge. Some of the toughest issues in assessment include:

- organizational analysis (what is the scope of an appraisal?)
- project sampling (what is a representative project? how many are needed in appraisal?
- what are the "exceptions to the rule" that need to be thoughtfully considered re their impact on process and organizational capability? E.g., legacy systems, very small projects (tasks), rapid prototyping projects,...

From James Hart:

There are (being somewhat realistic) a few requirements that must be met before an organization of that size and geographical separation can say they are at ML 2. For them to say they are at level 2, they must say this is true for ALL projects in the organization, with only the exclusion of outliers (e.g., short duration projects, 2-3 man efforts, those outside the normal business direction). In order for them to make this claim, then the projects must really be selected in a somewhat random way. Preselection of 4-5 projects based on criteria OTHER THAN PROCESS-RELATED will skew their results.

Second, you can only answer such questions as you ask in the context of why they want to do an assessment. If it is to validate their level and make public the results, they have potential problems with getting a REALITY-check. People will say and report what they think management wants (or needs) to hear. Since they wish to do a consolidated assessment (in essence, combining six assessments into one), then it does not sound like they are interested primarily in identifying key weaknesses.

Finally, projects selected for assessments only make up a major part of who is talked to on an assessment. During interview sessions with personnel, you will want to select those OUTSIDE of the projects selected. Their inputs will effect the results, so you can't really say the assessments will only cover their major systems. 60% of the business may not be 60% of the people or 60% coverage of project processes.

In summary, I would personally not hold much faith in results of the kind suggested. From experience, I have one data point: an organization wanted to do an assessment over a very broad range of software developers. We did as they wanted; but they were not happy with the results. The reason was due to diversity in processes across the groups; each of the resulting findings were issues for a PORTION of the assessment's scope, and NONE were REPRESENTATIVE of issues the ORGANIZATION faced as a whole. They subsequently sent 4-6 months following the assessment weeding through each of the findings to determine what portions of the organization needed to address what findings. Consolidating results across diverse areas leads to LESS meaningful data.

A related point raised by Balzer: at L4-5, it may not be appropriate to focus on specific practices these high-maturity companies use, but rather at the meta-level, what it is they do with process capability. Bob's thesis (which I believe has merit and which is consistent with a conversation we had with Curtis 3 years ago) is that as organizations mature, their processes more-and-more reflect their product and product family architectures and their important quality features - and these in turn drive the specific features of interest in measuring the capability (my paraphrase), so there may not be a lot of commonality in what is done at the detailed practice level, but more likely at the level of what they do with capability. This suggests the real enhancements at L4-5 may not be so much in additional KPAs, but rather:

1) achieving the right level of detail on practices determining, measuring, and manipulating process capability,

2) better examples (reflective of different product domains, etc.), and
3) making stronger meta-connects with business goals and needs.

If the customer won't pay

Here's a new one. Can an organization be rated Level 3 if a project does not comply with Level 2 or 3 KPAs if their customer has specifically said he doesn't want to pay for them? This is in a defense, contract based environment.

Yes, if it's a rare occurrence, but I would want to see a directive from the customer that they refuse this functionality and accept the risk that it entails. More generally, I would expect a company to "do the right thing" even if the customer says don't do this -- to the extent that a supplier would charge MORE for not doing SCM than for doing it (after all, the reasons for the KPAs are driven by business concerns).

Tailoring

We have been trying to understand the applicability of the CMM to short cycle time software support activities. We concluded that applying the CMM in all its rigor will add to the cycle time of the support activity and possible customer dissatisfaction.

I am now thinking of formulating a support maturity model with the same five levels as the CMM but with different key practices and KPA's.

Are there some guidelines to selecting KPA's and KP's for a maturity model? Is there a generic process for this? Are there any references available? Any light you can throw on this will be deeply appreciated.

A report on tailoring the CMM is going through the approval cycle now. It's by Mark Ginsberg. Check with our customer relations people in January (customer-relations@sei.cmu.edu) to see if it's been released and/or check the anonymous ftp site for it (ftp.sei.cmu.edu, pub/cmm or pub/documents).

I would disagree with the idea of having different KPAs and claiming it's the same maturity levels. I would strongly argue that all of the KPAs, with the exception of subcontract management, apply in any software development or maintenance environment. The implementation may differ dramatically, and so arguing that different key practices may needed may be appropriate, but organizations that have tailored the CMM for small projects observed that 90%+ of the key practices carried forward into that environment. Where you really get large project/organization specific is in the subpractices. What is important is mapping the concepts and roles of your environment to the concepts and roles in the CMM. Once you establish that relationship, then analyze the applicability of the practices.

I would also refer you to the first newsletter on tailoring the CMM.

I am an SQA member of the SEPG at ZZZ. Recently we have been attempting to institute Level 2 process on smaller projects. I was reading through the CMM v1.1 (Feb '93) document and came across the following paragraph:

The near-term focus on CMM development activities will be oriented towards tailored versions of the CMM, such as a CMM for small projects and/or small organizations. CMM v1.1 is expressed in terms of the normative practices of large, government contracting organizations, and these practices must be tailored to the needs of organizations that differ from this template.

Has a tailored version of the CMM addressing smaller projects been developed? If not, has this issue been addressed to some degree in other publications? How can we obtain such information?

We held a workshop on this topic a couple of years ago and concluded that there was too much variation between small projects and organizations to justify a CMM tailored specifically for the "small" environment. The challenges were shared, in general, by "non-small" environments. A resident affiliate, Mark Ginsberg, had come to the SEI to work on a CMM/small, but this changed the direction more to tailoring in general. The ultimate result was

Mark Ginsberg and Lauren Quinn, "Process Tailoring and the Software Capability Maturity Model," Software Engineering Institute, CMU/SEI-94-TR-024, November 1995.

The Software Capability Maturity Model(sm) (SW-CMM[sm]) is serving as the foundation for a major portion of the process improvement being undertaken in the software industry. It is composed of two volumes: the Capability Maturity Model for Software and the Key Practices of the Capability Maturity Model. The key practices of the SW-CMM are expressed in terms that reflect normal practices of organizations that work on large, government contracts. There is, however, a significant population of software-producing and -acquiring organizations, operating in different environments, for which the key practices require significant interpretation and/or tailoring, prior to application. This report presents a tailoring framework that identifies process artifacts, tailoring processes, and their relationships to project artifacts, and explores the nature of various kinds of tailoring used in the definition and development of software process descriptions. Techniques appropriate to each type of tailoring are then discussed. The general approach utilizes and builds upon the Software Process Framework, whose purpose is to provide guidance for designing, analyzing, and reviewing software processes for consistency with the SW-CMM.

Small projects

One of our major issues that continues to prevent us from reaching level 2: the degree of flexibility we may or may not allow for our very small projects. Small projects can be those that last one or two weeks and only have one or two people working on them.

Specifically, for the very small projects, how much of the level 2 key practices within a key process area can be waived? Taking the question a step further, can any KPAs, such as SQA, be waived for our very small projects? We feel that it would be very near impossible to have our small projects deliver a full project plan, risk analysis, SQA plan, SCM plan, test plans for all levels of testing, and other similar such activities normally provided by the larger projects.

Rather than saying waived, I'd say have radically different implementations. See the first CMM newsletter on small projects. Certainly any key practice is subject to radically different implementation in very small projects, but the general concerns of each of the KPAs still need to be addressed, albeit at a different level. Look at Watts Humphrey's new book A DISCIPLINE FOR SOFTWARE ENGINEERING which applies the concepts at the level of the individual professional for some thoughts on what the CMM ideas mean in the micro-project environment.

Our objective of course is to fulfill the intent of a CMM level two organization. As a possible solution, if we were to outline minimized guidelines for our small projects that meet the intent of the level 2 KPAs, AND all small projects were required to follow those guidelines, would an SEI assessment accept such an approach as satisfying the intent of level 2? By minimized guidelines, for example, instead of a full project plan (using MS Project for example), would a simple ten item task list suffice? Could formal estimating be waived or replaced by a much simpler hours estimate provided by a programmer?

Sounds pretty reasonable to me. The intent is to have a process that is documented, consistently implemented, and a foundation for improvement. You're the best judge of what a reasonable process in your environment is.

I am in the middle of a software improvement project for the information systems groups in our company. Our IS group would be considered small in the extreme by CMM standards (less than 35 people) so we are having to do some extensive interpretation of the key practices document.

What I am finding is the "Key Practices of the CMM" has the information that is needed but we are finding it very difficult to translate that into solid policies and procedures. This problem has been amplified by the fact that none of the members of the SEPG group including me are professional policy authors. This has put into motion two courses of action that I am responsible for. The first, I have started to search for books, white papers, or organizations that have for sale their version of CMM approved policies and/or procedures that could be tailored in their wording to meet our requirements and still meet CMM requirements. The second, I am looking to find information on CMM *approved or at least CMM centric implementation methodologies that can be purchased. Some that we have and are still evaluating are "The Guide" by The Guide associates, "Perform" by Cap Gemini Sogeti, and "Navigator" by Ernst & Young.*

The CMM has been successfully used by organizations as small as 20 folks before, so size should not be a major problem - you just have to apply your judgment as to what is appropriate in your area, given the guidance supplied by the CMM. Note that the CMM has no "shall" statements. There is nothing that is mandatory in the sense of a formal standard.

Policy statements are issued to set expectations on how a process should be performed within the organization. You don't need to be a professional policy author to say "we expect to do CM on every project - this includes identifying configuration items, doing change control according to procedures X, Y, or Z, auditing baselines by procedure X, baselining according to the criteria in standard Y, etc."

You may or may not call out specific standards and procedures - depends on how often you think they will be updated. You should identify training, procedures, and standards that will help folks implement the policies that you're establishing.

It's not really complex. It's just hard to do consistently.

As far as off-the-shelf policies and tools are concerned, look at the IEEE and ISO standards. They provide a lot of guidance. You can "buy policies", but in the long run you'll need to tailor them to the needs of your business. You may do better to write your own. There are several organizations selling such documents - rather than recommending any specifically, I suggest you try to get a copy of the exhibitors from the SEPG National Conference and the SEI Software Engineering Symposium (contact customer-relations@sei.cmu.edu) and check them out. Similarly for tools, although the case for automated support, so long as it's aligned with your existing processes, is a lot cleaner.

What is the definition of a smaller project? Is it based on: - the planned duration of the project (i.e. <6 months)? - the estimated size of the software (i.e. <1000 L.O.C)? - the number of software development engineers assigned to the project (i.e.<5 engineers)? 2. Would there be any tailoring of the CMM requirements for software development projects of the following nature: - goal is proof of concept, experimentation, demonstration, or prototype; - no external customer, in-house use only.

Have the guidelines governing smaller projects been formalized in a tailored CMM or other publication? If no formal guidelines are available, what generally accepted means have been applied by other companies to deal with these kinds of situations?

Please note that in rating processes, the goals of the key process areas are the normative component of the CMM. We believe that they are sufficiently general to be applied in any size organization or project, although implementations may be radically different. The key practices, subpractices, examples, and elaboration indicate what we would typically expect to see in a large, contracting organization addressing those goals (although they are also useful suggestions in many other environments). The specifics of your questions are addressed in the tailoring report. See
ftp://ftp.sei.cmu.edu/pub/documents/94.reports/tr24.94.ps

The approach I have led in my organization is to use the key practices for each level 2 KPA as a guideline. Our organization has evaluated each key practice and made a conscious choice as to whether we will make certain that our actual implemented practices satisfy that key practice. In almost all cases, we felt that including the intent of each key practice in our practices was beneficial. We have chosen to reject some key practices, but in each case it was due to a belief that it was inappropriate for our specific business practices. I primarily based this effort on a quote you forwarded once before:

The relevant point, as Charlie Weber has pointed out, is whether you are tailoring a top-level key practice or a subpractice. To tailor a goal, you should sweat blood (and then be conservative in mapping the relationship to "the" CMM). To tailor a key practice, you should just sweat. To tailor a subpractice, your conscience should hardly bother you.

Are we doing the right thing, or are we being far too rigorous? Are we taking too strict of a view of the CMM?

I think you're doing absolutely the right thing. You're applying intelligence! Finding the value! And I certainly want to encourage that!

At the same time, you cannot rate at the practice level without risking some severe usability issues. CMM ratings are very conservative in terms of what is required, and the application of professional judgement (as you are evidencing here) is critical to interpreting the "right process" for your business environment. When we release v2, I hope we'll fix this problem.

Common threads in the CMM

I'm doing some work on an integration strategy/mapping of the CMM and Malcolm Baldrige evaluation criteria. I am interested in knowing if there are any "officially" or "unofficially" recognized common threads in the CMM (e.g., Measurement, Senior Leadership Commitment & Sponsorship, Education & Training, etc.). I have reviewed the Bell Canada TRILLIUM model, which does a good job of merging the concepts of "maturity levels" with what they call "roadmaps". They have identified 28 common threads, which is more than the average person can deal with. I'm looking for approximately 7 +/- 2 of the major threads in the CMM. I have my own opinions, but *am interested in your thoughts (as the CMM authors) and the opinions of other experts from the CMM community.*

P.S. Adding the concept of threads to the CMM might be worth considering for Version 2.0. It would significantly enhance the usability of the model (especially for strategic planning of SPI efforts). Just a thought.

From Mary Beth Chrissis:

In fact there are several themes that exist in the CMM. We have a module in the Intro course that discusses each of them. This is not an exclusive list of themes, but it does cover the major ones.

1. Continuous improvement -- The CMM focuses on defining processes that are mature. An attribute of a mature process is that it is improvable. If you think about the way the CMM and key process areas are defined, you will see a focus on continuos improvement. Maturity levels build upon each other and key process areas have a set of common features that help to insure that mature processes are defined.

2. Defined, documented, and used processes - You will see many practices that address defined, documented, and used. These practices are especially prevalent in the Activities Performed common feature.

3. Commitment by senior management - This theme is contained primarily in the Commitment to Perform common feature. Policy statements are included in each key process area to address senior management sponsorship and commitment.

4. Stable processes - Processes must be stable and understood. Training helps to stabilize a process. Training practices are contained in the Ability to Perform common feature.

5. Measured processes - To objectively improve a process, it must be measured. Product measures are usually contained in the Activities Performed common feature. Measures of the process are contained in the Measurement and Analysis common feature.

6. Controlled processes - Processes need to be controlled to insure that they are being practiced as they are defined and documented. The Verifying Implementation common feature and the Software Quality Assurance key process area makes ensure that products and processes are verified and validated.

7. Process evolution - Processes evolve in the CMM. As processes improve, they will change over time. An example of this is the project management process.

Level 1 organizations really depend upon the project manager as the primary means of project management. At level 2, Software Project Planning and Software Project Tracking and Oversight key process areas are put into place to provide basic project management processes for the projects. At level 3, Integrated Software Management takes the project management processes from level 2 and develops the project's defined software process based on the organization's standard software process. At level 4, now that the project has a defined software process, data can be used to manage the project. This is addressed by the Quantitative Process Management key process area. The project manager has an expectation of how the project should perform prior to the start of the project. At level 5, a project now has the proper foundation in place to fine tune the project management process. This is addressed by the Process Change Management key process area.

As you can see the common features provide the primary themes in the CMM. I hope this addresses your question or at least points you in the right direction.

Incremental development

I think I have found a major weakness in the CMM. I'd like to hear your viewpoint on the matter. My CMM references are to SEI-91-TR-24.

The CMM does not emphasize incremental development of software systems. The term is not used in the CMM documentation as far as I know. The closest thing to "incremental development" that I find in the CMM is the mention of "serial build" in Level 2 Activity 5. The CMM seems to be neutral on the selection of the software life cycle. It requires "predefined stages of manageable size". But "stage" is ambiguous in this context. It could be taken to refer to a step in a process cycle rather than a complete increment that ends in test.

Incremental development has been identified as the most important component of the Cleanroom method by the leader of (to my knowledge) the largest Cleanroom project ever completed. ("OS32 and the Cleanroom", Proc. 1st European Industrial Symp. Cleanroom Software Engineering, 1993).

Terry Baker identified incremental development as the most important component of structured programming for large systems. (I don't know where the widespread belief that hierarchical, "go-to-less" module construction is the most important component of structured programming got started.) ("Structured Programming in a Production Programming Environment", IEEE Transactions on Software, Vol. SE-1, No. 2 1975, p. 105) Baker used the term "top-down development" rather than "incremental development". In its historical context, "top-down development" is an ambiguous term. In the 1970s, Baker and Harlan Mills tended to use the term "top-down development" for what was, in reality, an evolving concept. Its meaning evolved from layered step-wise refinement to incremental development during the 1970s. But, the context provided in

Baker's paper makes it clear that he was referring to the incremental development of new software systems.

Also, incremental development (under the name "the Milestone process") is apparently a key practice at Microsoft. (IEEE Software, Jan. 1995, p. 111).

Ironically, over a decade ago the proponents of incremental development identified two impediments to its widespread use: (1) it is often precluded by contract regulations and (2) it is often precluded by written development procedures, just the sort of regulations and procedures that the CMM is designed to influence. The Baker paper (cited earlier) describes ways to achieve incremental development in spite of government and commercial contracts regulations. Yourdon ("Top-Down Design and Testing" 1979, in "Software Design Strategies", Bergland, G. (editor), IEEE Computer Society Press, 1981, p. 60) found that in the organizations that had formal test procedures, those procedures commonly precluded incremental development.

CMM Activity 5 Level 2 calls for selecting an appropriate life cycle, but Yourdon and Baker have found that documents like the CMM have historically tended to obstruct the selection of the most appropriate life cycle model for many systems.

Incremental development potentiates process control by increasing the number of process cycles and the number of early opportunities to sample observed process capabilities. Incremental development potentiates defect prevention by allowing process defects to be identified in the early process cycles and eliminated.

The CMM calls for a incremental approach to process improvement, but it does not encourage an incremental approach to software development.

Overall, I am a supporter of the CMM. But the omission of a single key practice could greatly impact results.

What's your viewpoint on this matter?

I agree that software should be developed using some kind of evolutionary / incremental build / spiral life cycle. I don't think waterfall is a very good life cycle model for modern software projects.

The CMM does not, in general, prescribe how to solve specific problems. As you observe, all it says is "pick a life cycle and use it." This is one example of a general class of issues, where we have chosen to focus on what rather than how. There are cases were waterfall is the best life cycle. We do not want the CMM to be overly prescriptive.

If you write this up as a change request, it will be formally reviewed, tracked, and dispositioned. I would be interested in seeing the comments on such a proposed change, because my own natural tendency agrees with you. Once we start selecting how software projects should work, however, we're starting down a slippery path. What about replacing testing / peer reviews with formal methods? What about specifying inspections rather than the more general peer reviews? Why don't we recommend OOD? Should all projects use QFD?

We do rely, to large degree, on organizations and projects making reasonable decisions if they have a process that causes them to make the decision consciously. That is a risky assumption in many ways, but I'm more comfortable with under-specification than over-specification when it comes to an industry-wide "standard" such as the CMM.

Incremental process improvement

In the book Managing the Software Process, page 87, Section 6.2.1, regarding goals and objectives, it states that one of the first rules of thumb is to implement the product in small incremental steps, and to select each increment to support succeeding increments. I have chosen SEI Level 2 as my first increment, and therefore I am currently trying to clearly understand the difference between SEI Level 2 and SEI Level 3.

Actually, moving from Level 1 to Level 2 is a pretty big step. You may want to set smaller goals. The CMM does not say what those should be; it may differ based on your business environment. Empirically, the last KPAs that are mastered are Requirements Management, Software Project Planning, and SQA. Emphasis on MASTERED. When you do an assessment, the problems facing your organization should fairly obvious. The challenge is in prioritizing which ones to tackle first; the maturity levels give some good guidance that will help.

Legacy systems and maintenance documentation

How would you handle a situation where a large proportion of the organization's work load is legacy stuff and will continue to be so for the next couple of years? They have defined a life cycle model which essentially skips the high level design phase; thus they cannot answer yes to the PMM questions asking for traceability between top level design and requirements (2.4.8) or detailed design (2.4.11). They have other work which does follow "V" or Spiral SLCMs, but that is only about 35% of their work load and would be represented by at most three of the five PLs. [My opinion is that they should take the hit with No answers and hope that if 3 of 5 answers are Yes and they otherwise qualify for Level 3, the team would assess them at 3. In addition I'd produce a finding and a recommendation re creation of some re-engineered top level design document which could then be referenced in the traceability chains.]

This is almost the exact opposite of what I found useful when I was programming. I usually found that the detailed design was wrong and / or obsolete, but an accurate architecture, plus the code, let me understand what was going on in the system (after some study). I would agree with stating that, in this context, we don't do practices X, Y, or Z. I would probably argue (if I felt it was true), that rather than answering NO, they should be answered NOT APPLICABLE. You are definitely in a gray area here, where judgment is going to be crucial in scoring KPA satisfaction and level achievement. I would certainly ask the staff if they were having problems that were caused by the lack of high-level design info, concentrating especially on new people coming in (training issue). I would also write a finding that this was a potential problem and outline the pros and cons in doing something about it - both from a CMM scoring perspective and a maintaining the system perspective.

Software project dynamics

Is there a COST ESTIMATION model for CMM ? I mean, if I am an organization at CMM level (x), how can I estimate the cost (in person months, not dollars) of moving to level (x+1), x < 5.

I have read SEI-94-TR-12, and familiar with SEI-93-TR-24 and 25.

There is not a model, per se, but there has been some work in "software project dynamics" of modeling organizations at different maturity levels based on different assumptions. Herb Krasner and Stan Rifkin have done some work in this area. There's also a paper in American Programmer, Sept 1994, by Rubin, Johnson, and Yourdon on the topic.

Not Applicable versus risk

Another question is about the assessment of KPAs. If we have developed a written policy and a documented process for a certain KPA, e.g., Subcontractor Management, but the projects to be assessed do not have subcontractors, can we be considered to achieving that KPA by assessment?

Yes, but it might be more appropriate to score SSM as Not Applicable or to explicitly identify the risk if a new project will be doing extensive subcontracting, since there the implementation has not been successfully demonstrated yet.

Discipline versus bureaucracy

I have read CMM v1.1 carefully three times and discussed it with a group of my colleagues. I have the following observation. In an informal development environment such as the one here, the word "discipline" is easily and naturally read as "bureaucracy" or "rigidity". Personally, I believe that given a project's complexity and criticality, there is an appropriate level of formality which will optimize results.

The goal of CMM "Repeatable" Level seems to be to raise formality in order to conquer problems of complexity. In other words, it seems to concern itself solely with inappropriately LOW levels of formality. In "Managing the Software Process", Watts Humphrey says something like "No bureaucracy can ever produce truly great *products.". Many of us have worked at one time or another in bureaucratic organizations which seem to accomplish little real work, due to what looks like an excess of process. What mechanisms exist in the CMM to reduce inappropriately high levels of formality in a process?*

In particular, how does the Level 1 organization use the CMM to find the right degree of formality and not overshoot that goal? Is this a possible area for future development of the model?

The only mechanism in the CMM that addresses your concern is the hierarchy of the practices. There are no "shall" statements in the CMM, but to satisfy a key process area you have to satisfy each of the goals; to satisfy a maturity level you have to satisfy each of its KPAs.

The KPAs and goals are very abstract, and I would argue that they apply in any size or type of software project (with the exception of Software Subcontract Management if you aren't doing subcontracting). As you move down in detail, e.g., the subpractices and examples, you do move into a description of the normal behaviors we would expect to see in large-scale, government contracting kinds of projects.

We explicitly state that you have to apply professional judgment in appropriately applying the CMM as you move away from that specific context. Alternate implementations are quite possible, and their adequacy should always be considered with an open mind.

Because the CMM says what rather than how, you have to decide what the appropriate level of formality is in your business environment. We can't give generic advice that would apply to every conceivable user of the CMM beyond "apply intelligence." That may seem smug, but it's very hard to provide objective criteria for process standards in the software field (as Norm Fenton as pointed out).

When doing internal process improvement, this advice may be unsatisfactory. When you're concerned with external evaluations, a la SCEs, it can be even more difficult. The necessity is to be able to demonstrate to a neutral (in some cases, perhaps even adversarial) party that you have satisfactorily addressed a KPA's concerns. This leads to potential disagreements, i.e., reliability and consistency of evaluation issues, but in the last 7 years we have been unable to identify a better compromise position.

So the bottom line is, it's your responsibility to use the CMM as guidance for improving your process/evaluating contractors. You have to decide the appropriate degree of formality and rigor within your business environment. Your customers, who are part of your business environment, should share your "self-evaluation".

COTS

How is the CMM applied to COTS software integration projects?

As appropriate :-) Certainly there are implementation concerns for COTS, but the fundamental principles of planning, managing, communicating, etc., as described in the goals of the KPAs would certainly apply - just ask yourself what is a reasonable implementation of this practice or process for this project, and you won't go far wrong.

How is the CMM applied to Professional Services projects?

Ditto.

Required overtime

One of the tenets of CMM (as I understand it) is that as an organization moves up the levels things are going to work better and the organization will have "skilled, happy employees". With the downsizing we are seeing a very different picture. The last major job to be awarded was to XXX. When the award was announced it included 8 hours of uncompensated overtime per week per employee. So folks were immediately working 6 days a week for the same pay they had been receiving for 5 days! Happy campers they aren't. If a contractor had included this in a bid several years ago the Source Selection Board would have thrown out the proposal. This time they not only took it they encouraged it. We are now in the same mode for a much larger effort to support YYY. There are a lot of very unsure software people here. It is obvious this procurement may go the same way as the last one. Some of these people are currently in CMM self assessments. They are having a tough time matching the CMM material with the world they live in......

We have met the enemy and it is all of us! (apologies to Pogo)

That's pretty terrible. I'm afraid that organizations following this "improvement" tack are ignoring the human side of process improvement that's critical to continuous improvement. This sounds like just another way of going to the lowest bidder, but using SPI as an excuse to justify the low-ball. If this is more than a temporary, one-shot deal then the best people will leave for greener pastures - and they really will be greener. If you want me to work round the clock, you better offer to make me a millionaire so I can retire in a few years - but I doubt that's your environment ;-)

I think I'd identify this as a finding in my next assessment. Also, I have some neat comics that describe this kind of "improvement" in a graphic way. If you're around the SEI anytime I'm here, drop by and see them.

Methodologies

What impact/effect do methodologies such as Yourdon/DeMarco and TIs IEM have on the CMM effort?

Essentially none. In the CMM we basically say "pick a methodology that works well for you." The CMM does not recommend any specific methodology, and I don't believe that we have anything in the CMM inspired by a particular methodology that is generally valuable to all software efforts.

How do "tools" such as TIs IEF, Finkelstein's IE Advantage and ADW (to name a few) affect this process? Do these products shorten the cycle, are they an important part of the process or does SEI recommend its own tools?

We do not recommend any specific tools. In general, tools make you more productive, increase quality, decrease cycle time, etc., when they are integrated into a well-defined process. If they aren't integrated, they become shelfware. Technology transition of tools can be tricky; one needs to be sensitive to process and cultural impacts of change. No one, not even the SEI :-), can say "this is the right tool / methodology that will solve all of the software community's problems." (Even if there was a technically superior answer, you'd still have to consider the organization's processes and culture!) SEI is not a tool vendor and we do not endorse any specific tools or methods.

Regular versus periodic

We were going through compliance requirements attempting to make them more CMM-like in style and kept running into the word "regular" and "periodically." So it occurred to us that perhaps the CMM used one term consistently to convey the idea. So we did a key word search and discovered that the word "regular" occurs 15 times in the CMM and the word "periodic" occurs 40 times.

There's no significant difference between regular and periodic, as far as I recall. In fact, we tried to go to "periodic" as the normal word; I don't remember if "regular" appears for good reason or just because we didn't catch it in the editing process. I did check that it's only in subpractices. The other significant term is "event-driven", so things can be either periodic or event-driven.

So our question is what, if any, difference does it make which term is used?

Periodic is a little bit stronger in terms of "weekly" or "monthly" kinds of implication. I'd probably pick one that I was comfortable with (perhaps a bias to periodic) and use it.

The Rational Planning of (Software) Projects

Mark C. Paulk
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
U.S.A.

Abstract

The software crisis has persisted for decades. Our difficulties in planning and managing software projects may be rooted in fundamental human nature, as suggested by research in rational decision making, more than in the inherent difficulty of building software. The Capability Maturity Model^{SM} for Software, an application of the concepts of Total Quality Management to software development and maintenance, embodies one approach for improving the software process. The problems addressed by both the CMM^{SM} and TQM seem to lie in the basic ways that human beings think and organize themselves. In many circumstances, normal human decision making can be characterized as "irrational" because of systematic biases and fallacies in the way people make decisions. Mechanisms such as those suggested by the CMM support rational decision making.

1 Introduction

Over the last three decades we have heard many complaints about the "software crisis." In some software organizations, the typical software project is a year late and double the budget. In a review of one DOD software organization's 17 major projects, the average 28-month schedule was missed by an average of 20 months. A four year project was not delivered for seven years; no project was on time.

A recent Government Accounting Office (GAO) report /6/ on the major software challenges states, "We have repeatedly reported on cost rising by millions of dollars, schedule delays of not months but years, and multi-billion-dollar systems that don't perform as envisioned." The report summarizes over 20 GAO case studies involving software or software-related problems in the military. This report concludes, "The understanding of software as a product and of software development as a process is not keeping pace with the growing complexity and software dependence of existing and emerging mission-critical systems."

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

This paper was published in the **Proceedings of the First World Congress for Software Quality**, San Francisco, CA, 20-22 June 1995, section 4.

Is it accurate, however, to refer to a situation as a crisis that has persisted for decades? The software crisis is a chronic and severe problem that needs to be addressed, but it does not seem to be acute.

Some software savants have suggested that in other disciplines, such as construction, an estimate off by more than 6% is considered a disaster. Studies indicate, however, that "in capital investment projects, the typical project finishes late, comes in over budget when it is finally completed, and fails to achieve its initial goals." /10/ For the projects studied, the norm was for actual construction costs to more than double first estimates.

The problems facing the software industry may be more ubiquitous than we sometimes think. Many of the problems contributing to the software crisis derive more from human nature and ability than they do from the inherent complexity of software systems, daunting though that complexity may be. The "software crisis" is not unique to the software industry.

A sizable body of literature on rational decision making and human foibles has grown over the last few years. /1/ The purpose of this paper is to summarize some of those results, how they apply to the planning of software projects, and how the Capability Maturity Model for Software (CMM) /13, 14/ developed by the Software Engineering Institute (SEI) can contribute to making rational decisions, specifically when planning software projects.

2 Systematic Biases in Human Nature

Software engineering is a human-intensive, design-intensive endeavor. The largest single factor in the success of software projects is the competence of the people doing the work. People are fallible, however, and "an accumulating body of recent research on clinical judgment, decision making, and probability estimation has documented a substantial lack of ability across both individuals and situations." /4/

In this paper we shall concentrate on three general biases that all humans are systematically subject to:

- people tend to be risk averse when there is a potential of loss
- people are unduly optimistic in their plans and forecasts
- people prefer to use intuitive judgment rather than (quantitative) models

Managers are people also, and these human tendencies apply to the managers who make business decisions at the project to the executive level.

2.1 Problem: avoiding risk and uncertainty

In the business world, managers face contingencies they cannot predict and outcomes they will not control, no matter how intelligent, educated, or experienced they are. This variability can be characterized as risk. Managers agree that risk taking is essential to success in decision making and an essential component of the managerial role. /12/

According to game theory, decisions should be made that maximize expected value. In reality, people are risk averse when dealing with the possibility of loss. For many managers, studies indicate that losses loom about twice as large as gains. /10/ Avoiding risk could be characterized as being prudent, but the axiom "nothing ventured, nothing gained" also applies, especially in an increasingly competitive world.

What are the some of the factors that drive conservatism? First, society[1] values risk taking, but not gambling, and society defines gambling as risk taking that turns out badly. /12/ The results sort decision makers into winners and losers, and society interprets these differences as reflecting differences in judgment and ability, although events outside the individual's control may have strongly influenced success.

Second, the consequences of acting versus not acting are evaluated differently. This biases decision makers toward passivity, even when passivity may not be in their own best interests. /10/ In general, loss aversion favors conservatism and inertia in decision making. When a manager takes a risk, it is often because of optimistic denial rather than bold acceptance.

It is better, therefore, not to act than to act and be wrong; if you do act, you need to be right. Since acting – taking risks – is an essential part of a manager's job, the conflict results in the denial of uncontrollable risks. Kahneman and Lovallo state that "for managers, risk is a challenge to be overcome by the exercise of skill, and the role of uncontrollable contingencies is to be denied or minimized." /10/ While some risks can be allowed for in planning, many other risks are outside the control of an individual manager.

2.2 Problem: undue optimism

While there is a tendency for managers to be conservative, there is a counterbalancing bias that people tend to be overly optimistic about what they can do. Kahneman and Lovallo state that "there are three main forms of pervasive optimistic bias: 1) unrealistically positive self-evaluations, 2) unrealistic optimism about future events and plans, and 3) an illusion of control." /10/

When there is little effective feedback on how good a planning effort was, judgments are not well calibrated. In one study, Einhorn and Hogarth comment

[1] Society may not be the best term, but it seems the most general. March and Shapira use "society" and "history." An alternative would be "your company," but that seems inappropriate. Society as used here can be considered a term for any grouping of people.

that “self-confidence in judgments... was found to increase as a function of the amount of information available... but without any corresponding increase in judgmental accuracy.” /4/ We plan, we act, we replan, we react – and the accuracy of the judgments that contribute to the rework become lost in the noise of the ad hoc, chaotic environment of the typical project.

Managers also tend to ignore possible events that are very unlikely or very remote, regardless of their consequences. /12/ Managers focus on a few possible outcomes – the norm – rather than the whole distribution, and plan with respect to those few points. This aspect of denying risk also leads to unrealistic optimism.

Denial seems to be a common solution to many of the conflicting forces managers have to deal with. As March and Shapira express it, “Managers focus on ways to reduce the danger while retaining the gain. One simple action is to reject the estimates.” /12/ This is hardly unknown in the software world. Proposal managers have been known to cut a schedule by 25%, because the realistic bid would not win the contract. Even after estimates have been developed and revised, most managers believe they can do better than is expected. /12/

Undue optimism is perhaps not a major problem in the software world, though a feeling of futility may be. Few experienced software managers claim great confidence in their planned schedules, because of their experience with:

- volatile requirements. How closely will the system eventually built correspond to the one originally planned?
- rapidly changing technology. Is the underlying technology likely to go through one or more evolutionary advances during development?
- historical performance. How do we define good performance for a software project? Within 50% of estimate?

2.3 Problem: relying on intuitive judgment

Denial of risk and optimism are much easier when there is nothing to rub your nose in the facts. People resist documenting procedures and standards and using quantitative models. Typical reasons include fear of bureaucracy and rigid control of a dynamic process. These reasons are valid, if the process is poorly implemented.

Managers are more comfortable with verbal characterizations of risk than with numerical characterizations. March and Shapira found that “A majority... felt that risk could not be captured by a single number or distribution, that quantification of risks was not an easy task, ... there was no way to translate a multidimensional phenomenon into one number... [but] everything should be expressed in terms of the profit (or loss) at the end of the project.” /12/

Surveys of estimating practices in one organization /8/ indicate that disciplined estimating approaches, such as Delphi techniques or cost models, are rarely used. Variances are so large that there is a 30% probability that any one estimate can be more than 50% off. The vast majority of projects are planned based on intuitive judgment, and success is declared at the level of functionality present when the schedule or money run out. This is a fairly common state of affairs in many software development and maintenance environments.

The result of the intuitive approach to software estimating is that software planning is frequently considered a wasted effort. We thus have the worst of both possibilities mentioned by Einhorn and Hogarth: little judgmental accuracy and little confidence in spite of (or because of) our experience. We cannot address risk aversion and overoptimism biases without also dealing with also addressing the systematic fallacies inherent in global intuitive judgment.

2.4 Summing up the problem

As Hihn and Habib-Agahi point out, "when a person is forced to plan under severe schedule and budget constraints, planning consists mostly of a prioritized list and a lot of optimistic promises." /8/

Because most software organizations that we have observed (about 75% according to the SEI's 1994 data) do not use a disciplined approach, we consistently see the results of poor project planning and management evidenced as:

- unrealistic plans, based on optimistic estimates of what can be done (and managerial pessimism regarding the bids that need to be proffered to win a contract),
- ineffective tracking of performance, based on a lack of "measured insight" into how a project is progressing ("rework happens" being the confounding factor),
- volatile requirements that confound estimates and are incorporated into the project in an ad hoc fashion, and
- risks that unexpectedly, but not unpredictably, become catastrophes.

Can we learn from experience? Can we control these natural tendencies in human nature? Einhorn and Hogarth suggest that "the difficulty of learning from experience has been traced back to three main factors: (a) lack of search for and use of disconfirming evidence, (b) lack of awareness of environmental effects on outcomes, and (c) the use of unaided memory for coding, storing, and retrieving outcome information." /4/

3 Controlling Human Nature

March and Shapira suggest that "it may be more efficacious to try to modify managerial attention patterns and conceits than to try to change beliefs about the likelihood of events or to try to induce preferences for high variance alternatives." /12/ An example of the impact of paying attention is that people tend to optimize what is measured, which can confound the effectiveness of measurement programs.

The SEI's Capability Maturity Model for Software (CMM) recommends institutionalizing measures and procedures based on historical performance and statistical analysis as "the accepted way of doing business," i.e., part of an organizational culture. Tools and techniques, such as Delphi techniques and estimation models, have been developed to give some degree of objectivity to planning and overcome human frailty.

Specifically, the CMM recommends:

- documenting the way work is performed
- measuring its performance
- using data for controlling the performance
- planning based on historical performance
- training people

The CMM is structured into five maturity levels, which can be briefly described as:

1) Initial	The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.
2) Repeatable	Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
3) Defined	The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.
4) Managed	Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.
5) Optimizing	Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

These maturity levels, in turn, are decomposed into 18 key process areas. This paper concentrates on one of those areas: Software Project Planning. In the balance of this paper we shall review some of the ways of dealing with the systemic biases that people have and how the CMM can help with these biases.

3.1 Solution: dealing with risk and uncertainty

One of March and Shapira's observations is that risk taking in organizations is sustained more by personal than by organizational incentives. /12/ Organizational support in understanding and prioritizing risks can be a critical component of empowering managers to deal with risk more effectively (note that the CMM is a model for organizational improvement). Activity 13 of Software Project Planning addresses identifying, assessing, and documenting software risks.

Risk management may not be a precise science, but it is a reasonable approach to controlling contingencies that, while they may not be predictable in detail, are predictable in general. The exercise of documenting and tracking (foreseeable) risks is valuable in keeping us from losing them below our attention threshold.

3.2 Solution: being realistic

Mechanisms such as those advocated by the CMM can help a manager be more realistic in planning software projects. These include using documented procedures and standards for estimating and planning and taking advantage of historical performance data.

That does not remove the need to win the contract or to capture the customer, as a necessary first step for doing business. Kahneman and Lovallo point out that winning projects are more likely to be associated with optimistic errors. /10/ Time to market is a critical factor in the profitability of commercial products. Being realistic does not imply not being aggressive.

Realistic here does mean that managers have to decide whether software engineering is a "manageable" process. The CMM takes the position that it is and advocates a particular approach for evolving the process capability of an organization. This is not a unanimous view. Einhorn characterizes two possible classes of error when making judgments. If people decide that a phenomenon is systematic when it is random, the error that results is manifested in myths, magic, superstitions, and illusions of control. If people decide that a phenomenon is random when it is systematic, the error that results is manifested in lost opportunities and illusions of the lack of control. /5/

Managers have to act in the absence of complete information, what the military calls the "fog of war." Documented procedures and standards can help focus attention on issues that might otherwise be neglected; a logical complement is the use of quantitative models to support the judgments that are made.

In the academic world, we can perform controlled experiments. The concept of a control group is essential to the scientific method, but controlled experiments are typically impractical in the real world. We can, however, capture feedback on judgments, actions, process, environmental variables, and the resulting outcomes in the real world, which is an essential part of improving judgment.

Human beings are fallible. Rational decision making depends on the tools available to the decision maker. The rational, intelligent use of those tools provides a means for organizational learning. There are three conditions for learning: (1) feedback, which is necessary but not sufficient; (2) the ability to rearrange cases so that hypotheses can be verified or discounted; and (3) the ability to tally the accuracy of one's hypotheses. /4/

3.3 Solution: using models

The literature is rife with studies that demonstrate the power of even simple statistical models for combining information over the judgments of human experts. /4/ Dawes, Faust, and Meehl compared the power of the clinical method, where the decision-maker combines or processes information in his or her head, and the actuarial method, where the human judge is eliminated and conclusions rest solely on empirically established relations between data and the condition or event of interest.[2] Their conclusion was that the actuarial method is almost unanimously equal or superior to the clinical method. In nearly all of the comparative studies in the social sciences, the actuarial method has equaled or surpassed the clinical method. /2/

This is not an intuitively obvious result. Studies /1, 2, 4, 10/ have demonstrated that:

- simple statistical models based on human judges are more often correct than the human judges they are based on[3]
- even when human judges are provided the outputs of the model and allowed to use their discretion, relying uniformly on the actuarial conclusions provides greater overall accuracy than the human judge[4]

---

[2] Dawes, Faust, and Meehl state that "to be truly actuarial, interpretations must be both automatic (that is, prespecified or routinized) and based on empirically established relations." They then go on to say that "virtually any type of data is amenable to actuarial interpretation." While the latter quote may be an overstatement, it certainly applies in the domain of software project estimating.

[3] One of the advantages of models is that you get consistent answers when given the same inputs.

[4] When operating freely, human judges apparently identify too many "exceptions."

- optimal weighting of variables, so long as the sign of the coefficients are correct, is not crucial to performing better than the human judge[5]

Kahneman and Lovallo explain this phenomenon in terms of inside and outside views. /10/ An inside view forecast is generated by focusing on the case at hand, by considering the plan and the obstacles to its completion, by constructing scenarios of future progress, and by extrapolating current trends. An outside view forecast focuses on the statistics of a class of cases chosen to be similar in relevant respects to the present one. The critical question in both cases is whether a to treat a particular problem as unique or as an instance of a class of similar problems.

The inside view is overwhelmingly preferred in intuitive forecasting because it is viewed as a serious attempt to come to grips with the complexities of the unique case at hand. The outside view is rejected for relying on crude analogy from superficially similar instances. Critics of software process management, who emphasize the innovative, creative aspects of software engineering, are basically arguing from the inside view.

Take a simple example. /5/ There is an urn with 60% red balls and 40% green balls. Predict the color of balls as they are drawn from the urn. Knowing the odds, most people use a rule that leads to predicting red 60% of the time and green 40% of the time. That leads to 52% correct predictions (.060 * 0.60 + 0.40 * 0.40). The simple rule “always predict the most likely color” leads to 60% correct predictions. Such a strategy deliberately accepts error, but when dealing with a truly random process, it is the best selection rule.

The future of a long and complex undertaking is simply not foreseeable in detail. The outside view is a conservative approach, which will fail to predict extreme and exceptional events, but will do well with common ones; it is much more likely to yield a realistic estimate. Its main advantage is that it avoids the snares of scenario thinking.[6]

[5] Optimal weights are specific to the population in which they were derived, and any advantage gained in one setting may be lost when the same method is applied in another setting. Equally weighted linear regression models can outpredict models with “optimal” weights on new cases if there is poor data.

[6] In scenario thinking, a sequence of events may be judged more probable than one of its components. Dawes /1/ gives the example of an alcoholic tennis star who drinks a fifth a day winning a major tournament versus an alcoholic tennis star who drinks a fifth a day, joins Alcoholics Anonymous, quits drinking, and wins a major tournament. The probability of winning is higher than the probability of (joining AA) * (quitting drinking) * (winning), yet the sequence of events in the scenario sounds more plausible. The software planning equivalent could be making an aggressive commitment and bringing the contract in on time and on budget versus making the commitment, hiring highly competent people, providing powerful new tools and methods, and bringing the contract in on time and on budget.

Dawes, Faust, and Meehl state that “although surpassing clinical methods, actuarial procedures are far from infallible, sometimes achieving only modest results.” /2/ This highlights a difficult problem in using such techniques – people would rather remain ignorant of how bad their own judgment is than rely on “mechanical” techniques where they are all too aware of the limitations.

Models are simplifications of the real world by definition. They can never capture the richness and complexity of the real-world phenomenon they describe; therefore there will be errors in prediction. Would we be willing to accept such an obviously oversimplistic rule as “always predict the most likely color” if we were gambling? If we were managing risks in a project that we were responsible for?

Human beings do have a unique capacity to observe and make “atomic judgments,” but this is not the same as a unique capacity to predict on the basis of integration of observations. /2/ Human beings can build models, but when it comes to consistently performing better than the models they have produced, they are almost invariably unsuccessful – unless they “load the dice” by taking action that makes their predictions come true.

March and Shapira argue that “it is entirely sensible for a manager to conclude that the credibility of probability estimates is systematically less than is the credibility of estimates of the value of an outcome, and it is certainly arguable that the relative credibility of estimates should affect the relative attention paid to them.” /12/ However, having a rational reason, rather than a gut feel, for making managerial decisions provides a foundation for learning, both as individuals and organizations.

3.4 Summing up a solution

There has been a consistent theme running through this discussion of controlling human frailty:

- document how decisions are made
- provide guidance and quantifiable criteria where possible
- record decisions and the data used to make them
- analyze results and improve the process where possible
- learn – individually and organizationally

This is process management as advocated in the CMM. It is not a panacea to the difficult problems of management, but it is a powerful approach to attacking those problems.

March and Shapira argue that “We may prefer to have managers imagine (sometimes falsely) that they can control their fates, rather than suffer the consequences of their imagining (sometimes falsely) that they cannot.” /12/ Is ignorance bliss?

As George Box said, “All models are wrong; some models are useful.” If the models we use are successful, even if embedded in a human expert’s head, then there is little demand for changing the current way of doing business. That is not the case in the software world, as we have already shown. A strong need has been expressed for improvement in the way we develop and maintain software.

4 Empirical Data and Experimentation

There are a number of case studies of software process improvement based on the CMM /3, 9, 11, 15/. All indicate that predictability, control, and effectiveness of processes can be significantly improved by improving the software process as suggested by the CMM. The CMM, as argued in this paper, can help organizations control natural human biases related to decision making.

The typical return on investment, based on data from organizations that have done software process improvement for more than 3 years, is about 7:1, with an average gain in productivity of 37% per year, an average 18% increase each year in the proportion of defects found in pre-test, an average 19% reduction in time to market, and an average 45% reduction in field error reports per year.

This is not a statistically rigorous analysis. The SEI began such an effort in 1993, but it will require several years to develop rigorous evidence on the effects of CMM-based improved efforts, although some initial results have been published. /7/ That does not mean, however, that organizations cannot demonstrate the value of these concepts more quickly.

As already mentioned, there is significant resistance in many organizations to the concept of documented processes. They are viewed as being bureaucratic, rigid, and stultifying. Although there are instances where such counter-productive processes have been mandated by organizations, there are also instances where powerful, empowering, effective processes have been developed and deployed across organizations.

To overcome this resistance, it may be necessary to prototype and work our way up in complexity and criticality. Most sizable organizations have a range of software projects. Some are large, complex, and critical to the business. Others are comparatively small and/or of little impact. Software professionals value the concept of prototyping, so it seems reasonable to propose prototyping the use of documented standards, procedures, and models. Champions can then be identified, based on the success (or perhaps failure!) of the prototyping effort.

What would the confounding factors be? First, there might be a Hawthorne effect. Second, we would primarily be using outcome information. For small projects, it may be difficult to incorporate process feedback. Third, people will be acting on the basis of the plans and estimates generated, which may confound the results. Fourth, the customer is a critical component of the overall system. An unreasonable customer can cripple the process.

5 Conclusion

As Kahneman and Lovallo suggest, facing the facts can be intolerably demoralizing. A realistic view may not provide an acceptable basis for continuing an ongoing project, but no one is willing to abandon the sunk costs and draw the inescapable conclusion that the project should be scrapped – at least until far more money has been expended than was originally anticipated.

Perhaps unfounded optimism is the only effective remedy against paralysis; perhaps there is a genuine dilemma that will not yield to any simple rule. Ignoring the issue puts us in the same position as the lost platoon in Weick’s famous story that finds its way in the Alps by consulting a map of the Pyrenees.

The CMM provides a powerful tool for attacking our software management problems. We may be horrified by how “bad” some of the solutions are – yet the really terrifying observation may be that they are better than the current state of affairs. They also provide a foundation for learning and improvement that is currently lacking – and that is the critical driver for beginning the journey of continuous process improvement.

References

1. Robyn M. Dawes, *Rational Choice in an Uncertain World*, Harcourt Brace Jovanovich College Publishers, Orlando, FL, 1988.

2. Robyn M. Dawes, David Faust, and Paul E. Meehl, “Clinical Versus Actuarial Judgment” Science, Vol. 243, 31 March 1989, pp. 1668-1674.

3. Raymond Dion, "Process Improvement and the Corporate Balance Sheet," IEEE Software, Vol. 10, No. 4, July 1993, pp. 28-35.

4. Hillel J. Einhorn and Robin M. Hogarth, “Confidence in Judgment: Persistence of the Illusion of Validity,” Psychological Review, Vol. 85, No. 3, 1978, pp. 395-416.

5. Hillel J. Einhorn, “Accepting Error to Make Less Error,” Journal of Personality Assessment, Vol. 50, No. 3, Fall 1986, pp. 387-395.

6. *Mission-Critical Systems - Defense Attempting to Address Major Software Changes,* General Accounting Office, GAO / IMTEC-93-13, December 1992, pp. 1-29.

7. James Herbsleb, Anita Carleton, et al., “Benefits of CMM-Based Software Process Improvement: Initial Results,” Software Engineering Institute, CMU/SEI-94-TR-13, August 1994.

8. J. Hihn and H. Habib-Agahi, "Cost Estimation of Software Intensive Projects: A Survey of Current Practices," *Proceedings of the 13th International Conference on Software Engineering*, Austin, TX, 13-17 May 1991, pp. 276-287.

9. Watts S. Humphrey, T.R. Snyder, and Ronald R. Willis, "Software Process Improvement at Hughes Aircraft," IEEE Software, Vol. 8, No. 4, July 1991, pp. 11-23.

10. Daniel Kahneman and Dan Lovallo, “Timid Decisions and Bold Forecasts: A Cognitive Perspective on Risk Taking,” *Proceedings of Conference on Fundamental Issues in Strategy*, Silverado, 1990.

11. W.H. Lipke and K.L. Butler, "Software Process Improvement: A Success Story," Crosstalk: The Journal of Defense Software Engineering, No. 38, November 1992, pp. 29-31.

12. James G. March and Zur Shapira, “Managerial Perspectives on Risk and Risk Taking,” Management Science, Vol. 11, No. 11, November 1987, pp. 1404-1418.

13. M.C. Paulk, B. Curtis, M.B. Chrissis, and C.V. Weber, “Capability Maturity Model for Software, Version 1.1,” Software Engineering Institute, CMU/SEI-93-TR-24, DTIC Number ADA263403, February 1993.

14. M.C. Paulk, C.V. Weber, S. Garcia, M.B. Chrissis, and M. Bush, “Key Practices of the Capability Maturity Model, Version 1.1,” Software Engineering Institute, CMU/SEI-93-TR-25, DTIC Number ADA263432, February 1993.

15. H. Wohlwend and S. Rosenbaum, "Software Improvements in an International Company," *Proceedings of the 15th International Conference of Software Engineering*, Washington D.C, May 1993.

Mapping of SW-CMM v1.1 to SE-CMM v1.1

SW-CMM v1.1	SE-CMM v1.1
2.1.0.1 RM.GO.1 requirements baseline	
2.1.0.2 RM.GO.2 consistent with requirements	
2.1.1.1 RM.CO.1 RM policy	
2.1.2.1 RM.AB.1 requirements allocation	BP 02.06 Derive and Allocate Requirements: Allocate Requirements
2.1.2.1 RM.AB.1 requirements allocation	BP 03.04 Evolve System Architecture: Develop Interface Requirements
2.1.2.1 RM.AB.1 requirements allocation	BP 06.03 Understand Customer Needs and Expectations: Develop System Requirements
2.1.2.2 RM.AB.2 requirements documented	BP 02.06 Derive and Allocate Requirements: Allocate Requirements
2.1.2.2 RM.AB.2 requirements documented	BP 06.01 Understand Customer Needs and Expectations: Elicit Needs
2.1.2.2 RM.AB.2 requirements documented	BP 06.03 Understand Customer Needs and Expectations: Develop System Requirements
2.1.2.3 RM.AB.3 RM resources	
2.1.2.4 RM.AB.4 RM training	
2.1.3.1 RM.AC.1 review allocated requirements	BP 02.07 Derive and Allocate Requirements: Ensure Requirement Verifiability
2.1.3.2 RM.AC.2 use allocated requirements	
2.1.3.3 RM.AC.3 change allocated requirements	
2.1.4.1 RM.ME.1 measure RM status	
2.1.5.1 RM.VE.1 senior management review of RM	
2.1.5.2 RM.VE.2 project manager review of RM	
2.1.5.3 RM.VE.3 SQA audits of RM	
2.2.0.1 PP.GO.1 estimate	
2.2.0.2 PP.GO.2 plan the project	
2.2.0.3 PP.GO.3 establish commitments	
2.2.1.01 PP.CO.1 project software manager	
2.2.1.02 PP.CO.2 SPP policy	
2.2.2.01 PP.AB.1 statement of work	

2.2.2.02 PP.AB.2 responsibilities for planning	
2.2.2.03 PP.AB.3 SPP resources	
2.2.2.04 PP.AB.4 SPP training	
2.2.3.01 PP.AC.1 participate on proposal	BP 12.01 Plan Technical Effort: Identify Critical Resources
2.2.3.01 PP.AC.1 participate on proposal	BP 12.09 Plan Technical Effort: Develop Technical Management Plan
2.2.3.02 PP.AC.2 initiate planning early	BP 12.05 Plan Technical Effort: Identify Technical Activities
2.2.3.02 PP.AC.2 initiate planning early	BP 12.09 Plan Technical Effort: Develop Technical Management Plan
2.2.3.03 PP.AC.3 participate in project planning	BP 12.05 Plan Technical Effort: Identify Technical Activities
2.2.3.03 PP.AC.3 participate in project planning	BP 12.10 Plan Technical Effort: Review Project Plans
2.2.3.04 PP.AC.4 senior management reviews commitments	BP 12.06 Plan Technical Effort: Define Project Interface
2.2.3.04 PP.AC.4 senior management reviews commitments	BP 12.10 Plan Technical Effort: Review Project Plans
2.2.3.05 PP.AC.5 software life cycle	BP 12.04 Plan Technical Effort: Determine Project Process
2.2.3.06 PP.AC.6 develop software development plan	
2.2.3.07 PP.AC.7 document software development plan	BP 12.07 Plan Technical Effort: Develop Project Schedules
2.2.3.08 PP.AC.8 identify work products for control	
2.2.3.09 PP.AC.9 estimate size	
2.2.3.10 PP.AC.10 estimate effort and costs	BP 12.03 Plan Technical Effort: Estimate Project Costs
2.2.3.11 PP.AC.11 estimate critical computer resources	BP 12.02 Plan Technical Effort: Estimate Project Scope
2.2.3.11 PP.AC.11 estimate critical computer resources	BP 12.08 Plan Technical Effort: Establish Technical Parameters
2.2.3.12 PP.AC.12 develop schedule	
2.2.3.13 PP.AC.13 identify risks	BP 10.02 Manage Risk: Identify Risks
2.2.3.14 PP.AC.14 identify facilities and support tools	BP 16.02 Manage Systems Engineering Support Environment: Determine Support Requirements
2.2.3.15 PP.AC.15 record planning data	
2.2.4.1 PP.ME.1 measure SPP status	
2.2.5.1 PP.VE.1 senior management review of SPP	

2.2.5.2 PP.VE.2 project manager review of SPP	
2.2.5.3 PP.VE.3 SQA audits of SPP	
2.3.0.1 PT.GO.1 track	
2.3.0.2 PT.GO.2 take corrective actions	
2.3.0.3 PT.GO.3 agree to changes	
2.3.1.1 PT.CO.1 project software manager	
2.3.1.2 PT.CO.2 PTO policy	
2.3.2.1 PT.AB.1 software development plan	
2.3.2.2 PT.AB.2 responsibility assigned for work	
2.3.2.3 PT.AB.3 PTO resources	
2.3.2.4 PT.AB.4 PTO training	
2.3.2.5 PT.AB.5 PTO orientation	
2.3.3.01 PT.AC.1 use software development plan	BP 11.01 Monitor and Control Technical Effort: Direct Technical Effort
2.3.3.02 PT.AC.2 revise software development plan	
2.3.3.03 PT.AC.3 review commitments with senior management	BP 11.01 Monitor and Control Technical Effort: Direct Technical Effort
2.3.3.04 PT.AC.4 communicate changes to commitments	
2.3.3.05 PT.AC.5 track size	BP 08.07 Ensure Quality: Mechanism for corrective actions
2.3.3.06 PT.AC.6 track effort and costs	BP 08.07 Ensure Quality: Mechanism for corrective actions
2.3.3.06 PT.AC.6 track effort and costs	BP 11.02 Monitor and Control Technical Effort: Track Project Resources
2.3.3.06 PT.AC.6 track effort and costs	BP 11.06 Monitor and Control Technical Effort: Control Technical Effort
2.3.3.07 PT.AC.7 track critical computer resources	BP 08.07 Ensure Quality: Mechanism for corrective actions
2.3.3.08 PT.AC.8 track schedule	BP 08.07 Ensure Quality: Mechanism for corrective actions
2.3.3.08 PT.AC.8 track schedule	BP 11.04 Monitor and Control Technical Effort: Review Project Performance
2.3.3.09 PT.AC.9 track technical activities	BP 08.07 Ensure Quality: Mechanism for corrective actions
2.3.3.09 PT.AC.9 track technical activities	BP 11.01 Monitor and Control Technical Effort: Direct Technical Effort
2.3.3.09 PT.AC.9 track technical activities	BP 11.03 Monitor and Control Technical Effort: Track Technical Parameters

2.3.3.10 PT.AC.10 track risks	BP 08.07 Ensure Quality: Mechanism for corrective actions
2.3.3.11 PT.AC.11 record measurement and replanning data	
2.3.3.12 PT.AC.12 conduct internal technical reviews	BP 08.07 Ensure Quality: Mechanism for corrective actions
2.3.3.13 PT.AC.13 conduct formal reviews	BP 06.05 Understand Customer Needs and Expectations: Inform Customer
2.3.3.13 PT.AC.13 conduct formal reviews	BP 08.07 Ensure Quality: Mechanism for corrective actions
2.3.3.13 PT.AC.13 conduct formal reviews	BP 11.05 Monitor and Control Technical Effort: Analyze Project Issues
2.3.4.1 PT.ME.1 measure status of PTO	
2.3.5.1 PT.VE.1 senior management review of PTO	BP 11.05 Monitor and Control Technical Effort: Analyze Project Issues
2.3.5.2 PT.VE.2 project manager review of PTO	BP 11.05 Monitor and Control Technical Effort: Analyze Project Issues
2.3.5.3 PT.VE.3 SQA audits of PTO	BP 11.05 Monitor and Control Technical Effort: Analyze Project Issues
2.4.0.1 SM.GO.1 select qualified subcontractors	
2.4.0.2 SM.GO.2 agree to subcontractor commitments	
2.4.0.3 SM.GO.3 maintain ongoing communications	
2.4.0.4 SM.GO.4 track performance of subcontractors	
2.4.1.1 SM.CO.1 SSM policy	
2.4.1.2 SM.CO.2 subcontract manager	
2.4.2.1 SM.AB.1 SSM resources	
2.4.2.2 SM.AB.2 SSM training	
2.4.2.3 SM.AB.3 SSM orientation	
2.4.3.1 SM.AC.01 define work to be subcontracted	BP 18.01 Coordinate with Suppliers: Identify needed components
2.4.3.2 SM.AC.02 select subcontractor	BP 18.02 Coordinate with Suppliers: Identify suppliers
2.4.3.2 SM.AC.02 select subcontractor	BP 18.03 Coordinate with Suppliers: Choose suppliers
2.4.3.3 SM.AC.03 establish contractual agreement	
2.4.3.4 SM.AC.04 approve subcontractor's plan	
2.4.3.5 SM.AC.05 use subcontractor's plan	

2.4.3.6 SM.AC.06 deploy changes to work	BP 18.04 Coordinate with Suppliers: Provide needs
2.4.3.7 SM.AC.07 conduct status/coordination reviews	BP 18.05 Coordinate with Suppliers: Maintain communication
2.4.3.8 SM.AC.08 conduct technical interchanges	BP 18.05 Coordinate with Suppliers: Maintain communication
2.4.3.9 SM.AC.09 conduct formal reviews	BP 18.05 Coordinate with Suppliers: Maintain communication
2.4.3.10 SM.AC.10 SQA monitoring of subcontractor	
2.4.3.11 SM.AC.11 SCM monitoring of subcontractor	
2.4.3.12 SM.AC.12 acceptance testing of subcontracted product	
2.4.3.13 SM.AC.13 evaluate subcontractor performance	
2.4.4.1 SM.ME.1 measure status of SSM	
2.4.5.1 SM.VE.1 senior management review of SSM	
2.4.5.2 SM.VE.2 project manager review of SSM	
2.4.5.3 SM.VE.3 SQA audits of SSM	
2.5.0.1 QA.GO.1 plan SQA	
2.5.0.2 QA.GO.2 verify adherence	
2.5.0.3 QA.GO.3 inform of SQA	
2.5.0.4 QA.GO.4 resolve noncompliance	
2.5.1.1 QA.CO.1 SQA policy	
2.5.2.1 QA.AB.1 SQA group	
2.5.2.2 QA.AB.2 SQA resources	
2.5.2.3 QA.AB.3 SQA training	
2.5.2.4 QA.AB.4 SQA orientation	
2.5.3.1 QA.AC.1 develop SQA plan	
2.5.3.2 QA.AC.2 use SQA plan	
2.5.3.3 QA.AC.3 participate in plans, standards, and procedures	
2.5.3.4 QA.AC.4 review activities	BP 08.01 Ensure Quality: Monitor Conformance to the Defined Process
2.5.3.4 QA.AC.4 review activities	BP 08.01 Monitor Conformance to the Defined Process
2.5.3.5 QA.AC.5 audit designated work products	BP 08.02 Ensure Quality: Measure Quality of the Work Product
2.5.3.6 QA.AC.6 report results to software engineering	
2.5.3.7 QA.AC.7 handle deviations	

2.5.3.8 QA.AC.8 interact with customer SQA	
2.5.4.1 QA.ME.1 measure SQA status	
2.5.5.1 QA.VE.1 senior management review of SQA	
2.5.5.2 QA.VE.2 project manager review of SQA	
2.5.5.3 QA.VE.3 independent expert audits of SQA	
2.6.0.1 CM.GO.1 plan SCM	
2.6.0.2 CM.GO.2 identify and control work products	
2.6.0.3 CM.GO.3 control changes	
2.6.0.4 CM.GO.4 inform of SCM	
2.6.1.1 CM.CO.1 SCM policy	
2.6.2.1 CM.AB.1 control board	
2.6.2.2 CM.AB.2 SCM group	
2.6.2.3 CM.AB.3 SCM resources	
2.6.2.4 CM.AB.4 SCM training	
2.6.2.5 CM.AB.5 engineer SCM training	
2.6.3.01 CM.AC.1 develop SCM plan	BP 09.01 Manage Configurations: Establish Configuration Management Methodology
2.6.3.02 CM.AC.2 use SCM plan	
2.6.3.03 CM.AC.3 establish configuration management library system	BP 09.03 Manage Configurations: Maintain Configuration Data
2.6.3.03 CM.AC.3 establish configuration management library system	BP 09.05 Manage Configurations: Communicate Configuration Status
2.6.3.04 CM.AC.4 identify configuration	BP 09.02 Manage Configurations: Identify Configuration Units
2.6.3.05 CM.AC.5 address change requests and problem reports	BP 08.07 Ensure Quality: Mechanism for corrective actions
2.6.3.06 CM.AC.6 control changes	BP 09.04 Manage Configurations: Control Changes
2.6.3.07 CM.AC.7 release products	
2.6.3.08 CM.AC.8 record configuration status	
2.6.3.09 CM.AC.9 report configuration status	
2.6.3.10 CM.AC.10 audit configuration (baseline)	
2.6.4.1 CM.ME.1 measure status of SCM	
2.6.5.1 CM.VE.1 senior management review of SCM	

2.6.5.2 CM.VE.2 project manager review of SCM	
2.6.5.3 CM.VE.3 configuration audits	
2.6.5.4 CM.VE.4 SQA audits of SCM	
3.1.0.1 PF.GO.1 coordinate improvement	
3.1.0.2 PF.GO.2 assess strengths and weaknesses	
3.1.0.3 PF.GO.3 plan process improvement	
3.1.1.1 PF.CO.1 OPF policy	
3.1.1.2 PF.CO.2 senior management sponsorship of OPF	
3.1.1.3 PF.CO.3 senior management oversight of OPF	
3.1.2.1 PF.AB.1 process group	
3.1.2.2 PF.AB.2 OPF resources	
3.1.2.3 PF.AB.3 required OPF training	
3.1.2.4 PF.AB.4 OPF orientation	
3.1.3.1 PF.AC.1 assess process	BP 08.06 Ensure Quality: Initiate Quality Improvement Activities
3.1.3.1 PF.AC.1 assess process	BP 14.01 Improve Organization's Systems Engineering Processes: Appraise the Process
3.1.3.2 PF.AC.2 develop OPF plan	BP 13.01 Define Organization's Systems Engineering Process: Establish Process Goals
3.1.3.3 PF.AC.3 coordinate improvement	
3.1.3.4 PF.AC.4 coordinate organization's software process database	BP 13.02 Define Organization's Systems Engineering Process: Collect Process Assets
3.1.3.5 PF.AC.5 evaluate new processes, methods, and tools	BP 13.02 Define Organization's Systems Engineering Process: Collect Process Assets
3.1.3.5 PF.AC.5 evaluate new processes, methods, and tools	BP 14.04 Improve Organization's Systems Engineering Processes: Communicate Process Improvements
3.1.3.6 PF.AC.6 coordinate process training	
3.1.3.7 PF.AC.7 inform implementers	BP 13.02 Define Organization's Systems Engineering Process: Collect Process Assets
3.1.3.7 PF.AC.7 inform implementers	BP 14.04 Improve Organization's Systems Engineering Processes: Communicate Process Improvements

3.1.3.7 PF.AC.7 inform implementers	BP 16.07 Manage Systems Engineering Support Environment: Monitor Systems Engineering Support Environment
3.1.4.1 PF.ME.1 measure status of OPF	
3.1.5.1 PF.VE.1 senior management review of OPF	
3.2.0.1 PD.GO.1 establish standard software process for organization	
3.2.0.2 PD.GO.2 inform on OPD	
3.2.1.1 PD.CO.1 OPD policy	
3.2.2.1 PD.AB.1 OPD resources	
3.2.2.2 PD.AB.2 required OPD training	
3.2.3.1 PD.AC.1 develop organization's standard software process	BP 13.03 Define Organization's Systems Engineering Process: Develop Organization's Systems Engineering Process
3.2.3.2 PD.AC.2 document organization's standard software process	
3.2.3.3 PD.AC.3 specify software life cycles	
3.2.3.4 PD.AC.4 provide tailoring guidelines	BP 13.04 Define Organization's Systems Engineering Process: Define Tailoring Guidelines
3.2.3.4 PD.AC.4 provide tailoring guidelines	BP 16.04 Manage Systems Engineering Support Environment: Tailor Systems Engineering Support Environment
3.2.3.5 PD.AC.5 establish organization's software process database	
3.2.3.6 PD.AC.6 establish library of software process-related documentation	
3.2.4.1 PD.ME.1 measure status of OPD	
3.2.5.1 PD.VE.1 SQA audits of OPD	
3.3.0.1 TP.GO.1 plan training	
3.3.0.2 TP.GO.2 provide skills training	
3.3.0.3 TP.GO.3 provide role training	
3.3.1.1 TP.CO.1 TP policy	
3.3.2.1 TP.AB.1 training group	
3.3.2.2 TP.AB.2 TP resources	
3.3.2.3 TP.AB.3 skills and knowledge (required training)	
3.3.2.4 TP.AB.4 TP orientation	
3.3.3.1 TP.AC.1 develop project training plan	BP 17.01 Provide Ongoing Skills and Knowledge: Identify needed skills

3.3.3.2 TP.AC.2 develop organization training plan	BP 17.01 Provide Ongoing Skills and Knowledge: Identify needed skills
3.3.3.3 TP.AC.3 use TP plan	BP 17.05 Provide Ongoing Skills and Knowledge: Train Personnel
3.3.3.4 TP.AC.4 develop organizational training	BP 17.04 Provide Ongoing Skills and Knowledge: Prepare Training Materials
3.3.3.4 TP.AC.4 develop organizational training	BP 17.08 Provide Ongoing Skills and Knowledge: Maintain Training Materials
3.3.3.5 TP.AC.5 establish training waivers	
3.3.3.6 TP.AC.6 maintain training records	BP 17.07 Provide Ongoing Skills and Knowledge: Maintain Training Records
3.3.4.1 TP.ME.1 measure status of TP	
3.3.4.2 TP.ME.2 measure quality of training	BP 17.06 Provide Ongoing Skills and Knowledge: Assess Training Effectiveness
3.3.5.1 TP.VE.1 senior management review of TP	
3.3.5.2 TP.VE.2 independent evaluation of TP	BP 17.06 Provide Ongoing Skills and Knowledge: Assess Training Effectiveness
3.3.5.3 TP.VE.3 audits (QA) of TP	BP 17.06 Provide Ongoing Skills and Knowledge: Assess Training Effectiveness
3.4.0.1 IM.GO.1 tailor defined process	
3.4.0.2 IM.GO.2 plan defined process	
3.4.1.1 IM.CO.1 ISM policy	
3.4.2.1 IM.AB.1 ISM resources	
3.4.2.2 IM.AB.2 required training in tailoring	
3.4.2.3 IM.AB.3 required training in management	
3.4.3.01 IM.AC.1 develop defined process	
3.4.3.02 IM.AC.2 revise defined process	
3.4.3.03 IM.AC.3 plan defined process	
3.4.3.04 IM.AC.4 use defined process	
3.4.3.05 IM.AC.5 use organization's software process database	
3.4.3.06 IM.AC.6 manage size	
3.4.3.07 IM.AC.7 manage effort and costs	
3.4.3.08 IM.AC.8 manage critical computer resources	
3.4.3.09 IM.AC.9 manage critical dependencies	

3.4.3.10 IM.AC.10 manage risks	BP 10.01 Manage Risk: Develop Risk Management Approach
3.4.3.10 IM.AC.10 manage risks	BP 10.02 Manage Risk: Identify Risks
3.4.3.10 IM.AC.10 manage risks	BP 10.03 Manage Risk: Assess Risks
3.4.3.10 IM.AC.10 manage risks	BP 10.04 Manage Risk: Review Risk Assessment
3.4.3.10 IM.AC.10 manage risks	BP 10.05 Manage Risk: Execute Risk Mitigations
3.4.3.10 IM.AC.10 manage risks	BP 10.06 Manage Risk: Track Risk Mitigations
3.4.3.11 IM.AC.11 take corrective action	
3.4.4.1 IM.ME.1 measure effectiveness of defined process	
3.4.5.1 IM.VE.1 senior management review of project	
3.4.5.2 IM.VE.2 project manager review of project	
3.4.5.3 IM.VE.3 SQA audits of ISM	
3.5.0.1 PE.GO.1 define engineering tasks	
3.5.0.2 PE.GO.2 maintain consistent work products	
3.5.1.1 PE.CO.1 SPE policy	
3.5.2.1 PE.AB.1 SPE resources	BP 16.03 Manage Systems Engineering Support Environment: Obtain Systems Engineering Support Environment
3.5.2.1 PE.AB.1 SPE resources	BP 16.06 Manage Systems Engineering Support Environment: Maintain Environment
3.5.2.2 PE.AB.2 required SPE training	
3.5.2.3 PE.AB.3 SPE orientation for engineers	
3.5.2.4 PE.AB.4 SPE orientation for managers	
3.5.3.01 PE.AC.1 integrate methods and tools	BP 16.01 Manage Systems Engineering Support Environment: Maintain Technical Awareness
3.5.3.01 PE.AC.1 integrate methods and tools	BP 16.02 Manage Systems Engineering Support Environment: Determine Support Requirements
3.5.3.02 PE.AC.2 analyze software requirements	BP 06.02 Understand Customer Needs and Expectations: Analyze Needs
3.5.3.02 PE.AC.2 analyze software requirements	BP 06.03 Understand Customer Needs and Expectations: Develop System Requirements

3.5.3.02 PE.AC.2 analyze software requirements	BP 06.04 Understand Customer Needs and Expectations: Obtain Concurrence
3.5.3.03 PE.AC.3 develop software design	
3.5.3.04 PE.AC.4 develop software code	
3.5.3.05 PE.AC.5 perform testing	BP 07.01 Verify and Validate System: Establish Verification and Validation Plans
3.5.3.06 PE.AC.6 perform integration testing	BP 05.05 Integrate System: Verify System Element Interfaces
3.5.3.06 PE.AC.6 perform integration testing	BP 07.02 Verify and Validate System: Define Incremental Verification
3.5.3.07 PE.AC.7 perform system and acceptance testing	BP 05.06 Integrate System: Assemble Aggregates of System Elements
3.5.3.07 PE.AC.7 perform system and acceptance testing	BP 05.07 Integrate System: Check Aggregates of System Elements
3.5.3.07 PE.AC.7 perform system and acceptance testing	BP 07.01 Verify and Validate System: Establish Verification and Validation Plans
3.5.3.07 PE.AC.7 perform system and acceptance testing	BP 07.03 Verify and Validate System: Define System Verification
3.5.3.07 PE.AC.7 perform system and acceptance testing	BP 07.04 Verify and Validate System: Define Validation
3.5.3.07 PE.AC.7 perform system and acceptance testing	BP 07.05 Verify and Validate System: Perform and Capture Verification and Validation
3.5.3.08 PE.AC.8 develop documentation	
3.5.3.09 PE.AC.9 collect defect data	
3.5.3.10 PE.AC.10 maintain consistency across work products	BP 02.09 Derive and Allocate Requirements: Capture Results and Rationale
3.5.4.1 PE.ME.1 measure functionality and quality	BP 11.03 Monitor and Control Technical Effort: Track Technical Parameters
3.5.4.2 PE.ME.2 measure status of SPE	
3.5.5.1 PE.VE.1 senior management review of SPE	
3.5.5.2 PE.VE.2 project manager review of SPE	
3.5.5.3 PE.VE.3 SQA audits of SPE	BP 07.06 Verify and Validate System: Assess Verification and Validation Success
3.6.0.1 IC.GO.1 agree to customer requirements	
3.6.0.2 IC.GO.2 agree to commitments	
3.6.0.3 IC.GO.3 resolve intergroup issues	

3.6.1.1 IC.CO.1 IC policy	
3.6.2.1 IC.AB.1 IC resources	
3.6.2.2 IC.AB.2 compatible support tools	BP 16.03 Manage Systems Engineering Support Environment: Obtain Systems Engineering Support Environment
3.6.2.2 IC.AB.2 compatible support tools	BP 16.06 Manage Systems Engineering Support Environment: Maintain Environment
3.6.2.3 IC.AB.3 required IC training	
3.6.2.4 IC.AB.4 IC orientation for leaders	
3.6.2.4 IC.AB.4 IC orientation for leaders	BP 04.01 Integrate Disciplines: Involve Disciplines
3.6.2.5 IC.AB.5 IC orientation for engineers	BP 04.02 Integrate Disciplines: Foster Cross-Discipline Understanding
3.6.2.5 IC.AB.5 IC orientation for engineers	BP 04.02 Integrate Disciplines: Train Inter Roles
3.6.3.1 IC.AC.1 participate with customer and end users	BP 02.01 Derive and Allocate Requirements: Develop Detailed Operational Concept
3.6.3.1 IC.AC.1 participate with customer and end users	BP 02.02 Derive and Allocate Requirements: Identify Key Requirement Issues
3.6.3.1 IC.AC.1 participate with customer and end users	BP 06.01 Understand Customer Needs and Expectations: Elicit Needs
3.6.3.2 IC.AC.2 work with other engineering groups	BP 02.03 Derive and Allocate Requirements: Partition Functions
3.6.3.2 IC.AC.2 work with other engineering groups	BP 02.05 Derive and Allocate Requirements: Develop Interface Requirements
3.6.3.2 IC.AC.2 work with other engineering groups	BP 02.08 Derive and Allocate Requirements: Maintain Requirement Sufficiency and Traceability
3.6.3.2 IC.AC.2 work with other engineering groups	BP 04.04 Integrate Disciplines: Establish Resolution Methods
3.6.3.2 IC.AC.2 work with other engineering groups	BP 04.06 Integrate Disciplines: Develop and Communicate Project Goals
3.6.3.2 IC.AC.2 work with other engineering groups	BP 11.04 Monitor and Control Technical Effort: Review Project Performance
3.6.3.2 IC.AC.2 work with other engineering groups	BP 11.06 Monitor and Control Technical Effort: Control Technical Effort
3.6.3.2 IC.AC.2 work with other engineering groups	BP 02.06 Derive and Allocate Requirements: Allocate Requirements
3.6.3.2 IC.AC.2 work with other engineering groups (1)	BP 05.01 Integrate System: Define Interfaces

3.6.3.3 IC.AC.3 develop defined process plan	BP 04.03 Integrate Disciplines: Establish Coordination Methods
3.6.3.4 IC.AC.4 manage critical dependencies	
3.6.3.5 IC.AC.5 accept internal work products	BP 05.03 Integrate System: Verify Receipt of System Elements
3.6.3.5 IC.AC.5 accept internal work products	BP 05.04 Integrate System: Verify System Element Correctness
3.6.3.5 IC.AC.5 accept internal work products	BP 15.04 Manage Product Line Evolution: Ensure Critical Component Availability
3.6.3.6 IC.AC.6 handle intergroup issues	
3.6.3.7 IC.AC.7 conduct technical interchanges	BP 04.05 Integrate Disciplines: Communicate Results
3.6.4.1 IC.ME.1 measure status of IC	
3.6.5.1 IC.VE.1 senior management review of IC	
3.6.5.2 IC.VE.2 project manager review of IC	
3.6.5.3 IC.VE.3 SQA audits of IC	
3.7.0.1 PR.GO.1 plan peer reviews	
3.7.0.2 PR.GO.2 remove defects	
3.7.1.1 PR.CO.1 PR policy	
3.7.2.1 PR.AB.1 PR resources	
3.7.2.2 PR.AB.2 required PR training for leaders	
3.7.2.3 PR.AB.3 required PR training for reviewers	
3.7.3.1 PR.AC.1 develop PR plan	
3.7.3.2 PR.AC.2 perform peer reviews	
3.7.3.3 PR.AC.3 record peer review data	
3.7.4.1 PR.ME.1 measure status of PR	
3.7.5.1 PR.VE.1 SQA audits of PR	
4.1.0.1 QP.GO.1 plan QPM	
4.1.0.2 QP.GO.2 control process performance quantitatively	
4.1.0.3 QP.GO.3 determine process capability	
4.1.1.1 QP.CO.1 project QPM policy	
4.1.1.2 QP.CO.2 organization QPM policy	
4.1.2.1 QP.AB.1 QPM group	
4.1.2.2 QP.AB.2 QPM resources	
4.1.2.3 QP.AB.3 support analyzing data	
4.1.2.4 QP.AB.4 required QPM training	

4.1.2.5 QP.AB.5 QPM orientation	
4.1.3.1 QP.AC.1 develop QPM plan	
4.1.3.2 QP.AC.2 use QPM plan	BP 08.03 Ensure Quality: Measure Quality of the Process
4.1.3.3 QP.AC.3 determine data analysis strategy	BP 08.04 Ensure Quality: Analyze Quality Measurements
4.1.3.4 QP.AC.4 collect measurement data	
4.1.3.5 QP.AC.5 control defined software process	
4.1.3.6 QP.AC.6 report results of QPM	
4.1.3.7 QP.AC.7 establish organizational process capability baseline	BP 13.01 Define Organization's Systems Engineering Process: Establish Process Goals
4.1.4.1 QP.ME.1 measure status of QPM	
4.1.5.1 QP.VE.1 senior management review of QPM	
4.1.5.2 QP.VE.2 project manager review of QPM	
4.1.5.3 QP.VE.3 SQA audits	
4.2.0.1 QM.GO.1 plan SQM	
4.2.0.2 QM.GO.2 define quality goals	
4.2.0.3 QM.GO.3 manage progress toward quality goals	
4.2.1.1 QM.CO.1 SQM policy	
4.2.2.1 QM.AB.1 SQM resources	
4.2.2.2 QM.AB.2 required SQM training for managers	
4.2.2.3 QM.AB.3 required SQM training for engineers	
4.2.3.1 QM.AC.1 develop SQM plan	
4.2.3.2 QM.AC.2 use SQM plan	
4.2.3.3 QM.AC.3 define quality goals	
4.2.3.4 QM.AC.4 measure product quality	
4.2.3.5 QM.AC.5 allocate subcontractor quality goals	
4.2.4.1 QM.ME.1 measure status of SQM	
4.2.5.1 QM.VE.1 senior management review of SQM	
4.2.5.2 QM.VE.2 project manager review of SQM	
4.2.5.3 QM.VE.3 SQA audits of SQM	
5.1.0.1 DP.GO.1 plan DP	
5.1.0.2 DP.GO.2 identify common defect causes	

5.1.0.3 DP.GO.3 eliminate common defect causes	
5.1.1.1 DP.CO.1 organization DP policy	
5.1.1.2 DP.CO.2 project DP policy	
5.1.2.1 DP.AB.1 organization DP team	
5.1.2.2 DP.AB.2 project DP team	
5.1.2.3 DP.AB.3 DP resources	
5.1.2.4 DP.AB.4 required DP training	
5.1.3.1 DP.AC.1 develop DP plan	BP 08.06 Ensure Quality: Initiate Quality Improvement Activities
5.1.3.2 DP.AC.2 conduct DP kick-off meeting	
5.1.3.3 DP.AC.3 conduct causal analysis	
5.1.3.4 DP.AC.4 coordinate DP activities	
5.1.3.5 DP.AC.5 document DP data	
5.1.3.6 DP.AC.6 revise organization's standard software process	
5.1.3.7 DP.AC.7 revise defined software process	
5.1.3.8 DP.AC.8 receive feedback on DP	
5.1.4.1 DP.ME.1 measure status of DP	
5.1.5.1 DP.VE.1 senior management review of DP	
5.1.5.2 DP.VE.2 project manager review of DP	
5.1.5.3 DP.VE.3 SQA audits of DP	
5.2.0.1 TM.GO.1 plan TCM	
5.2.0.2 TM.GO.2 evaluate new technologies	
5.2.0.3 TM.GO.3 transfer new technologies	
5.2.1.1 TM.CO.1 TCM policy	
5.2.1.2 TM.CO.2 senior management sponsorship of TCM	
5.2.1.3 TM.CO.3 senior management oversight of TCM	
5.2.2.1 TM.AB.1 TCM group	
5.2.2.2 TM.AB.2 TCM resources	
5.2.2.3 TM.AB.3 support analyzing technology change	
5.2.2.4 TM.AB.4 technology change data	
5.2.2.5 TM.AB.5 required TCM training	
5.2.3.1 TM.AC.1 develop TCM plan	BP 15.04 Manage Product Line Evolution: Ensure Critical Component Availability

5.2.3.2 TM.AC.2 work with projects	BP 15.01 Manage Product Line Evolution: Define Product Evolution
5.2.3.2 TM.AC.2 work with projects	BP 15.02 Manage Product Line Evolution: Identify New Product Technologies
5.2.3.3 TM.AC.3 inform of TCM	
5.2.3.4 TM.AC.4 analyze need for technology change	
5.2.3.4 TM.AC.4 analyze need for technology change	BP 15.02 Manage Product Line Evolution: Identify New Product Technologies
5.2.3.5 TM.AC.5 select technologies	BP 15.02 Manage Product Line Evolution: Identify New Product Technologies
5.2.3.5 TM.AC.5 select technologies	BP 16.03 Manage Systems Engineering Support Environment: Obtain Systems Engineering Support Environment
5.2.3.6 TM.AC.6 pilot new technology	BP 15.03 Manage Product Line Evolution: Adapt Development Processes
5.2.3.7 TM.AC.7 incorporate new organizational technologies	BP 15.03 Manage Product Line Evolution: Adapt Development Processes
5.2.3.7 TM.AC.7 incorporate new organizational technologies	BP 15.04 Manage Product Line Evolution: Ensure Critical Component Availability
5.2.3.7 TM.AC.7 incorporate new organizational technologies	BP 15.05 Manage Product Line Evolution: Manage Product Technology Insertion
5.2.3.7 TM.AC.7 incorporate new organizational technologies	BP 16.05 Manage Systems Engineering Support Environment: Insert New Technology
5.2.3.8 TM.AC.8 incorporate new project technologies	BP 15.05 Manage Product Line Evolution: Manage Product Technology Insertion
5.2.4.1 TM.ME.1 measure status of TCM	
5.2.5.1 TM.VE.1 senior management review of TCM	
5.2.5.2 TM.VE.2 SQA audits of TCM	
5.3.0.1 PC.GO.1 plan PCM	
5.3.0.2 PC.GO.2 organization-wide participation	
5.3.0.3 PC.GO.3 continual improvement	
5.3.1.1 PC.CO.1 PCM policy	
5.3.1.2 PC.CO.2 senior management sponsorship of PCM	
5.3.2.1 PC.AB.1 PCM resources	
5.3.2.2 PC.AB.2 required PCM training for managers	
5.3.2.3 PC.AB.3 required PCM training for engineers (SWE mgr)	

5.3.2.4 PC.AB.4 required PCM training for senior management	
5.3.3.01 PC.AC.1 establish empowered improvement	
5.3.3.02 PC.AC.2 coordinate improvement	
5.3.3.03 PC.AC.3 develop PCM plan	
5.3.3.04 PC.AC.4 use PCM plan	BP 14.02 Improve Organization's Systems Engineering Processes: Plan Process Improvements
5.3.3.05 PC.AC.5 handle improvement proposals	
5.3.3.06 PC.AC.6 participate in improvements	BP 08.05 Ensure Quality: Obtain employee participation
5.3.3.07 PC.AC.7 pilot improvements	
5.3.3.08 PC.AC.8 transfer improvements	BP 14.03 Improve Organization's Systems Engineering Processes: Change the Standard Process
5.3.3.09 PC.AC.9 maintain improvement records	
5.3.3.10 PC.AC.10 receive feedback on improvement	BP 14.04 Improve Organization's Systems Engineering Processes: Communicate Process Improvements
5.3.3.10 PC.AC.10 receive feedback on improvement	BP 16.07 Manage Systems Engineering Support Environment: Monitor Systems Engineering Support Environment
5.3.4.1 PC.ME.1 measure status of PCM	
5.3.5.1 PC.VE.1 senior management review of PCM	
5.3.5.2 PC.VE.2 SQA audits of PCM	
	BP 01.01 Analyze Candidate Solutions: Establish Evaluation Criteria
	BP 01.02 Analyze Candidate Solutions: Define Analysis Approach
	BP 01.03 Analyze Candidate Solutions: Identify Additional Alternatives
	BP 01.04 Analyze Candidate Solutions: Analyze Candidate Solutions
	BP 01.05 Analyze Candidate Solutions: Select Solution
	BP 01.06 Analyze Candidate Solutions: Capture Results
	BP 02.04 Derive and Allocate Requirements: Derive Requirements

	BP 03.01 Evolve System Architecture: Derive the System Architecture Requirements
	BP 03.02 Evolve System Architecture: Identify Key Design Issues
	BP 03.03 Evolve System Architecture: Develop System Structure
	BP 03.05 Evolve System Architecture: Allocate System Requirements
	BP 03.06 Evolve System Architecture: Maintain Requirement Sufficiency and Traceability
	BP 03.07 Evolve System Architecture: Capture Results and Rationale
	BP 03.08 Evolve System Architecture: Identify post-development requirements
	BP 17.02 Provide Ongoing Skills and Knowledge: Select mode of acquiring knowledge
	BP 17.03 Provide Ongoing Skills and Knowledge: Ensure skill available

Effective CMM-Based Process Improvement

Mark C. Paulk, Software Engineering Institute, USA

Abstract
The Capability Maturity Model^{SM} for Software developed by the Software Engineering Institute has had a major influence on software process and quality improvement around the world. Although the CMM^{SM} has been widely adopted, there remain many misunderstandings about how to use it effectively for business-driven software process improvement. This paper discusses how to use the CMM correctly and effectively. It also discusses aspects of successful process improvement efforts that are not explicitly addressed by the CMM, but which are critical to achieving business and process improvement goals.

Keywords: CMM, Capability Maturity Model, software process, process improvement.

1. Introduction

The Capability Maturity Model^{SM} for Software (CMM^{SM} or SW-CMM[1]) developed by the Software Engineering Institute (SEI) has had a major influence on software process and quality improvement around the world [Paulk95]. The SW-CMM defines a five-level framework for how an organization matures its software process. These levels describe an evolutionary path from ad hoc, chaotic processes to mature, disciplined software processes. The five levels, and the 18 key process areas that describe them in detail, are summarized in Figure 1.

The five levels can be briefly described as:

1) Initial	The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.

This paper was published as in **Proceedings of the 6th International Conference on Software Quality**, Ottawa, Canada, 28-31 October 1996, pp. 226-237.

[1] Because of the proliferation of capability maturity models inspired by the success of the CMM, we are starting to use the acronym SW-CMM when referring to the CMM for Software.

2) Repeatable	Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
3) Defined	Management and engineering activities are documented, standardized, and integrated into a family of standard software processes for the organization. Projects use a tailored version of the organization's standard software processes for developing and maintaining software.
4) Managed	Detailed measures of the software process and product quality are collected. Software processes and products are quantitatively understood and controlled.
5) Optimizing	Continuous process improvement is facilitated by quantitative feedback from the process and from piloting innovative ideas and technologies.

The key process areas are satisfied by achieving goals, which are described by key practices, subpractices, and examples. The rating components of the SW-CMM are maturity levels, key process areas, and goals. The other components are informative and provide guidance on how to interpret the model.

Level	Focus	Key Process Areas
5 Optimizing	*Continuous process improvement*	Defect Prevention Technology Change Management Process Change Management
4 Managed	*Product and process quality*	Quantitative Process Management Software Quality Management
3 Defined	*Engineering processes and organizational support*	Organization Process Focus Organization Process Definition Training Program Integrated Software Management Software Product Engineering Intergroup Coordination Peer Reviews
2 Repeatable	*Project management processes*	Requirements Management Software Project Planning Software Project Tracking & Oversight Software Subcontract Management Software Quality Assurance Software Configuration Management
1 Initial	*Competent people and heroics*	

Figure 1. The key process areas in the SW-CMM.

While the SW-CMM has been very influential around the world in inspiring and guiding software process improvement (SPI), it has also been misused and abused by some and not used effectively by others. Although the SEI's work has its critics [Bach94, Jones95], there is a growing body of data indicating the power of CMM-based process improvement [Herbsleb94, Lawlis95]. The purpose of this paper is to discuss the correct, effective use of the SW-CMM and to recommend specific software engineering and management practices that map to the SW-CMM, as noted in the footnotes. The recommendations in this paper reflect my personal opinion, based on my experience in working with a number of "world-class" software organizations.

Ideally, effective use of the SW-CMM should build an organization that can dynamically adapt to a rapidly changing, even chaotic, environment; an organization that knows what business it is in and pursues software projects aligned with its strategic business objectives; a learning organization that explicitly, rather than implicitly, captures knowledge; an organization managed by facts rather than intuition, while still valuing creativity; an organization that empowers its most crucial asset: its people.

2. Effective CMM-Based Process Improvement

Using the SW-CMM correctly means balancing conflicting objectives. CMM-based appraisals require the use of professional judgment. Although the SW-CMM provides a significant amount of guidance in making these judgments, removing subjectivity implies a deterministic, repetitive process that is not characteristic of engineering design work.

The SW-CMM is sometimes referred to as a set of process requirements, but it does not contain any "shall" statements. The goals, key practices, and subpractices of its key process areas are arranged in a hierarchy useful for interpreting or tailoring the SW-CMM. As Charlie Weber has pointed out, "To tailor a goal, you should sweat blood (and then be conservative in mapping the relationship to "the" CMM). To tailor a key practice, you should just sweat. To tailor a subpractice, your conscience should hardly bother you."

To a knowledgeable software professional, many of these recommendations may seem obvious. I fear, however, that the consistent implementation of even well-known good engineering and management practices is far from common.

2.1 Management Sponsorship

Trite though it may seem, obtaining senior management sponsorship is a crucial component of building organizational capability.[2] As individuals, we can exercise professionalism and discipline within our sphere of control, but if an organization as a whole is to change its performance, then its senior management

[2] Organization Process Focus, Commitment 2; Process Change Management, Commitment 2.

must actively support the change. Bottom-up SPI, without sponsorship and coordination, leads to islands of excellence rather than predictably improved organizational capability. Effective sponsorship will only occur, however, when there is significant dissatisfaction with the status quo.

Sponsorship involves such activities as setting policies[3] and providing resources for process work,[4] but the most crucial factor is that senior management believe in the strategic importance of process improvement, demonstrate their support, and pay attention.[5] If, on the other hand, management provides annual speeches on the importance of quality and then returns to the "really important" issues of cost and schedule, then that message will be received also.

"Paying attention" may lead to the Hawthorne Effect: people increase their efforts as a result of the attention, and productivity and quality could improve even if the work environment becomes worse. Sustaining improvement over an extended period, however, implies systematic change aligned with the business objectives of the organization.

If process improvement is a true priority, then management will monitor it closely.[6] Management should set aggressive improvement goals, while at the same time recognizing that achieving higher levels of software process maturity is incremental and requires a long-term commitment to continuous process improvement. It may be useful to view software process improvement as a project, with goals, plans for achieving those goals, and management of progress. Do not forget, however, that changing the culture requires the long view. CMM-based improvement is most effective within a systematic approach to SPI, such as the SEI's IDEAL^{SM} approach.[7]

2.2 Commitments and Management by Fact

Management by fact is a paradigm shift for most organizations, which must be based on a measurement foundation.[8] To make data analysis useful, you need to understand what the data means and how to analyze it meaningfully. Begin by collecting a simple set of useful data (I recommend the Goal / Question / Metric approach [Basili92].). Then deploy standard definitions for typical measures[9] that support aggregating and comparing data from different environments.

3 Policy practices in Commitment to Perform.

4 Organization Process Focus, Abilities 1 and 2; in general, the resource practices in Ability to Perform.

5 Organization Process Focus, Commitment 3; management oversight practices in Verifying Implementation.

6 Organization Process Focus, Verification 1.

7 IDEAL stands for Initiate, Diagnose, Establish, Act, and Leverage.

8 Measurement practices in Measurement and Analysis.

9 Organization Process Definition, Activity 5.

A corollary to collecting data is to believe what it tells you. If you have historically developed 100 lines of code per month, “betting the company” that you can bring in a critical project at 1000 lines of code per month is almost certainly unwise. And changing the size estimates without changing commitments and functionality is not likely to help matters.

You also have to be sensitive to the potential for causing dysfunctional behavior by what you measure [Austin94]. People will focus their efforts where management is paying attention. If management only pays attention to schedule, then quality is likely to suffer. The act of measuring identifies what is important, but some things are difficult to measure. Management needs to ensure that attention is visibly paid to all critical aspects of the project, including those difficult to measure, not just those it is easy to measure and track. The corollary to believing the data is to recognize that no limited set of data can completely capture a complex operation.

Establish an internal commitment process[10] based on what you know you can do. If the workers buy in to a set of commitments as being realistic, even if aggressive, then they have made a personal commitment. If they consider the stated commitments as being a management pipe dream, they are likely to consider the resulting problems to be management’s. This implies that management should seek worker feedback when establishing commitments.

2.3 Process Focus

In most organizations, a software engineering process group (SEPG) or some equivalent should be formed to coordinate process definition, improvement, and deployment activities.[11] The SEPG should be formed early, e.g., in the Initiating phase of the IDEAL approach, and be staffed with competent and respected individuals possessing both managerial and technical skills. It is crucial that these individuals have good interpersonal skills. Success of the SEPG depends on their ability to communicate, teach, negotiate, and consult [Mogilensky94].

One of the reasons for dedicating resources to an SEPG is to ensure follow-through on appraisal findings.[12] Many improvement programs have foundered simply because no action resulted from the appraisal. Improvement comes from action planning, assigning individuals to do the work, piloting and deploying improved processes, and management oversight throughout.

It is desirable to have part-time participants on process improvement teams. In saying that SEPGs should coordinate process activities, the word “coordinate”

[10] Software Project Planning, Activities 1-3.

[11] Organization Process Focus, Ability 1.

[12] Organization Process Focus, Activity 1.

was deliberately chosen. The skills and knowledge of the organization's best people should be brought to bear on addressing its problems. In a world-class organization, I would expect everyone to participate in process and quality improvement activities.[13] Worker participation and empowerment enable successful process improvement.

Begin with the "as is" process, not the "should be" process, to leverage effective practices and co-opt resisters. Mandating top-down that everyone will follow the new "should be" process, particularly if not developed by empowered workers, is a common recipe for failure. The "as is" process evolved because the people doing the work needed to get the job done – even if that meant going around the system. The "should be" process may, or may not, be feasible in the given culture and environment. With an organizational focus on process management and improvement, the "as is" and "should be" processes will converge, resulting in organizational learning.

It is important to remember that software process improvement occurs in a business context. There may be many other crucial business issues being worked at the same time; there may even be a Total Quality Management (TQM) initiative under way. Since CMM-based improvement is an application of TQM principles to software, the synergy of aligning these initiatives seems obvious. Some of the most effective SPI programs I have seen operated under the umbrella of a TQM initiative – and in the instances where I have observed disjoint TQM and SPI programs, both were ineffective.

2.4 Useful Processes

Document your processes.[14] The reasons for documenting a process (or product) are 1) to communicate – to others now and perhaps to yourself later; 2) to understand – if you can't write it down, you don't really understand; and 3) to encourage consistency – take advantage of repeatability.

Building software is a design-intensive, creative activity. While the discipline of process is a crucial enabler of success, the objective is to solve a problem, and this requires creativity. Software processes should be repeatable, even if they are not repetitive. The balance between discipline and creativity can be challenging [Glass95]. Losing sight of the creative, design-intense nature of software work leads to stifling rigidity. Losing sight of the need for discipline leads to chaos.

Processes need to be tailored to the needs of the project [Ginsberg95].[15] Although standard processes provide a foundation, each project will also have

---

[13] Process Change Management, Goal 2.

[14] Organization Process Definition.

[15] Organization Process Definition, Activity 4.

unique needs. Unreasonable constraints on tailoring can lead to significant resistance to following the process.

The following list contains some process attributes that are important for useful and effective processes:

- documented – but keep it simple
- trained – impart the skill to use it
- practiced – used even in a crisis
- measured – simple measures for insight into critical questions
- owned – by a responsible party
- maintained – updated as needed
- supported – in plans and by management expectations and rewards
- controlled – process changes are disciplined
- verified and validated – checked for correct execution

Keep the process simple because we live in a rapidly changing world. Processes do not need to be lengthy or complex. The SW-CMM is about doing things, not having things. A 1-2 page process description may suffice [Humphrey95], and subprocesses and procedures[16] can be invoked as needed and useful. Order is not created by complex controls, but by the presence of a few guiding formulae or principles [Wheatley92, page 11].

2.5 Training

Documented processes are of little value if they are not effectively deployed. Training, via a wide variety of mechanisms, is critical to consistent and effective software engineering and management.[17] Training is an investment in our greatest asset – our people [Curtis95].

The reason for training is to develop skills. There are many “training mechanisms” other than formal classroom training that can be effective in building skills. One that should be seriously considered is a formal mentoring program. In this case, formality means going beyond assigning a mentor and hoping that experience will rub off. Formality implies training people on how to mentor and monitoring the effectiveness of the mentoring. For example, the Onboard Shuttle project considers mentoring in doing causal analysis of defects [Paulk95].

[16] Process descriptions are frequently composed of subprocesses, methods, and/or procedures, which in turn may be supported by automated tools. Procedures may be lengthy, but they are also usually deterministic.
[17] Training Program.

Management training is particularly important because ineffective management can cripple a good team.[18] People who are promoted to management because of their technical skills have to acquire a new set of skills, including interpersonal skills [Mogilensky94, Weinberg94].

Having taken the Personal Software Process (PSP) course, I can highly recommend it for building self-discipline [Humphrey95]. Note that the effect of reading the book is not the same as taking the course and doing the work! Where the SW-CMM addresses the organizational side of process improvement, PSP addresses building the capability of individual practitioners. The PSP course convinces the individual, based on his or her own data, of the value of a disciplined, engineering approach to building software. Even if the organization is not interested in PSP, I would recommend it for professional development; applying PSP can be a survival skill in a level 1 organization.

2.6 Risk Management

Some argue that software project management is really risk management.[19] In one sense, the SW-CMM is about managing risk. We attempt to establish stable requirements[20] so that we can plan[21] and manage[22] effectively, but the business environment changes rapidly, perhaps chaotically. We try to establish an island of order in the sea of software chaos, but both order and chaos have a place. As Wheatley suggests, "To stay viable, open systems maintain a state of non-equilibrium, keeping the system in balance so that it can change and grow." [Wheatley92, page 78] Although we can establish processes that help us manage the risks of a chaotic world, we also need to change and grow.

This implies that you should use an incremental or evolutionary life cycle.[23] If you want to focus on risk management, the spiral model may be the preferred life cycle model. If you want to focus on involving the customer, perhaps rapid prototyping or joint application design would be preferable. Few long-term projects have the luxury of the stable environment necessary for the waterfall life cycle to be the preferred choice – yet it is probably the most common life cycle. (Note that for small projects, the waterfall life cycle may be an excellent choice.)

---

18 Software Project Planning, Ability 4; Software Project Tracking and Oversight, Ability 4; Integrated Software Management, Ability 3.

19 Software Project Planning, Activity 13; Software Project Tracking and Oversight, Activity 10; Integrated Software Management, Activity 10.

20 Requirements Management.

21 Software Project Planning.

22 Software Project Tracking and Oversight; Integrated Software Management.

23 Software Project Planning, Activity 5.

2.7 Customer-Supplier Relationship

Establish a good, working relationship with the customer and end user based on open communication and integrity.[24] This requires cooperation from the customer, and the danger of customer irrationality is difficult to overstate. Building good customer-supplier relationships is a long-term effort, but if the organization has (or wants to have) a stable customer base, the customer must appreciate the complexities of software engineering just as the supplier must understand the application domain and business environment for the software product. Even those organizations that operate in an environment where customer "churn" is the norm can profit from a (deserved) reputation for quality and integrity.

Talk to the customer. In the exercises for our Introduction to the CMM training, one of our most common observations is that the teams do not talk to their customer. This is crucial in making good cost, schedule, functionality, and quality tradeoff decisions. In a study done for one multinational company, which focused on leading-edge technology as its primary competitive advantage, the #1 and #2 issues for satisfying customer expectations were "quality" and "meeting commitments." "Technology" was #7.

One of the drivers behind software capability evaluations (SCE) is the customer's desire for predictable costs and schedules. Communication is critical for setting (and changing) customer expectations. Our contact at one of the early SCE pilot organizations commented later that he considered the improved customer-supplier communication to have been worth the (nontrivial) cost of doing SCEs, even if there had been no other improvement.

And, of course, be a good customer yourself when the tables are turned.

2.8 Peer Reviews

Although you can argue over the best kind of peer review, the simple fact is that the benefits of peer reviews far outweigh their costs.[25] The data suggests some form of inspection should be used [Ackerman89], but any form of collegial or disciplined review, such as structured walkthroughs, adds significant value.

Some time ago, one of my colleagues had an interesting question about peer reviews. He had been consulting with a company that had its supervisors review code. This had been successful for them and was firmly entrenched in their culture. His question was whether this satisfied the Peer Reviews key process

[24] Requirements Management; Software Project Tracking and Oversight, Activity 13; Software Quality Assurance, Activity 8; Intergroup Coordination, Goal 1, Activity 1.

[25] Peer Reviews.

area. My answer was that it did not. Peers are not supervisors, and having only one person do the review seems inadequate.

My colleague later reported that the company had performed an experiment comparing their supervisor reviews with peer reviews. They discovered that peer reviews found significantly more errors than supervisor reviews. They also discovered that supervisor reviews tended to identify "form errors" while peer reviews tended to identify "content errors." They also performed reviews with designers as well as "code-level peers" and found that these reviews identified more (and more design-oriented) errors than the code-level peer reviews.[26]

The significant point to me was that they experimented to find a data-based answer to an open issue. They did not take a consultant's advice on faith. I consider this one of the hallmarks of a learning organization, one that can take advantage of the SW-CMM effectively.

3. Conclusion

The SW-CMM represents a "common sense engineering" approach to software process improvement. Its maturity levels, key process areas, goals, and key practices have been extensively discussed and reviewed within the software community. While the SW-CMM is neither perfect nor comprehensive [Curtis95, Bate95], it does represent a broad consensus of the software community and is a useful tool for guiding SPI efforts.

Never forget why process improvement is important. Standards and models such as the SW-CMM can help organizations improve their software process, but focusing on achieving a maturity level without really improving the underlying process is a danger. Maturity levels should be a measure of improvement, not the goal of improvement. What are the business needs and business goals of the improvement effort? What is the impact on cost, cycle time, productivity, quality, and – most importantly – customer satisfaction?

4. References

Ackerman89	A.F. Ackerman, L.S. Buchwald, and F.H. Lewski, "Software Inspections: An Effective Verification Process," IEEE Software, Vol. 6, No. 3, May 1989, pp. 31-36.
Austin94	Robert D. Austin, "Theories of Measurement and Dysfunction in Organizations," PhD Dissertation, Department of Social and

[26] I think most peer review experts would agree that including designers in a peer review is a good idea. Including supervisors may be acceptable also, but there is a potential for dysfunction if they do performance reviews and are responsible for raises and promotions.

	Decision Sciences, Carnegie Mellon University, 10 September 1994.
Bach94	James Bach, "The Immaturity of the CMM," American Programmer, Vol. 7, No. 9, September 1994, pp. 13-18.
Basili92	V.R. Basili, "Software Modeling and Measurement: The Goal / Question / Metric Paradigm," University of Maryland, CS-TR-2956, UMIACS-TR-92-96, 1992.
Bate95	Roger Bate, Dorothy Kuhn, Curt Wells, et al, "A Systems Engineering Capability Maturity Model, Version 1.1," Software Engineering Institute, CMU / SEI-95-MM-003, November 1995.
Curtis95	Bill Curtis, William E. Hefley, and Sally Miller, "People Capability Maturity Model," Software Engineering Institute, CMU / SEI-95-MM-02, September 1995.
Ginsberg95	Mark Ginsberg and Lauren Quinn, "Process Tailoring and the Software Capability Maturity Model," Software Engineering Institute, CMU / SEI-94-TR-024, November 1995.
Glass95	Robert L. Glass, **Software Creativity**, Prentice Hall, Englewood Cliffs, NJ, 1995.
Herbsleb94	James Herbsleb, Anita Carleton, et al., "Benefits of CMM-Based Software Process Improvement: Initial Results," Software Engineering Institute, CMU / SEI-94-TR-13, August 1994.
Humphrey95	Watts S. Humphrey, **A Discipline for Software Engineering**, ISBN 0-201-54610-8, Addison-Wesley Publishing Company, Reading, MA, 1995.
Jones95	Capers Jones, "The SEI's CMM – Flawed?," Software Development, Vol. 3, No. 3, March 1995, pp. 41-48.
Lawlis95	Patricia K. Lawlis, Robert M. Flowe, and James B. Thordahl, "A Correlational Study of the CMM and Software Development Performance," Crosstalk: The Journal of Defense Software Engineering, Vol. 8, No. 9, September 1995, pp. 21-25.
Mogilensky94	Judah Mogilensky and Betty L. Deimel, "Where Do People Fit in the CMM?," American Programmer, Vol. 7, No. 9, September 1994, pp. 36-43.

Paulk95	Carnegie Mellon University, Software Engineering Institute (Principal Contributors and Editors: Mark C. Paulk, Charles V. Weber, Bill Curtis, and Mary Beth Chrissis), **The Capability Maturity Model: Guidelines for Improving the Software Process**, ISBN 0-201-54664-7, Addison-Wesley Publishing Company, Reading, MA, 1995.
Weinberg94	Gerald M. Weinberg, **Quality Software Management, Volume 3: Congruent Action**, ISBN 0-932633-28-5, Dorset House, New York, NY, 1994.
Wheatley92	Margaret J. Wheatley, **Leadership and the New Science**, Berrett-Koehler Publishers, San Francisco, CA, 1992.

For Further Information

SEI Customer Relations
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-5800
Internet: customer-relations@sei.cmu.edu

The SEI Web page is
http://www.sei.cmu.edu/

For information specifically on the SW-CMM, visit
http://www.sei.cmu.edu/technology/cmm/

Acknowledgments

I would like to thank Donna Dunaway, Watts Humphrey, Susanne Kelly, Bill Peterson, Charlie Weber, and Dave Zubrow for their comments on this paper.

Top-Level Standards Map

ISO 12207, ISO 15504 (Jan 1998 TR), Software CMM v1.1 and v2 Draft C

Done 23 February 1998

The mapping between ISO 12207 and ISO 15504 comes from Annex A in ISO 15504. This mapping is at the process level (although a more detailed mapping of the development process subprocesses is included because of the potential impact on Software CMM v2). Items in (parentheses) indicate a judgmental or inferential relationship, rather than a direct relationship processes and key process areas.

This mapping shows how a set of topics in one document "lie on top" of the equivalent topics in another. Topics are typically not isomorphic but are highly correlated. Anyone adequately implementing, for example, the Configuration Management Process in ISO 12207 or ISO 15504 could reasonably expect to have satisfied the Software Configuration Management key process area in the Software CMM. Topics are not usually isomorphic because of extensions that may be added (e.g., in ISO 15504 in comparison to ISO 12207) or different levels of abstraction that may have been chosen (e.g., the Development Process in ISO 12207 addresses the same set of concerns as the Software Product Engineering key process area in the Software CMM; the Maintenance Process in ISO 12207 is addressed as a subpractice in Software CMM v2C since maintenance is considered a project environment where all of the CMM key process areas are appropriately implemented).

ISO 12207	**ISO 15504**	**SW-CMM v1.1**	**SW-CMM v2 Draft C**
5. Primary life cycle processes			
5.1 Acquisition process	CUS.1 Acquisition process	Software Subcontract Management	Software Acquisition Management
5.2 Supply process	CUS.2 Supply process[1]	(Software Project Planning; Software Project Tracking & Oversight; Software Product Engineering)	(Software Project Planning; Software Project Control; Software Product Engineering)
	CUS.3 Requirements elicitation process		Software Product Engineering, Activity 2
5.3 Development process	ENG.1 Development process	Software Product Engineering	Software Product Engineering
5.3.1 Process implementation	*ENG.1 Development process*	*Software Product Engineering*	*Software Product Engineering*
5.3.2 System requirements analysis	*ENG.1.1 System requirements analysis and design process*		*(Software Product Engineering, Activity 2)*[2]
5.3.3 System architectural design	*ENG.1.1 System requirements analysis and design process*		*(Software Product Engineering, Activity 2)*[3]
5.3.4 Software requirements analysis	*ENG.1.2 Software requirements analysis process*	*Software Product Engineering, Activity 2*	*Software Product Engineering, Activity 3*

[1] The Supply Process deals with providing software to the customer that meets the agreed requirements. Establishing a contract, developing the software, and delivering it to the customer, which are the issues for this process, are addressed in various KPAs, although the Supply Process itself is not explicitly specified in the SW-CMM.

[2] Although not explicitly called out as system requirements analysis, PE.AC.2 will frequently be implemented as such.

[3] Although not explicitly called out as system requirements analysis, PE.AC.2 will frequently be implemented as such.

ISO 12207	**ISO 15504**	**SW-CMM v1.1**	**SW-CMM v2 Draft C**
5.3.5 Software architectural design	*ENG.1.3 Software design process*	*Software Product Engineering, Activity 3*	*Software Product Engineering, Activity 4*
5.3.6 Software detailed design	*ENG.1.3 Software design process*	*Software Product Engineering, Activity 3*	*Software Product Engineering, Activity 4*
5.3.7 Software coding and testing	*ENG.1.4 Software construction process*	*Software Product Engineering, Activity 4*	*Software Product Engineering, Activity 5*
5.3.8 Software integration	*ENG.1.5 Software integration process*	*Software Product Engineering, Activity 6*	*Software Product Engineering, Activity 6*
5.3.9 Software qualification testing	*ENG.1.6 Software testing process*	*Software Product Engineering, Activity 7*	*Software Product Engineering, Activities 7 and 8*
5.3.10 System integration	*ENG.1.7 System integration and testing process*	*(Software Product Engineering, Activity 6)*	*(Software Product Engineering, Activity 6)*
5.3.11 System qualification testing	*ENG.1.7 System integration and testing process*	*(Software Product Engineering, Activity 7)*	*(Software Product Engineering, Activities 7 and 8)*
5.3.12 Software installation	*CUS.2 Supply process*		*Software Product Engineering, Activity 10*
5.3.13 Software acceptance support	*CUS.2 Supply process*		*Software Product Engineering, Activities 10 and 11*
5.4 Operation process	CUS.4 Operational use process		Software Product Engineering,

ISO 12207	**ISO 15504**	**SW-CMM v1.1**	**SW-CMM v2 Draft C**
			Activity 11
5.5 Maintenance process	ENG.2 System and software maintenance process		(Software Product Engineering, Activity 11)[4]
6. Supporting life cycle processes			
6.1 Documentation process	SUP.1 Documentation process	Software Product Engineering, Activity 8	Software Product Engineering, Activity 9
6.2 Configuration management process	SUP.2 Configuration management process	Software Configuration Management	Software Configuration Management
6.3 Quality assurance process	SUP.3 Quality assurance process	Software Quality Assurance	Software Quality Assurance
6.4 Verification process	SUP.4 Verification process	(Peer Reviews; Software Product Engineering, Activities 5 and 6)	(Peer Reviews; Software Product Engineering, Activities 6 and 7)
6.5 Validation process	SUP.5 Validation process	Software Product Engineering, Activity 5	Software Product Engineering, Activities 7 and 8
6.6 Joint review process	SUP.6 Joint review process	Software Project Tracking & Oversight, Activity 13	(Software Project Control, Activity 10)
6.7 Audit process	SUP.7 Audit process	(Software Quality Assurance)[5]	(Software Quality Assurance)

[4] In general, the SW-CMM considers maintenance to be a particular environment in which all of the KPAs are implemented as appropriate. Maintenance is, however, specifically addressed in the subpractices of PE.AC.11 (as is retirement) to provide a complete picture of the support key practice.

[5] SQA covers both quality assurance and audits. To large degree, audits add the attribute of independence to QA. The SQA KPA can be implemented as an independent function or not; the requirement is objective verification rather than independent verification. SQA may, or may not, therefore cover the Audit Process in a particular environment.

ISO 12207	**ISO 15504**	**SW-CMM v1.1**	**SW-CMM v2 Draft C**
6.8 Problem resolution process	SUP.8 Problem resolution process	Software Configuration Management, Activity 5	Software Configuration Management, Activity 4
7. Organizational life cycle processes			
7.1 Management process	MAN.1 Management process[6]	(Software Project Planning; Software Project Tracking & Oversight; Integrated Software Management)	(Software Project Planning; Software Project Control; Integrated Software Management)
	MAN.2 Project management process	Software Project Tracking & Oversight; Integrated Software Management	Software Project Planning; Software Project Control; Integrated Software Management
	MAN.3 Quality Management Process	Software Quality Management	(Statistical Process Management)[7]
	MAN.4 Risk Management Process	Software Project Planning, Activity 13; Software Project Tracking & Oversight, Activity 10; Integrated Software Management, Activity 10	Software Project Planning, Activity 11; Software Project Tracking & Oversight, Activity 8; Integrated Software Management, Activities 6 and 7

[6] This is the generic planning and management process that is to be applied to any process, rather than specifically to the project.
[7] The process and product issues at level 4 that were separated in v1.1 were combined in SPM in v2.

ISO 12207	ISO 15504	SW-CMM v1.1	SW-CMM v2 Draft C
	ORG.1 Organizational alignment process[8]		(Organization Process Focus; Organization Software Asset Commonality)
7.2 Infrastructure process	ORG.4 Infrastructure process	Organization Process Definition	Organization Process Definition
7.3 Improvement process	ORG.2 Improvement process	Organization Process Definition	Organization Process Definition
7.4 Training process	ORG.3 Human Resource management process	Training Program	Organization Training Program
	ORG.5 Measurement process	Measurement and Analysis (common feature)	Measurement and Analysis (common feature); (Organization Process Performance)
	ORG.6 Reuse process		Organization Software Asset Commonality
		Requirements Management	Requirements Management
		Intergroup Coordination	Project Interface Coordination
		Peer Reviews	Peer Reviews
		Quantitative Process Management	Statistical Process Management
			Organization Process Performance

[8] The purpose of the Organizational Alignment Process is to ensure that individuals share a common vision, culture, and understanding of business goals.

ISO 12207	ISO 15504	SW-CMM v1.1	SW-CMM v2 Draft C
		Defect Prevention	Defect Prevention
		Technology Change Management	Organization Process & Technology Innovation
		Process Change Management	Organization Improvement Deployment

Using the Software CMM® in Small Organizations

Mark C. Paulk

Abstract

The Capability Maturity Model^{SM} for Software developed by the Software Engineering Institute has had a major influence on software process and quality improvement around the world. Although the CMM has been widely adopted, there remain many misunderstandings about how to use it effectively for business-driven software process improvement, particularly for small organizations and small projects. Some of the common problems with interpreting the Software CMM for the small project/organization include:

- What does "small" mean? In terms of people? Time? Size of project? Criticality of product?
- What are the CMM "requirements"? Are there key process areas or goals that should not be applied to small projects/organizations? Are there "invariants" of good processes?
- What are the drivers and motivations that cause abuse of the CMM?

This paper discusses how to use the CMM correctly and effectively in any business environment, with examples for the small organization. The conclusion is that the issues associated with interpreting the Software CMM for the small project or organization may be different in degree, but they are not different in kind, from those for any organization interested in improving its software processes. Using the Software CMM effectively and correctly requires professional judgment and an understanding of how the CMM is structured to be used for different purposes.

MARK C. PAULK
Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh, PA 15213
Telephone: +1 (412) 268-5794
Fax: +1 (412) 268-5758
Internet: mcp@sei.cmu.edu

Mark is a Senior Member of the Technical Staff at the Software Engineering Institute. He has been with the SEI since 1987, initially working with the Software Capability Evaluation project. Mark has worked with the Capability Maturity Model project since its inception and was the project leader during the development of Version 1.1 of the Software CMM. He is also actively involved with software engineering standards, including

- ISO 15504 (aka SPICE -- Software Process Improvement and Capability dEtermination), an emerging suite of international standards for software process assessment
- ISO 12207, Software Life Cycle Processes
- ISO 15288, System Life Cycle Processes

Prior to joining the SEI, Mark was a Senior Systems Analyst for System Development Corporation (later Unisys Defense Systems) at the Ballistic Missile Defense Advanced Research Center in Huntsville, Alabama.

Mark received his master's degree in computer science from Vanderbilt University. He received his bachelor's degree in mathematics and computer science from the University of Alabama in Huntsville.

Professional society memberships and certifications

- Senior Member of the Institute of Electrical and Electronics Engineers (IEEE)
- Senior Member of the American Society for Quality (ASQ)
- ASQ Certified Software Quality Engineer

Using the Software CMM[®] in Small Organizations

Mark C. Paulk

1. Introduction

The Software Engineering Institute (SEI) is a federally funded research and development center established in 1984 by the U.S. Department of Defense with a broad charter to address the transition of software engineering technology – the actual adoption of improved software engineering practices. The SEI's existence is, in a sense, the result of the "software crisis" – software projects that are chronically late, over budget, with less functionality than desired, and of dubious quality. [Gibbs94] To be blunt, much of the crisis is self-inflicted, as when a Chief Information Officer says, "I'd rather have it wrong than have it late. We can always fix it later." The emphasis in many organizations on achieving cost and schedule goals, frequently at the cost of quality, once again teaches a lesson supposedly learned by American industry over twenty years ago and now enshrined in Total Quality Management (TQM).

To quote DeMarco [DeMarco95], this situation is the not-surprising result of a combination of factors:

- "People complain to us because they know we work harder when they complain."
- "The great majority [report] that their software estimates are dismal… but they weren't on the whole dissatisfied with the estimating process."
- "The right schedule is one that is utterly impossible, just not obviously impossible."

DeMarco goes on to observe that our industry is over-goaded, and the only real (perceived) option is to pay for speed by reducing quality.

The lesson of TQM is that focusing on quality leads to decreases in cycle time, increases in productivity, greater customer satisfaction, and business success. The challenge, of course, is defining what "focusing on quality" really means and then systematically addressing the quality issues. Perhaps the SEI's most successful product is the Capability Maturity Model for Software (CMM), a roadmap for software process improvement that has had a major influence on the software community around the world [Paulk95]. The Software CMM defines a five-level framework for how an organization matures its software process. These levels describe an evolutionary path from ad hoc, chaotic processes to mature, disciplined software processes. The five levels, and the 18 key process areas that describe them in detail, are summarized in Figure 1. The five maturity levels prescribe priorities for successful process improvement, whose validity has been documented in many case studies and surveys [Herbsleb97, Lawlis95, Clark97].

Level	Focus	Key Process Areas
5 Optimizing	*Continual process improvement*	Defect Prevention Technology Change Management Process Change Management
4 Managed	*Product and process quality*	Quantitative Process Management Software Quality Management
3 Defined	*Engineering processes and organizational support*	Organization Process Focus Organization Process Definition Training Program Integrated Software Management Software Product Engineering Intergroup Coordination Peer Reviews
2 Repeatable	*Project management processes*	Requirements Management Software Project Planning Software Project Tracking & Oversight Software Subcontract Management Software Quality Assurance Software Configuration Management
1 Initial	*Competent people and heroics*	

Figure 1. An overview of the Software CMM.

Although the focus of the current release of the Software CMM, Version 1.1, is on large organizations and large projects contracting with the government, the CMM is written in a hierarchical form that runs from "universally true" abstractions for software engineering and project management to detailed guidance and examples. The key process areas in the CMM are satisfied by achieving goals, which are described by key practices, subpractices, and examples. The rating components of the CMM are maturity levels, key process areas, and goals. The other components are informative and provide guidance on how to interpret the model. There are 52 goals and 316 key practices for the 18 key process areas. Although the "requirements" for the CMM can be summarized in the 52 sentences that are the goals, the supporting material comprises nearly 500 pages of information. The practices and examples describe what good engineering and management practices are, but they are not prescriptive on how to implement the processes.

The CMM can be a useful tool to guide process improvement because it has historically been a common-sense application of Total Quality Management (TQM) concepts to software that was developed with broad review by the software community. Its five levels are simplistic, but when intelligently used they provide a lever for moving people such as the DOD program manager who bluntly stated, ""The bottom line is schedule. My promotions and raises are based on meeting schedule first and foremost."

While the Software CMM has been very influential around the world in inspiring and guiding software process improvement, it has also been misused and abused by some and not used effectively by others. The guidance provided by CMM v1.1 tends to be oriented towards large projects and large organizations. Small organizations find this problematic, although the fundamental concepts are, we believe, useful to any size organization in any application domain and for any business context.

Are meeting schedules, budgets, and requirements important to small projects? To small organizations? It is arguable that in some environments, such as the commercial shrinkwrap segment, cost is comparatively trivial when compared to the market share available to the first "good enough" product to ship. If the employees of an organization are satisfied with the status quo, there is little that the CMM can provide that will lead to true change; change occurs only when there is sufficient dissatisfaction with the status quo that managers and staff are willing to do things differently. This is as true for small organizations as large.

The CMM provides good advice on desirable management and engineering practices, with an emphasis on management, communication, and coordination of the human-centric, design-intensive processes that characterize software development and maintenance. It should be considered a guidebook rather than a dictate, however, and the CMM user must apply professional judgment based on knowledge and experience in software engineering and management, plus the application domains and business environment of the organization. Because the CMM is focused on software, there are important aspects of TQM that are not directly addressed in the model, such as people issues and the broader perspective of systems engineering, which may also be crucial to the business. The CMM is a tool that should be used in the context of a systematic approach to software process improvement, such as the SEI's IDEAL model, illustrated in Figure 2 [McFeeley96].

An opening question for software process improvement discussions should always be: Why is the organization interested in using the Software CMM? If the desire is to improve process, with a direct tie to business objectives and a willingness to invest in improvement, then the CMM is a useful and powerful tool. If the CMM is simply the flavor of the month, then you have a prescription for disaster. If the driver is customer concerns, ideally the concerns will lead to collaborative improvement between customer and supplier. Sometimes the supplier's concern centers on software capability evaluations (SCEs), such as are performed by government acquisition agencies in source selection and contract monitoring. DOD policies on the criteria for performing SCEs would exclude most small organizations and small projects [Barbour96], but there are circumstances under which they may occur.

Many of the abuses of the Software CMM spring out of a fear of what "others" may do. If an organization applies common sense to the guidance in the CMM as guidance rather than requirements, then many of the interpretation problems of the model vanish. There are cases, however, where ignorance of good engineering and management practices is the problem. This is particularly problematic for good technical people who have been promoted into management positions, but who have little management experience or training. This contributes to the problems identified by a DOD task force [DOD87]:

- "Few fields have so large a gap between best current practice and average current practice."
- "The big problem is not technical... today's major problems with military software development are not technical problems, but management problems."

2. Small Organizations and Small Projects

The focus of this paper is on using the Software CMM correctly and effectively for small organizations because I am frequently asked, "Can the Software CMM be used for small projects (or small organizations)?" Yet the definition of "small" is challengingly ambiguous, as illustrated in Table 1. At one time there was an effort to develop a tailored CMM for small projects and organizations, but the conclusion of a 1995 CMM tailoring workshop was that we could not even agree on what "small" really meant! The result was a report on how to tailor the CMM rather than a tailored CMM for small organizations [Ginsberg95]. In a 1998 SEPG conference panel on the CMM and small projects [Hadden98a], small was defined as "3-4 months in duration with 5 or fewer staff." Brodman and Johnson define a small organization as fewer than 50 software developers and a small project as fewer than 20 developers [Johnson98].

Table 1. Defining a "Small" Project

Variant of "Small"	Number of People	Amount of Time
Small	3-5	6 months
Very small	2-3	4 months
Tiny	1-2	2 months
Individual	1	1 week
Ridiculous!	1	1 hour

Note that small to tiny projects are in the range being addressed by Humphrey in his Team Software Process^{SM} (TSP) work, and the individual effort is in the range of the Personal Software Process^{SM} (PSP) [Humphrey95]. TSP and PSP illustrate how CMM concepts are being applied to small projects. The "ridiculous" variant represents an interpretational problem. On the two occasions this variant has been discussed, the problem was the definition of "project." In both cases it was a maintenance environment, and the organization’s “projects” would have been described as tasks in the CMM; the more accurate interpretation for a CMM “project” was a baseline upgrade or maintenance release… but the terminology clash was confusing.

One of the first challenges for small organizations in using the CMM is that their primary business objective is to survive! Even after deciding the status quo is unsatisfactory and process improvement will help, finding the resources and assigning responsibility for process improvement, and then following through by defining and deploying processes is a difficult business decision. The small organization tends to believe

- we are all competent – people were hired to do the job, and we can’t afford training in terms of either time or money
- we all communicate with one another – “osmosis” works because we’re so “close”
- we are all heroes – we do whatever needs to be done, the rules don’t apply to us (they just get in the way of getting the job done), we live with short cycle times and high stress

Yet small organizations, just like large ones, will have problems with undocumented requirements, the mistakes of inexperienced managers, resource allocation, training, peer reviews, and documenting the product. Despite these challenges, small organizations can be extraordinarily innovative and productive. Although there are massive problems that may require large numbers of people to solve, in general small teams are more productive than large teams – they jell quicker and there are far fewer communication problems. The question remains, however, is process discipline needed for small teams? To answer this CMM mantra, we need to consider what discipline involves – and that leads to the heart of this paper’s CMM interpretation discussion.

One last precursor, however. When assessing “small” organizations, it is advisable to use a streamlined assessment process; the formality of a two-week CMM-based appraisal for internal process improvement (CBA IPI) is probably excessive [Strigel95, Paquin98, Williams98]. The emphasis should be on efficiently identifying important problems, even if some are missed due to lack of rigor. I recommend focusing on the institutionalization practices that establish the organization’s culture: planning, training, etc.; and explicitly tying process improvement to business needs.

3. Interpreting the CMM

Where does the Software CMM apply? The CMM was written to provide good software engineering and management practices for <u>any</u> project in <u>any</u> environment. The model is described in a hierarchy

Maturity levels	(5)
→ **Key process areas**	(18)
→ **Goals**	(52)
→ *Key practices*	(316)
® *Subpractices and examples*	(many)

In my experience over the last decade of software process work, environments where interpretation and tailoring of the CMM are needed include:

- very large programs
- virtual projects or organizations
- geographically distributed projects
- rapid prototyping projects
- research and development organizations
- software services organizations
- small projects and organizations

The interpretation guidance for small projects and small organizations is also applicable to large projects and organizations. Intelligence and common sense are required to use the CMM correctly and effectively [Paulk96]. It is simultaneously true that all (software) projects are different and all (software) projects are the same. We are required to balance conflicting realities: similarity versus uniqueness, order versus chaos. Those who succeed build lasting organizations [Collins94] that are truly capable of organizational learning [Senge90]; the rest must derive their success elsewhere.

The "normative" components of the CMM are maturity levels, key process areas, and goals. All practices in the CMM are informative. Since the detailed practices primarily support large, contracting software organizations, they are not necessarily appropriate, as written, for direct use by small projects and small organizations – but they do provide insight into how to achieve the goals and implement repeatable, defined, measured, and continually improving software processes. Thus we prevent such "processes" as the estimating procedure that was simply "Go ask George."

My most frequent interpretation recommendation is to develop a mapping between CMM terminology and the language used by the organization. In particular, terms dealing with organizational structures, roles and relationships, and formality of processes need to be mapped into their organizational equivalents to prevent misunderstandings such as the "ridiculous one-hour project." Examples of organizational structures include "independent groups" such as quality assurance, testing, and configuration management. Appropriate organizational terminology for roles such as project manager and project software manager should be specified. People may fill multiple roles; for example, one person may be the project manager, project software manager, SCM manager, etc. Explicitly stating this makes interpretation of the CMM much simpler and more consistent.

Once the terminology issues are understood, we can think about what the "invariants" for a disciplined process are and which practices depend on the context. In general we assume that key process areas and goals are always relevant to any environment, with the exception of *Software Subcontract Management*, which may be "not applicable" if there is no subcontracting. In contrast, I can conceive of no circumstances under which *Peer Reviews* can be reasonably tailored out for a Level 3 organization. This is a matter of competent professional judgment, although an alternative practice such as formal methods might replace peer reviews. Professional judgment and trained, experienced assessors are crucial, even for small organizations! [Abbott97]

I have never seen an environment where the following were not needed (though implementations differ):

- documented customer (system) requirements
- communication with customer (and end users)
- agreed-to commitments
- planning
- documented processes
- work breakdown structure

Some practices, however, deal with "large-project implementations." A small project is unlikely to need an SCM group or a Change Control Board… but configuration management and change control are always necessary. An independent SQA group may not be desirable, but objective verification that requirements are satisfied always is. An independent testing group may not be established, but testing is always necessary. We thus see that even for context-sensitive practices, the intent is critical even if the implementation is radically different between small organizations and large. Many of the context-sensitive, large-project implementation issues relate to organizational structure. If one reads the CMM definition of "group," it states that "a group could vary from a single individual assigned part time, to several part-time individuals assigned from different departments, to several individuals dedicated full time," which is intended to cater to a variety of contexts.

In addition to these, specific questions that arise repeatedly, especially for small organizations, relate to:

- management sponsorship
- measurement
- SEPGs
- "as is" processes
- documented processes

- tailoring
- training
- risk management
- planning
- peer reviews

Trite though it may seem, obtaining senior management sponsorship is a crucial component of building organizational capability. As individuals, we can exercise professionalism and discipline within our sphere of control, but if an organization as a whole is to change its performance, then its senior management must actively support the change. Bottom-up improvement, without sponsorship and coordination, leads to islands of excellence rather than predictably improved organizational capability. It should be noted, however, that for small organizations, while the president (or founder) is the primary role model, a respected "champion" frequently has the influence to move the entire organization – including the president.

Management by fact is a paradigm shift for most organizations, which must be based on a measurement foundation. To make data analysis useful, you need to understand what the data means and how to analyze it meaningfully. Begin by collecting a simple set of <u>useful</u> data. You also have to be sensitive to the potential for causing dysfunctional behavior by what you measure [Austin96]. The act of measuring identifies what is important, but some things are difficult to measure. Management needs to ensure that attention is visibly paid to all critical aspects of the project, including those difficult to measure, not just those it is easy to measure and track.

In most organizations, a software engineering process group (SEPG) or some equivalent should be formed to coordinate process definition, improvement, and deployment activities. One of the reasons for dedicating resources to an SEPG is to ensure follow-through on appraisal findings. Many improvement programs have foundered simply because no action resulted from the appraisal. Small organizations may not have full-time SEPG staff, but the responsibility for improvement should be explicitly assigned and monitored.

Begin with the "as is" process, not the "should be" process, to leverage effective practices and co-opt resisters. Mandating top-down that everyone will follow the new "should be" process, particularly if not developed by empowered workers, is a common recipe for failure. The "as is" process evolved because the people doing the work needed to get the job done – even if that meant going around the system. The "should be" process may, or may not, be feasible in the given culture and environment. With an organizational focus on process management and improvement, the "as is" and "should be" processes will converge, resulting in organizational learning.

Document your processes. The reasons for documenting a process (or product) are 1) to communicate – to others now and perhaps to yourself later; 2) to understand – if you can't write it down, you don't really understand; and 3) to encourage consistency – take advantage of repeatability. Documented processes support organizational learning and prevent reinventing the wheel for common problems – they put repeatable processes in place. Documentation is therefore important, but documents need not be lengthy or complex to be useful. Keep the process simple because we live in a rapidly changing world. Processes do not need to be lengthy or complex. The CMM is about <u>doing</u> things, not <u>having</u> things. A 1-2 page process description may suffice, and subprocesses and procedures can be invoked as needed and useful. Use good software design principles, such as locality, information hiding, and abstraction, in defining processes. Another useful rule of thumb is to track work at 2-3 tasks per week <u>at most</u>. Order is not created by complex controls, but by the presence of a few guiding formulae or principles [Wheatley92, page 11].

Processes need to be tailored to the needs of the project [Ginsberg95, Ade96]. Although standard processes provide a foundation, each project will also have unique needs. Unreasonable constraints on tailoring can lead to significant resistance to following the process. As Hoffman expresses it, "Don't require processes that don't make sense." [Hoffman98]

The degree of formality needed for processes is a frequent challenge for both large and small organizations [Comer98]. Should there be separate procedure for each of the 25 key practices at Level 2 that mention "according to a documented procedure?" [Hadden98a, Pitterman98] The answer, as discussed in section 4.5.5 "Documentation and the CMM" of the CMM book [Paulk95], is a resounding NO! Packaging of documentation is an organizational decision.

Documented processes are of little value if they are not effectively deployed. To achieve buy-in for the documented, process implementers must be part of process definition and improvement. Training, via a wide variety of mechanisms, is critical to consistent and effective software engineering and management. The reason for training is to develop skills. There are many "training mechanisms" other than formal classroom training that can be effective in building skills. One that should be seriously considered is a formal mentoring program. In this case, formality means going beyond assigning a mentor and hoping that experience will rub off. Formality implies training people on how to mentor and monitoring the effectiveness of the mentoring.

Training remains an issue after the initial deployment of a process or technology [Abbott97, Williams98]. As personnel change, the incremental need for training may not be adequately addressed. Mentoring and apprentice programs may suffice to address this issue, but they cannot be assumed to be satisfactory without careful monitoring.

Management training is particularly important because ineffective management can cripple a good team. People who are promoted to management because of their technical skills have to acquire a new set of skills, including interpersonal skills [Mogilensky94, Curtis95, Weinberg94].

Some argue that software project management is really risk management. In one sense, the CMM is about managing risk. We attempt to establish stable requirements so that we can plan and manage effectively, but the business environment changes rapidly, perhaps chaotically. We try to establish an island of order in the sea of software chaos, but both order and chaos have a place. As Wheatley suggests, "To stay viable, open systems maintain a state of non-equilibrium, keeping the system in balance so that it can change and grow." [Wheatley92, page 78] Although we can establish processes that help us manage the risks of a chaotic world, we also need to change and grow.

This implies that you should use an incremental or evolutionary life cycle. If you want to focus on risk management, the spiral model may be the preferred life cycle model. If you want to focus on involving the customer, perhaps rapid prototyping or joint application design would be preferable. Few long-term projects have the luxury of the stable environment necessary for the waterfall life cycle to be the preferred choice – yet it is probably the most common life cycle. Note, however, that for small projects, the waterfall life cycle may be an excellent choice.

The #1 factor in successful process definition and improvement is "planfulness" [Curtis96]. Planning is needed for every major software process, but within the bounds of reasonable judgment, the organization determines what is "major" and how the plan should be packaged. A plan may reside in several different artifacts or be embedded in a larger plan.

Although you can argue over the best kind of peer review, the simple fact is that the benefits of peer reviews far outweigh their costs. The data suggests some form of inspection should be used [Ackerman89], but any form of collegial or disciplined review, such as structured walkthroughs, adds significant value. Recognizing the value of peer reviews does not mean, unfortunately, that we do them systematically. We need to "walk the walk," not just "talk the talk." This is very frustrating for technical people who do not understand the emphasis on management in the CMM, yet poor management leads to abandoning good engineering practices such as peer reviews.

There are other issues that have been identified for small organizations and projects. Paquin [Paquin98] identifies five:

- assessments
- project focus

- documentation
- required functions
- maturity questionnaire

We have not discussed the project focus of Level 2 as being a challenge for small organizations. Software process improvement involves overhead that may be excessive for a small project. Some recommend attacking small project process improvement from an organizational perspective [Comer98, Paquin98], which is certainly a reasonable approach, even it does seem to mix Levels 2 and 3. This is a consideration for any size organization or project [Paulk96]. Although an organization can achieve Level 2 without an organization process focus, the most effective organizational learning strategy will be one that stresses organizational assets that lessen the overhead of projects. At the same time, it must be recognized that there may be resistance to change at the project level, perhaps based on valid concerns, and addressing resistance needs to be considered part of the organization's learning process.

Required functions are an issue because there may be more CMM functions than there are people. This issue has been discussed as terminology or role mapping. The maturity questionnaire is a concern because it uses CMM terminology, thus it may be confusing to those filling it out. Expressing the questionnaire in the terminology of the organization is thus a desirable precursor to even an informal assessment or survey.

Abbott [Abbott97] identifies six keys to software process improvement in small organizations:

- senior management support
- adequate staffing
- applying project management principles to process improvement
- integration with ISO 9001
- assistance from process improvement consultants
- focus on providing value to projects and to the business

If applying good project management to software projects is the best way to ensure success, then the same should be true for process improvement, which should be treated like any other project. ISO 9001 is more frequently an issue for large organizations than small, so it is interesting that Abbott points this out for his small company.

Brodman and Johnson [Johnson98] identify seven small organization/small project challenges:

- handling requirements
- generating documentation
- managing projects
- allocating resources
- measuring progress
- conducing reviews
- providing training

Brodman and Johnson have developed a tailored version of the CMM for small businesses, organizations, and projects [Johnson96, Johnson97, Brodman94]. Although the majority of the key practices in the CMM were tailored in the LOGOS Tailored CMM, the changes can be characterized as:

- clarification of existing practices
- exaggeration of the obvious
- introduction of alternative practices (particularly as examples)
- alignment of practices with small business/small organization/small project structure and resources

Therefore the changes involved in tailoring the CMM for small organizations should not be considered radical.

4. Abusing the Software CMM

Using the CMM correctly means balancing conflicting objectives. CMM-based appraisals require the use of professional judgment. Although the CMM provides a significant amount of guidance in making these judgments, removing subjectivity implies a deterministic, repetitive process that is not characteristic of engineering design work. The CMM is sometimes referred to as a set of process requirements, but it does not contain any “shall” statements. That is why it is an abuse of the CMM to check off (sub)practices for conformance.

Some are unwilling or unable to interpret, tailor, or apply judgment. It is easy to mandate the key practices, but foolhardy. This foolishness is frequently driven by paranoia about customer intentions and competence. On more than one occasion I have heard someone say they were doing something that was foolish, but they ware afraid that the customer was so ignorant or incompetent that they would be unable to understand the rationale for doing things differently than literally described in the CMM. This is particularly problematic for SCEs. It is true that judgments may differ – and sometimes legitimately so. What is adequate in one environment may not suffice for a new project. That is why we recommend that process maturity be included in risk assessment rather than using maturity levels to filter offerors [Barbour96]. Small organizations should have less of a concern with this problem since it is unlikely that SCEs for small organizations are cost-effective. It is more of a problem for large organizations with many small projects.

Unfortunately I have no solution for this problem. “Standards” such as the CMM can help organizations improve their software process, but focusing on achieving a maturity level without addressing the underlying process can cause dysfunctional behavior. Maturity levels should be measures of improvement, not goals of improvement. That is why we emphasize the need to tie improvement to business objectives.

5. Conclusion

The bottom line is that software process improvement should be done to help the business – not for its own sake. This is true for both large organizations and small. The best advice comes from Sanjiv Ahuja, President of Bellcore: “Let common sense prevail!”

Building software is a design-intensive, creative activity. While the discipline of process is a crucial enabler of success, the objective is to solve a problem, and this requires creativity. Software processes should be repeatable, even if they are not repetitive. The balance between discipline and creativity can be challenging [Glass95]. Losing sight of the creative, design-intense nature of software work leads to stifling rigidity. Losing sight of the need for discipline leads to chaos.

The CMM represents a “common sense engineering” approach to software process improvement. Its maturity levels, key process areas, goals, and key practices have been extensively discussed and reviewed within the software community. While the CMM is neither perfect nor comprehensive, it does represent a broad consensus of the software community and is a useful tool for guiding improvement efforts, and it can be uses to help small software organizations improve their processes [Abbott97, Hadden98b, Hoffman98, Pitterman98, Sanders98].

Small organizations should seriously consider PSP and TSP [Ferguson97, Hayes97]. Having taken the PSP course, I can highly recommend it for building self-discipline. Note that the effect of reading the book is not the same as taking the course and doing the work! Where the CMM addresses the organizational side of process improvement, PSP addresses building the capability of individual practitioners. The PSP course convinces the individual, based on his or her own data, of the value of a disciplined, engineering approach to building software.

References

Abbott97 John J. Abbott, “Software Process Improvement in a Small Commercial Software Company,” , **Proceedings of the 1997 Software Engineering Process Group Conference**, San Jose, CA, 17-20 March 1997.

Ackerman89 A.F. Ackerman, L.S. Buchwald, and F.H. Lewski, “Software Inspections: An Effective Verification Process,” IEEE Software, Vol. 6, No. 3, May 1989, pp. 31-36.

Ade96 Randy W. Ade and Joyce P. Bailey, "CMM Lite: SEPG Tailoring Guidance for Applying the Capability Maturity Model for Software to Small Projects," **Proceedings of the 1996 Software Engineering Process Group Conference: Wednesday Papers**, Atlantic City, NJ, 20-23 May 1996.

Austin96 Robert D. Austin, **Measuring and Managing Performance in Organizations**, Dorset House Publishing, ISBN: 0-932633-36-6, New York, NY, 1996.

Barbour96 Rick Barbour, “Software Capability Evaluation Version 3.0 Implementation Guide for Supplier Selection,” Software Engineering Institute, Carnegie Mellon University, CMU/SEI-95-TR-012, April 1996.

Brodman94 J.G. Brodman and D.L. Johnson, "What Small Businesses and Small Organizations Say About the CMM," **Proceedings of the 16th International Conference on Software Engineering**, IEEE Computer Society Press, Sorrento, Italy, 16-21 May 1994, pp. 331-340.

Clark97 Bradford K. Clark, "The Effects of Software Process Maturity on Software Development Effort," PhD Dissertation, Computer Science Department, University of Southern California, August 1997.

Collins94 James C. Collins and Jerry I. Porras, **Built to Last**, HarperCollins Publishers, New York, NY, 1994.

Curtis95 Bill Curtis, William E. Hefley, and Sally Miller, “People Capability Maturity Model,” Software Engineering Institute, CMU/SEI-95-MM-02, September 1995.

Curtis96 Bill Curtis, "The Factor Structure of the CMM and Other Latent Issues," **Proceedings of the 1996 Software Engineering Process Group Conference: Tuesday Presentations**, Atlantic City, NJ, 20-23 May 1996.

DeMarco95 Tom DeMarco, **Why Does Software Cost So Much?**, ISBN 0-932633-34-X, Dorset House, New York, NY, 1995.

DOD87 Department of Defense, "Report of the Defense Science Board Task Force on Military Software," Office of the Under Secretary of Defense for Acquisition, Washington, D.C., September 1987.

Ferguson97 Pat Ferguson and Jeanie Kitson, “CMM-Based Process Improvement Supplemented by the Personal Software Process in a Small Company Environment,” , **Proceedings of the 1997 Software Engineering Process Group Conference**, San Jose, CA, 17-20 March 1997.

Gibbs94 W. Wayt Gibbs, "Software's Chronic Chrisis," Scientific American, September 1994, pp. 86-95.

Ginsberg95 Mark Ginsberg and Lauren Quinn, “Process Tailoring and the Software Capability Maturity Model,” Software Engineering Institute, CMU/SEI-94-TR-024, November 1995.

Glass95 Robert L. Glass, **Software Creativity**, Prentice Hall, Englewood Cliffs, NJ, 1995.

Hadden98a Rita Hadden, “How Scalable are CMM Key Practices?” Crosstalk: The Journal of Defense Software Engineering, Vol. 11, No. 4, April 1998, pp. 18-20, 23.

Hadden98b	Rita Hadden, "Key Practices to the CMM: Inappropriate for Small Projects?" panel, Rita Hadden moderator, **Proceedings of the 1998 Software Engineering Process Group Conference**, Chicago, IL, 9-12 March 1998.
Hayes97	Will Hayes and James W. Over, "The Personal Software Process (PSP): An Empirical Study of the Impact of PSP on Individual Engineers," Software Engineering Institute, Carnegie Mellon University, CMU/SEI-97-TR-001, December 1997.
Herbsleb97	James Herbsleb, David Zubrow, Dennis Goldenson, Will Hayes, and Mark Paulk, "Software Quality and the Capability Maturity Model," Communications of the ACM, Vol. 40, No. 6, June 1997, pp. 30-40.
Hoffman98	Leo Hoffman, "Small Projects and the CMM," in "Key Practices to the CMM: Inappropriate for Small Projects?" panel, Rita Hadden moderator, **Proceedings of the 1998 Software Engineering Process Group Conference**, Chicago, IL, 9-12 March 1998.
Humphrey95	Watts S. Humphrey, **A Discipline for Software Engineering**, ISBN 0-201-54610-8, Addison-Wesley Publishing Company, Reading, MA, 1995.
Johnson96	Donna L. Johnson and Judith G. Brodman, **The LOGOS Tailored Version of the CMM for Small Businesses, Small Organizations, and Small Projects**, Version 1.0, August 1996.
Johnson97	Donna L. Johnson and Judith G. Brodman, "Tailoring the CMM for Small Businesses, Small Organizations, and Small Projects," Software Process Newsletter, IEEE Computer Society Technical Council on Software Engineering, No. 8, Winter 1997, p. 1-6.
Johnson98	Donna L. Johnson and Judith G. Brodman, "Applying the CMM to Small Organizations and Small Projects," **Proceedings of the 1998 Software Engineering Process Group Conference**, Chicago, IL, 9-12 March 1998.
Lawlis95	Patricia K. Lawlis, Robert M. Flowe, and James B. Thordahl, "A Correlational Study of the CMM and Software Development Performance," Crosstalk: The Journal of Defense Software Engineering, Vol. 8, No. 9, September 1995, pp. 21-25. Reprinted in Software Process Newsletter, IEEE Computer Society Technical Council on Software Engineering, No. 7, Fall 1996, pp. 1-5.
McFeeley96	Bob McFeeley, "IDEAL: A User's Guide for Software Process Improvement," Software Engineering Institute, CMU/SEI-96-HB-001, February 1996.
Mogilensky94	Judah Mogilensky and Betty L. Deimel, "Where Do People Fit in the CMM?," American Programmer, Vol. 7, No. 9, September 1994, pp. 36-43.
Paquin98	Sherry Paquin, "Struggling with the CMM: Real Life and Small Projects," in "Key Practices to the CMM: Inappropriate for Small Projects?" panel, Rita Hadden moderator, **Proceedings of the 1998 Software Engineering Process Group Conference**, Chicago, IL, 9-12 March 1998.
Paulk95	Carnegie Mellon University, Software Engineering Institute (Principal Contributors and Editors: Mark C. Paulk, Charles V. Weber, Bill Curtis, and Mary Beth Chrissis), **The Capability Maturity Model: Guidelines for Improving the Software Process**, ISBN 0-201-54664-7, Addison-Wesley Publishing Company, Reading, MA, 1995.
Paulk96	Mark C. Paulk, "Effective CMM-Based Process Improvement," **Proceedings of the 6th International Conference on Software Quality**, Ottawa, Canada, 28-31 October 1996, pp. 226-237.

Pitterman98	Bill Pitterman, "Key Practices to the CMM: Inappropriate for Small Projects?" panel, Rita Hadden moderator, **Proceedings of the 1998 Software Engineering Process Group Conference**, Chicago, IL, 9-12 March 1998.
Sanders98	Marty Sanders, "Small Company Action Training and Enabling," in **The CMM and Small Projects**, Society for Software Quality Roundtable, Washington, DC, 26 January 1998.
Senge90	Peter M. Senge, **The Fifth Discipline: The Art & Practice of the Learning Organization**, Doubleday/Currency, New York, NY, 1990.
Strigel95	Wolfgang B. Strigel, "Assessment in Small Software Companies," **Proceedings of the 1995 Pacific Northwest Software Quality Conference**, 1995, pp. 45-56.
Weinberg94	Gerald M. Weinberg, **Quality Software Management, Volume 3: Congruent Action**, ISBN 0-932633-28-5, Dorset House, New York, NY, 1994.
Wheatley92	Margaret J. Wheatley, **Leadership and the New Science**, Berrett-Koehler Publishers, San Francisco, CA, 1992.
Williams98	Louise B. Williams, "SPI Best Practices for 'Small' Projects," in **The CMM and Small Projects**, Society for Software Quality Roundtable, Washington, DC, 26 January 1998.

Using the Software CMM® With Good Judgment

MARK C. PAULK
Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh, PA 15213
Telephone: +1 (412) 268-5794
Fax: +1 (412) 268-5758
Internet: mcp@sei.cmu.edu –or– Mark.Paulk@ieee.org

Abstract

The Capability Maturity Model® for Software (CMM) developed by the Software Engineering Institute (SEI) has had a major influence on software process and quality improvement around the world. Although the CMM has been widely adopted, there remain many misunderstandings about how to use it effectively for business-driven software process improvement, particularly for small organizations and small projects. This paper discusses how to use the CMM correctly and effectively in any business environment, with examples for small organizations, rapid prototyping projects, maintenance shops, R&D outfits, and other environments. The conclusion is that the issues associated with interpreting the Software CMM are essentially the same for any organization interested in improving its software processes – the differences are of degree rather than kind. Using the Software CMM effectively and correctly requires professional judgment and an understanding of how the CMM is structured to be used for different purposes.

Key Words: Capability Maturity Model, CMM, software capability evaluation, software process assessment, small organizations, small projects, software process improvement.

Published in ASQ Software Quality Professional, Vol. 1, No. 3, June 1999, pp. 19-29.

1. Introduction

The Software Engineering Institute (SEI) is a federally funded research and development center established in 1984 by the U.S. Department of Defense (DOD) with a broad charter to improve the state of the practice in software engineering. The SEI's existence is, in a sense, the result of the "software crisis" – software projects that are chronically late, over budget, with less functionality than desired, and of dubious quality.

Much of the software crisis is self-inflicted, as when a Chief Information Officer says, "I'd rather have it wrong than have it late. We can always fix it later." The emphasis in all too many organizations is on achieving cost and schedule goals, frequently at the cost of quality. To quote DeMarco [DeMarco95], this situation is the not-surprising result of a combination of factors:

- "People complain to us [software developers] because they know we work harder when they complain."
- "The great majority [report] that their software estimates are dismal… but they weren't on the whole dissatisfied with the estimating process."
- "The right schedule is one that is utterly impossible, just not obviously impossible."

DeMarco goes on to observe that our industry is over-goaded, and the only real (perceived) option is to pay for speed by reducing quality. This violates the principle of Total Quality Management (TQM) that focusing on quality leads to decreases in cycle time, increases in productivity, greater customer satisfaction, and business success. Admittedly the challenge of determining "good enough" is an on-going debate within the software community and a delicate business decision, but the quality focus is central to the SEI's work.

Perhaps the SEI's most successful product is the Capability Maturity Model® for Software (CMM), a roadmap for software process improvement that applies TQM ideas to software and has had a major influence on the software community around the world [Paulk95]. The Software CMM defines a five-level framework for how an organization matures its software process capability. These levels describe an evolutionary path from ad hoc, chaotic processes to mature, disciplined software processes. The five levels, and the 18 key process areas that describe them in detail, are summarized in Figure 1.

The purpose of this paper is to discuss how the CMM can be effectively used in a wide range of environments, with a focus on small organizations but also including examples for rapid prototyping projects, maintenance shops, and R&D outfits. This paper summarizes the observations and recommendations of the author, based on over a decade of experience in CMM-based assessments and process improvement. The recommendations may appear dogmatic, but it seems unlikely that many will disagree with their intent, although there is likely to be some debate over what constitutes an "adequate" implementation in any given environment.

Level	Focus	Key Process Areas
5 Optimizing	*Continual process improvement*	Defect Prevention Technology Change Management Process Change Management
4 Managed	*Product and process quality*	Quantitative Process Management Software Quality Management
3 Defined	*Engineering processes and organizational support*	Organization Process Focus Organization Process Definition Training Program Integrated Software Management Software Product Engineering Intergroup Coordination Peer Reviews
2 Repeatable	*Project management processes*	Requirements Management Software Project Planning Software Project Tracking & Oversight Software Subcontract Management Software Quality Assurance Software Configuration Management
1 Initial	*Competent people and heroics*	

Figure 1. An overview of the Software CMM.

Although the focus of the current release of the Software CMM, Version 1.1, is on large organizations and large projects contracting with the government, the CMM is written in a hierarchical form that runs from "universally true" abstractions for software engineering and project management to detailed guidance and examples. The key process areas in the CMM are satisfied by achieving goals, which are described by key practices, subpractices, and examples. The rating components of the CMM are maturity levels, key process areas, and goals. The other components are informative and provide guidance on how to interpret the model. Although the "requirements" for the CMM can be summarized in the 52 sentences that are the goals, the supporting material comprises nearly 500 pages of information. The practices and examples describe what good engineering and management practices are, but they are not prescriptive on how to implement the processes.

Although key practices are not requirements, they are intended to be generally applicable. The goals of each key process area address end-states, and each key practice contributes to achieving one or more goals. Although they set expectations, the key practices are not required; there may be alternative methods for achieving a goal. Assessment teams usually find a few key practices, typically 3-5, where “alternate implementations” are used that satisfy the goals of a key process area. This is a large enough percentage of the 316 key practices in the CMM that the need for judgment is clear, but small enough to indicate that the key practices are generally good guidance. About 10-15% of the key practices usually have to be "interpreted” – the team has to discuss at length whether an implementation is adequate as opposed to arriving at a quick consensus. Key practices are not requirements and there may be alternate implementations, but this does not abrogate the responsibility to make informed, reasonable, and professional judgments about each key practice and its associated goals – and assessment findings may be written against key practices and subpractices when an implementation is judged inadequate.

Although the CMM is a common-sense application of TQM concepts to software that was developed with broad review by the software community, small organizations may find the large organization/project orientation of the CMM problematic. Its fundamental concepts are, we believe, useful to any size organization in any application domain and for any business context. Are meeting schedules, budgets, and requirements important to small projects? To small organizations? If the employees of an organization are satisfied with the status quo, there is little that the CMM can provide that will lead to true change. Change occurs only when there is sufficient dissatisfaction with the status quo that managers and staff are willing to do things differently. This is as true for small organizations as large.

Intelligence and common sense are needed to use the CMM correctly and effectively [Paulk96], and this is true in all environments. Based on over a decade’s experience in software process work, environments where interpretation and tailoring of the CMM are needed include

- very large programs, with many organizations interacting
- virtual projects or organizations
- geographically distributed projects
- rapid prototyping projects
- research and development (R&D) organizations
- maintenance (sustaining engineering) shops
- software services organizations
- small projects and organizations

2. Small Organizations and Small Projects

The emphasis of this paper on small organizations is due to a frequently asked question, “Can the Software CMM be used for small projects (or small organizations)?” Yet the definition of “small” is challengingly ambiguous, as illustrated in Table 1. At one time there was an effort to develop a tailored CMM for small projects and organizations, but the conclusion of a 1995 CMM tailoring workshop was that we could not even agree on what “small” really meant. The result was a report on how to tailor the CMM rather than a tailored CMM for small organizations [Ginsberg95]. In a 1998 SEPG conference panel on the CMM and small projects [Hadden98b], small was defined as “3-4 months in duration with 5 or fewer staff.” Brodman and Johnson define a small organization as fewer than 50 software developers and a small project as fewer than 20 developers [Brodman96, Johnson97].

Table 1. Defining a "Small" Project

Variant of "Small"	Number of People	Duration
Small	3-5	6 months
Very small	2-3	4 months
Tiny	1-2	2 months
Individual	1	1 week
Ridiculous!	1	1 hour

Small to tiny projects are in the range being addressed by Humphrey in his Team Software Process^{SM} (TSP) work, and the individual effort is directly addressed by the Personal Software Process^{SM} (PSP) [Humphrey95]. TSP and PSP illustrate how CMM concepts are being applied to small projects. The "ridiculous" variant represents an interpretational problem. On the two occasions this variant has been discussed, the problem was the definition of "project." In both cases it was a maintenance environment, and the organization's "projects" would have been described as tasks in the CMM; the more accurate interpretation for a CMM "project" was a baseline upgrade or maintenance release, but there was a confusing terminology clash.

Appropriately interpreted, the CMM is being effectively used in organizations with less than 15 employees and for projects with as few as two people. The SEI's maturity profile as of December 1998 shows 27 organizations with fewer than 25 employees that have performed CBA IPI assessments.

Small organizations, just like large ones, will have problems with undocumented requirements, the mistakes of inexperienced managers, resource allocation, training, peer reviews, and documenting the product. The challenge of providing resources for process improvement – both to formally identify problems and systematically address them – will frequently result in these resources being part-time rather than full-time. The business question is whether the kind of process discipline advocated by the CMM is really needed by small projects and organizations. To answer this question, we need to consider what discipline involves – and that leads to the heart of this paper's CMM interpretation discussion.

3. Improvement Issues from the IDEAL Perspective

The CMM is a tool that should be used in the context of a systematic approach to software process improvement, such as the SEI's IDEAL model [McFeeley96], which depicts the activities of an improvement program in five phases:

- **I** Initiating (the improvement program)
- **D** Diagnosing (the current state of practice)
- **E** Establishing (the plans for the improvement program)
- **A** Acting (on the plans and recommended improvements)
- **L** Learning (the lessons learned and the business results of the improvement effort)

Obtaining senior management sponsorship in the *Initiating* phase is a crucial component of building organizational capability. As individuals, we can exercise professionalism and discipline within our sphere of control, but if an organization as a whole is to change its performance, then its senior management must actively support the change. Bottom-up improvement, without sponsorship and coordination, leads to islands of excellence rather than predictably improved organizational capability. For small organizations, while the president (or founder) is the primary role model, a respected "champion" frequently has the influence to move the entire organization – including the president.

An opening question should always be

Why is the organization interested in using the Software CMM?

If the desire is to improve process, with a direct tie to business objectives and a willingness to invest in improvement, then the CMM is a useful and powerful tool. If the CMM is simply the flavor of the month, then you have a prescription for disaster. If the driver is customer concerns, ideally the concerns will lead to collaborative improvement between customer and supplier. Sometimes the supplier's concerns center on software capability evaluations (SCEs), such as are performed by government acquisition agencies in source selection and contract monitoring. DOD policies on the criteria for performing SCEs would exclude most small organizations and small projects [Barbour96]. There are circumstances, however, under which SCEs may still occur, such as field training an evaluation team.

The relationship to any existing TQM programs should be identified in the Initiating phase. Software process improvement efforts should be aligned with quality improvement initiatives since the goals are the same, even if the scopes initially differ. TQM programs are relatively uncommon in small organizations, although following the principles of TQM is always recommended.

When assessing "small" organizations in the *Diagnosing* phase, it is advisable to use a streamlined assessment process. A full-blown CMM-based appraisal for internal process improvement (CBA IPI) [Dunaway96], with its criteria for validity, accuracy, corroboration, consistency, and sufficiency, can last two weeks, which is probably excessive for a small organization [Strigel95, Paquin98, Williams98]. The emphasis should be on efficiently identifying important problems, even if some are missed due to lack of rigor. The CBA IPI assessment method can be tailored down, however, and there are a number of other assessment methods, such as Interim Profile, that may be more appropriate [Whitney94, Daskalantonakis94].

Perhaps the best recommendation regarding CMM interpretation is to develop a mapping between CMM terminology and the language used by the organization. In particular, terms dealing with organizational structures, roles and relationships, and formality of processes need to be mapped into their organizational equivalents. Examples of organizational structures include "independent groups" such as quality assurance, testing, and configuration management. Appropriate organizational terminology for roles such as project manager and project software manager should be specified. People may fill multiple roles; for example, one person may be the project manager, project software manager, SCM manager, etc. Explicitly stating this makes interpretation of the CMM much simpler and more consistent.

Small organizations have an opportunity in the *Establishing* phase that large organizations would find difficult – combining Levels 2 and 3. Level 2 focuses on projects, yet a small organization should find it comparatively easy to develop organizational standards at the same time that it is defining its project-level processes since there is much less "cultural inertia" to overcome, even when significant cultural change occurs. The most effective organizational learning strategy is likely to be one stressing organizational assets that lessen the overhead of projects. In large organizations resistance to change makes this strategy problematic; worker participation in improvement activities is crucial to deployment but more difficult to orchestrate. Even in small organizations there may be resistance to change, perhaps based on valid concerns, and addressing resistance should be part of the organization's learning process. Similarly, even small organizations have to focus on the "vital few" improvement issues, especially given their limited resources, which reemphasizes the Level 2 priorities for improvement.

During the *Acting* phase organizations should take advantage of the more detailed components of the CMM: subpractices and examples. These informative components are useful in characterizing an adequate process, yet they do not specify a particular implementation. For example, the estimating practices in Software Project Planning have subpractices on using historical data, but they do not specify a cost model, such as COCOMO, Price-S, or Slim, or even state that a cost model should be used.

For a small organization, the loop to the *Learning* phase may be much tighter than for large organizations. A one-year improvement cycle may be realistic, where large organizations will more typically take 2-3 years between assessments.[1] This reinforces the need for a low-overhead assessment process.

[1] Note that results for specific improvement actions should be observed well before the organizational improvement cycle delineated by an assessment rolls around.

4. Where Does the Software CMM Apply?

The CMM is intended to provide good software engineering and management practices for any project in any environment. The model is described in a hierarchy, as shown in Figure 2.

Maturity levels	(5 levels)
→ **Key process areas**	(18 KPAs)
→ **Goals**	(52 goals)
→ *Key practices*	(316 key practices)
® *Subpractices and examples*	(many)

Figure 2. The CMM structural hierarchy.

The "normative" components of the CMM are maturity levels, key process areas, and goals. All practices in the CMM are informative as opposed to normative. Since the detailed practices primarily support large, contracting software organizations, they are not necessarily appropriate, as written, for direct use by small projects and small organizations – but they do provide insight into how to achieve the goals and implement repeatable, defined, measured, and continually improving software processes. This emphasizes the need for reasonable and professional practices and helps prevent such "processes" as the estimating procedure that was simply "Go ask George."

In general, key process areas and goals are always relevant to any environment, with the exception of *Software Subcontract Management*, which may be "not applicable" if there is no subcontracting. In contrast, there are no circumstances under which *Peer Reviews* could be reasonably tailored out for a Level 3 organization. Deciding what is "not applicable" or an "alternate implementation" is a matter of competent professional judgment, implying a need for trained, experienced assessors and process definers, even for small organizations.

4.1 Getting to Level 2

At Level 2, the CMM emphasis is on managing software projects. The question for small organizations is whether the ability to manage projects is a crucial business concern. If the organization is doing contract work, the answer is almost certainly yes. If the organization is doing commercial shrinkwrap development, this issue is more debatable. Will bad management lead to serious business consequences? For many successful shrinkwrap software companies, no new product in company history has ever been finished on time or provided the functionality originally envisaged [Moody95]. It is unlikely, however, that bad management has contributed to the success of the thriving shrinkwrap companies, and it has certainly contributed to the failure of many that have not survived.

Requirements Management. Documented customer (system) requirements and communication with customer (and end users) are always important, although the documentation may be as simple as a one-page letter of intent. Documenting the requirements can be a challenge in a rapid prototyping environment or for R&D. The requirements may be scattered across several prototypes and actively evolving. This is acceptable so long as the history is maintained, and the requirements are consolidated before getting out of hand, i.e., too many prototypes capturing different potential requirements or before transitioning to full-scale development. The customer requirements / prototype functionality are similar to a lab notebook in the R&D context. Always document commitments and the requirements for the work to be performed – these documents are crucial for clarification and conflict resolution as the project progresses.

Software Project Planning. The #1 factor in successful process definition and improvement is "planfulness" [Curtis96]. Planning is needed for every major software process, but within the bounds of reasonable judgment, the organization determines what "major" means and how the plan should be packaged. A plan may reside in several different artifacts or be embedded in a larger plan.

Maintenance work is frequently level-of-effort. Problem reports and change requests are used to identify tasks that are performed in some identified priority order. A "project" is the next baseline update or major revision, which consists of a collection of changes to the basedlined software that must be carefully controlled. Similarly, R&D projects are likely to be level-of-effort. Examples of key practices that may provide minimal value to small projects include the size estimating and risk identification practices – and both may be critical for many small projects. For small projects, however, effort and schedules may be estimated directly from a work breakdown structure, and the consequences of the project failing may be relatively minor.

A project management plan that includes a work breakdown structure and any external commitments is always important, although it may be simple and concise. It is also a good practice to document internal commitments, especially those that cross organizational boundaries.

Software Project Tracking & Oversight. Knowing what you have accomplished versus what you have committed to is always important. Admittedly, managers have to be careful not to over-control – asking for more progress data than provides true insight. Gantt charts and earned value are popular and reasonably effective mechanisms for tracking progress, with two caveats. First, work packages should be binary - done or not done - since detailed estimates of work completed (e.g., we're 87% done) are notoriously inaccurate. This has implications for how the packages should be defined. A useful rule of thumb is to track work at a granularity somewhere between 2-3 weeks for a work package and 2-3 work packages per week. Second, remember the critical path to project completion (and possible resource conflicts) when tracking progress.

Management by fact is a paradigm shift for most organizations, which must be based on a measurement foundation. To make data analysis useful, you need to understand what the data means and how to analyze it, which implies collecting a simple set of <u>useful</u> data. The effort of collecting the data needs to be minimized so the small organization is not overwhelmed by overhead. This is also a problem for large organizations, but the margin for error is razor-thin for small organizations.

Software Subcontract Management. For small organizations, rapid prototyping projects, maintenance shops, and R&D outfits, Software Subcontract Management will usually be not applicable.

Software Quality Assurance. A small project is unlikely to have an independent SQA group, but it is always important to objectively verify that requirements are satisfied, including the process requirements in standards and procedures. It is usually a good idea to embed the QA function in your process via checklists, tools, etc., even if an independent SQA group exists.

Software Configuration Management. A small project is unlikely to need an SCM group or a Change Control Board, but configuration management and change control are always important. Always baseline work products and control changes to them. Microsoft, for example, uses daily builds to ensure that a stable baseline always exists [McConnell96].

Closing Level 2 Thoughts. Many of the context-sensitive, large-project implementation issues relate to organizational structure. Similar to the SCM and SQA examples above, an independent testing group may not be established, but testing is always necessary. The intent of a practice is important even if the implementation is radically different between small organizations and large. If one reads the CMM definition of "group," it states that "a group could vary from a single individual assigned part-time, to several part-time individuals assigned from different departments, to several individuals dedicated full-time." This flexibility (or ambiguity) is intended to cater to a variety of contexts. "Independence" is a consequence of organizational structure. "Objectivity" is the concept emphasized in the key process area goals.

4.2 Getting to Level 3

Organization Process Focus. At Level 3, the CMM emphasis is on consistency and organizational learning. In most organizations, a software engineering process group (SEPG) or some equivalent should be formed to coordinate process definition, improvement, and deployment activities. One of the reasons for dedicating resources to an SEPG is to ensure follow-through on appraisal findings. Many improvement programs have foundered simply because no action resulted from the appraisal. Small organizations may not have full-time SEPG staff, but the responsibility for improvement should be explicitly assigned and monitored. Thinking about the way you work and how to do better is always important, regardless of the models or standards that you may use to structure your thoughts. Always assign responsibility, authority, and accountability for process definition and improvement, whether an SEPG is formed or not.

Organization Process Definition. The reasons for documenting a process (or product) are to

1) communicate – to others now and perhaps to yourself later;
2) understand – if you can't write it down, you don't really understand it; and
3) encourage consistency – take advantage of repeatability.

This is a general characteristic of learning organizations – even small, R&D, or maintenance organizations. Documented processes are always important, and important processes should always be documented.

Documented processes support organizational learning and prevent reinventing the wheel for common problems – they put repeatable processes in place. Documents do not, however, need to be lengthy or complex to be useful. Keep the process simple. The CMM is about <u>doing</u> things, not <u>having</u> things. A 1-2 page process description may suffice, and subprocesses and procedures can be invoked as needed and useful. Use good software design principles, such as locality, information hiding, and abstraction, in defining processes.

The degree of formality needed for processes is a frequent challenge for both large and small organizations [Comer98, Hadden98a, Pitterman98, Sanders98]. Should there be separate procedures for each of the 25 key practices at Level 2 that mention "according to a documented procedure"]? The answer, as discussed in section 4.5.5 "Documentation and the CMM" of **The Capability Maturity Model: Guidelines for Improving the Software Process** [Paulk95], is a resounding NO! Packaging of documentation is an organizational decision.

Training Program. The reason for training is to develop skills. There are many "training mechanisms" other than formal classroom training that can be effective in building skills. One that should be seriously considered is a formal mentoring program. In this case, formality means going beyond assigning a mentor and hoping that experience will rub off. Formality implies training people on how to mentor and monitoring the effectiveness of the mentoring.

Training remains an issue after the initial deployment of a process or technology [Abbott97, Williams98]. As personnel change, the incremental need for training may not be adequately addressed. Mentoring and apprentice programs may suffice to address this issue, but they cannot be assumed to be satisfactory without careful monitoring.

Integrated Software Management. The thoughtful use of organizational assets (overcoming the "not-invented here" syndrome) is always important. Processes need to be tailored to the needs of the project [Ginsberg95, Ade96, Ahlgren96]. Although standard processes provide a foundation, most projects will also have unique needs. Unreasonable constraints on tailoring can lead to significant resistance to following the process. As Hoffman expresses it, "Don't require processes that don't make sense" [Hoffman98]. Use organizational assets, but use them intelligently.

Some argue that software project management is really risk management. This implies that you should use an incremental or evolutionary life cycle. If you want to focus on risk management, the spiral model may be the preferred life cycle model. If you want to focus on involving the customer, perhaps rapid prototyping or joint application design would be preferable. Few long-term projects have the luxury of the stable environment necessary for the waterfall life cycle to be the preferred choice – yet it is probably the most common life cycle. For small projects, however, the waterfall life cycle may be an excellent choice. In some cases, the risk to the organization if a small project fails is minimal, and formal risk management is not worth the overhead. R&D organizations, on the other hand, are continually pushing the envelope as they explore new areas, thus their intrinsic life cycle is evolutionary.

Software Product Engineering. Although some may disagree on the need for requirements analysis and design in small projects, thoughtfully defined and consistent software life cycle processes – requirements analysis, design, coding, testing, installation, operations, and maintenance – are always important. Always spend significant time on requirements analysis, design, and test planning. Always consider the entire life cycle in planning a development project.

Intergroup Coordination. For small projects and organizations, Intergroup Coordination may appear inapplicable because there are no "other groups" to coordinate with. This misses the point that this key process area is about communicating with the customer, documenting and tracking commitments, and resolving conflicts, which are as crucial for individuals as for different organizational entities. Communication and coordination are always important – even for one-person projects, communication over time and perhaps with those who succeed you is important.

Peer Reviews. Although you can argue over the best kind of peer review, the simple fact is that the benefits of peer reviews far outweigh their costs. Most effective is some variant of inspections, but any form of collegial or disciplined review, such as structured walkthroughs, adds significant value. R&D organizations reflect this by emphasizing the scientific method. Small organizations, however, may be more vulnerable to schedule pressure. Unfortunately, recognizing the value of peer reviews does not mean that we do them systematically, thus their placement at Level 3. Peer reviews are always important, and even recommended for Level 1 projects where the chaotic environment makes their use inconsistent.

4.3 Getting to Levels 4 and 5

Any organization embarking on Levels 4 and 5 needs little guidance in CMM interpretation principles. Small organizations should seriously consider the Personal Software Process^{SM} (PSP^{SM}) and Team Software Process^{SM} (TSP^{SM}) to bootstrap their process improvement efforts to Level 5 [Ferguson97, Hayes97]. Where the CMM addresses the organizational side of process improvement, PSP addresses building the capability of individual practitioners. The PSP course convinces the individual professional, based on his or her own data, of the value of a disciplined, engineering approach to building software. PSP and TSP take the individual and the team to a Level 5 process capability that the organization can leverage.

4.4 Miscellaneous Improvement Issues

Paquin identifies five issues for small organizations and projects [Paquin98]:

- assessments
- project focus
- documentation
- required functions
- maturity questionnaire

Since there are no "shall" statements in the CMM, there are no "required functions," but the CMM may describe more functions than there are people to fill the slots. This issue has been discussed as terminology or role mapping. The maturity questionnaire is a concern because it uses CMM terminology, which may be unclear to the people filling out the questionnaire. Expressing the questionnaire in the terminology of the organization is therefore a desirable precursor to even an informal assessment or survey. Assessments should be "light weight," and documentation should focus on minimum essential information.

Abbott identifies six keys to software process improvement in small organizations [Abbott97]:

- senior management support
- adequate staffing
- applying project management principles to process improvement
- integration with ISO 9001
- assistance from process improvement consultants
- focus on providing value to projects and to the business

Senior management support and adequate staffing are universal issues. If applying good project management to software projects is the best way to ensure success, then the same should be true for process improvement, which should be treated like any other project. ISO 9001 is more frequently an issue for large organizations than small, so it is interesting that Abbott points this out for his small company. The advise to use process improvement consultants can be problematic. The experience and expertise of a good consultant is unquestionably of great value, yet skilled consultants charge a premium price – frequently beyond what a small organization can afford. Unskilled consultants can be actively detrimental. Although none of the process guidance in the CMM or similar models and standards is particularly "rocket science," the pitfalls in changing individual, team, and organizational behaviors are non-trivial. There are no easy answers to this issue.

Brodman and Johnson identify seven small organization/small project challenges [Johnson97]:

- handling requirements
- generating documentation
- managing projects
- allocating resources
- measuring progress
- conducting reviews
- providing training

They have also developed a tailored version of the CMM for small businesses, organizations, and projects [Brodman96]. Although the majority of the key practices (but very few of the goals) in the CMM were tailored in the LOGOS Tailored CMM, they characterize these changes as:

- clarification of existing practices
- exaggeration of the obvious
- introduction of alternative practices (particularly as examples)
- alignment of practices with small business/small organization/small project structure and resources

Their analysis therefore agrees that the changes involved in tailoring the CMM for small organizations should not be considered radical.

5. Conclusion

To summarize this discussion and capture the essence of Level 3 in language that may communicate better with small projects and organizations, the recommendations for effective software processes include:

- document the requirements
- define a work breakdown structure and plan the work
- track significant accomplishments (no more than 2 or 3 per week)
- build the SQA function into the process (as a "buddy system" or part of peer reviews, with an escalation mechanism to resolve conflicts)
- determine how changes to work products will be identified and approved
- establish a configuration management system
- assign responsibility for defining and improving specific processes

- concisely document both management and engineering processes and standardize them
- document commitments and systematically resolve conflicts within the project
- install a peer review process, with a preference for inspections

This may seem (and be) simplistic, but this is the core of a disciplined process that with a philosophy of continual process improvement can lead even a small software organization to Level 5.

Many of the abuses of the Software CMM spring from a fear of what "others" may do, such as evaluate simple or alternative implementations adversely, thus leading to loss of a contract. If an organization applies common sense to the guidance in the CMM as guidance rather than requirements, then many of the interpretation problems for the model vanish. Although the CMM provides a significant amount of guidance in making judgments, removing subjectivity implies a deterministic, repetitive process that is not characteristic of engineering design work. That is why it is an abuse of the CMM to check off practices for conformance. When the CBA IPI method requires collecting data on each key practice[2], it is for making consistent and comprehensive judgments; it is not implying a requirement that each key practice be literally implemented. Even weaknesses against a key practice that result in findings do not necessarily result in failing to achieve a goal.

Some are unwilling or unable to interpret, tailor, or apply judgment. It is easy to mandate the key practices, but foolhardy, even when driven by concerns about customer intentions and competence. On more than one occasion I have heard someone say they were doing something that was foolish, but they were afraid that the customer was so ignorant or incompetent that they would be unable to understand the rationale for doing things differently than literally described in the CMM. This is particularly problematic if SCEs by the customer are feared. It is true that judgments may differ – and sometimes legitimately so. What is adequate in one environment may not suffice for a new project. That is why we recommend that process maturity be included in risk assessment during source selection, rather than using maturity levels to filter offerors [Barbour96].

Unfortunately there is no simple solution to this problem. In the SEI's CMM training, these points are repeatedly emphasized, but the problems persist. "Standards" such as the CMM can help organizations improve their software process, but focusing on achieving a maturity level without addressing the underlying process can cause dysfunctional behavior. Maturity levels should be measures of improvement, not goals of improvement. That is why we emphasize the need to tie improvement to business objectives.

The challenges in interpreting the CMM appropriately for different environments differ in degree rather than in kind. The bottom line is that software process improvement should be done to help the business – not for its own sake. The best advice comes from Sanjiv Ahuja, President of Bellcore: "Let common sense prevail!" The CMM is a proven tool to support process appraisal and software process improvement in a wide range of environments[3], but it must be used with professional judgment and common sense to be truly effective.

References

Abbott97 John J. Abbott, "Software Process Improvement in a Small Commercial Software Company," , **Proceedings of the 1997 Software Engineering Process Group Conference**, San Jose, CA, 17-20 March 1997.

---

[2] The CBA IPI method requires that each Activity Performed be investigated and the key practices in the other common features be sampled. I recommend investigating each key practice rather than just sampling the practices in the institutionalization common features.

[3] There are far too many case studies and analyses of CMM-based improvement to cite here, but those interested may wish to examine *http://www.sei.cmu.edu/cmm/cmm.articles.html#biblio.case.studies*.

Ade96	Randy W. Ade and Joyce P. Bailey, "CMM Lite: SEPG Tailoring Guidance for Applying the Capability Maturity Model for Software to Small Projects," **Proceedings of the 1996 Software Engineering Process Group Conference: Wednesday Papers**, Atlantic City, NJ, 20-23 May 1996.
Ahlgren96	Magnus Ahlgren, "CMM Light for SMEs," **Conference Notebook: The First Annual European Software Engineering Process Group Conference**, Amsterdam, The Netherlands, 26-27 June 1996, section C413.
Barbour96	Rick Barbour, "Software Capability Evaluation Version 3.0 Implementation Guide for Supplier Selection," Software Engineering Institute, Carnegie Mellon University, CMU/SEI-95-TR-012, April 1996.
Brodman96	Judith G. Brodman and Donna L. Johnson, **The LOGOS Tailored Version of the CMM for Small Businesses, Small Organizations, and Small Projects**, Version 1.0, August 1996.
Curtis96	Bill Curtis, "The Factor Structure of the CMM and Other Latent Issues," **Proceedings of the 1996 Software Engineering Process Group Conference: Tuesday Presentations**, Atlantic City, NJ, 20-23 May 1996.
Daskalantonakis94	Michael K. Daskalantonakis, "Achieving Higher SEI Levels," IEEE Software, Vol. 11, No. 4, July 1994, pp. 17-24.
DeMarco95	Tom DeMarco, **Why Does Software Cost So Much?**, ISBN 0-932633-34-X, Dorset House, New York, NY, 1995.
Dunaway96	Donna K. Dunaway and Steve M. Masters, "CMM-Based Appraisal for Internal Process Improvement (CBA IPI): Method Description," Software Engineering Institute, Carnegie Mellon University, CMU/SEI-96-TR-007, DTIC Number ADA307934, 1996.
Ferguson97	Pat Ferguson and Jeanie Kitson, "CMM-Based Process Improvement Supplemented by the Personal Software Process in a Small Company Environment," , **Proceedings of the 1997 Software Engineering Process Group Conference**, San Jose, CA, 17-20 March 1997.
Ginsberg95	Mark Ginsberg and Lauren Quinn, "Process Tailoring and the Software Capability Maturity Model," Software Engineering Institute, Carnegie Mellon University, CMU/SEI-94-TR-024, November 1995.
Hadden98a	Rita Hadden, "How Scalable are CMM Key Practices?" Crosstalk: The Journal of Defense Software Engineering, Vol. 11, No. 4, April 1998, pp. 18-20, 23.
Hadden98b	Rita Hadden, "Key Practices to the CMM: Inappropriate for Small Projects?" panel, Rita Hadden moderator, **Proceedings of the 1998 Software Engineering Process Group Conference**, Chicago, IL, 9-12 March 1998.
Hayes97	Will Hayes and James W. Over, "The Personal Software Process (PSP): An Empirical Study of the Impact of PSP on Individual Engineers," Software Engineering Institute, Carnegie Mellon University, CMU/SEI-97-TR-001, December 1997.
Hoffman98	Leo Hoffman, "Small Projects and the CMM," in "Key Practices to the CMM: Inappropriate for Small Projects?" panel, Rita Hadden moderator, **Proceedings of the 1998 Software Engineering Process Group Conference**, Chicago, IL, 9-12 March 1998.
Humphrey95	Watts S. Humphrey, **A Discipline for Software Engineering**, ISBN 0-201-54610-8, Addison-Wesley Publishing Company, Reading, MA, 1995.

Johnson97	Donna L. Johnson and Judith G. Brodman, "Tailoring the CMM for Small Businesses, Small Organizations, and Small Projects," Software Process Newsletter, IEEE Computer Society Technical Council on Software Engineering, No. 8, Winter 1997, p. 1-6.
McConnell96	Steve McConnell, **Rapid Development: Taming Wild Software Schedules** , Microsoft Press, Redmond, WA, 1996.
McFeeley96	Bob McFeeley, "IDEAL: A User's Guide for Software Process Improvement," Software Engineering Institute, Carnegie Mellon University, CMU/SEI-96-HB-001, February 1996.
Moody95	Fred Moody, **I Sing the Body Electronic**, Viking, Penguin Books, New York, NY, 1995.
Paquin98	Sherry Paquin, “Struggling with the CMM: Real Life and Small Projects,” in “Key Practices to the CMM: Inappropriate for Small Projects?” panel, Rita Hadden moderator, **Proceedings of the 1998 Software Engineering Process Group Conference**, Chicago, IL, 9-12 March 1998.
Paulk95	Carnegie Mellon University, Software Engineering Institute (Principal Contributors and Editors: Mark C. Paulk, Charles V. Weber, Bill Curtis, and Mary Beth Chrissis), **The Capability Maturity Model: Guidelines for Improving the Software Process**, ISBN 0-201-54664-7, Addison-Wesley Publishing Company, Reading, MA, 1995.
Paulk96	Mark C. Paulk, "Effective CMM-Based Process Improvement," **Proceedings of the 6th International Conference on Software Quality**, Ottawa, Canada, 28-31 October 1996, pp. 226-237.
Pitterman98	Bill Pitterman, “Key Practices to the CMM: Inappropriate for Small Projects?” panel, Rita Hadden moderator, **Proceedings of the 1998 Software Engineering Process Group Conference**, Chicago, IL, 9-12 March 1998.
Sanders98	Marty Sanders, “Small Company Action Training and Enabling,” in **The CMM and Small Projects**, Society for Software Quality Roundtable, Washington, DC, 26 January 1998.
Strigel95	Wolfgang B. Strigel, "Assessment in Small Software Companies," **Proceedings of the 1995 Pacific Northwest Software Quality Conference**, 1995, pp. 45-56.
Whitney94	Roselyn Whitney, Elise Nawrocki, Will Hayes, and Jane Siegel, "Instant Profile: Development and Trial of a Method to Measure Software Engineering Maturity Status," CMU/SEI-94-TR-4, Software Engineering Institute, Carnegie Mellon University, March 1994.
Williams98	Louise B. Williams, “SPI Best Practices for ‘Small’ Projects,” in **The CMM and Small Projects**, Society for Software Quality Roundtable, Washington, DC, 26 January 1998.

Acknowledgements. I would like to thank the folks who reviewed this paper: Judi Brodman, Rita Hadden, Stuart Locklear, Henry Mendenhall, Gladys Mercier, Sherry Paquin, Kathy Paulk, Pedro Pinto, Marty Sanders, Francisco Valeriano, and Bin-Lan Woo, plus the anonymous reviewers. Any errors are sadly mine.

Analyzing the Conceptual Relationship Between ISO/IEC 15504 (Software Process Assessment) and the Capability Maturity Model for Software

Mark C. Paulk
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA, USA

Abstract

The Capability Maturity Model® for Software (Software CMM®) is probably the best known and most widely used model world-wide for software process improvement. ISO/IEC 15504 is a suite of standards currently under development for software process assessment, which can be expected to affect the continuing evolution of the Software CMM. This paper discusses the similarities and differences between the two models and how they may influence each other as they both continue to evolve.

Introduction

The Software Engineering Institute (SEI) has been evolving a process maturity framework now known as the Capability Maturity Model for Software (Software CMM) since 1986 [Paulk95a, Paulk95c]. This model provides organizations with guidance for measuring software process maturity and establishing process improvement programs. The Software CMM is probably the best known and most widely used model world-wide for software process improvement at this writing.

ISO/IEC[1] 15504 is a suite of standards for software process assessment currently under development as an international standard . ISO/IEC 15504 has been published as a type 2 technical report, which is a stage in the development of a standard. Of the nine parts to ISO/IEC 15504, the parts directly relevant to the Software CMM are ISO/IEC 15504-2, the reference model, and ISO/IEC 15504-5, which provides an example model. As an international standard, ISO/IEC 15504 can be expected to affect the continuing evolution of CMM-related products.

® Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark Office.
The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.
[1] ISO/IEC 15504 is being developed by working group 10 (WG10) under the software engineering subcommittee (SC7) of the information technology joint committee (JTC1) established by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC).

This paper provides an overview of the Software CMM and ISO/IEC 15504, discusses the similarities and differences between them, and speculates how they may influence each other as they both continue to evolve.

The Capability Maturity Model for Software

The Capability Maturity Model for Software describes the principles and practices underlying software process maturity and is intended to help software organizations improve the maturity of their software processes in terms of an evolutionary path from ad hoc, chaotic processes to mature, disciplined software processes. The Software CMM is organized into five maturity levels, described in Table 1.

Table 1. Software CMM Maturity Levels.

Software CMM Maturity Level	**Description of Software CMM Maturity Levels**
1) Initial	The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.
2) Repeatable	Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
3) Defined	The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.
4) Managed	Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.
5) Optimizing	Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

Except for Level 1, each maturity level is decomposed into several key process areas that indicate the areas an organization should focus on to improve its software process. The key process areas in Version 1.1 of the Software CMM are listed in Table 2.

For convenience, the key process areas are internally organized by common features. The common features are attributes that indicate whether the implementation and institutionalization of a key process area is effective, repeatable, and lasting. The five common features are Commitment to Perform, Ability to Perform, Activities Performed, Measurement and Analysis, and Verifying Implementation. General practices that apply to every key process area at every maturity level are categorized by the common features. For example, establishing policies is a common practice in Commitment to Perform and providing training is a common practice in Ability to Perform.

Each key process area is described in terms of the key practices that contribute to satisfying its goals and that are allocated to the common features. The key practices describe the specific infrastructure and activities that contribute most to the effective implementation and institutionalization of the key process area.

Table 2. The Key Process Areas in the Software CMM.

Level	Focus	Key Process Areas
5 Optimizing	*Continuous process improvement*	Defect Prevention Technology Change Management Process Change Management
4 Managed	*Product and process quality*	Quantitative Process Management Software Quality Management
3 Defined	*Engineering processes and organizational support*	Organization Process Focus Organization Process Definition Training Program Integrated Software Management Software Product Engineering Intergroup Coordination Peer Reviews
2 Repeatable	*Project management processes*	Requirements Management Software Project Planning Software Project Tracking & Oversight Software Subcontract Management Software Quality Assurance Software Configuration Management
1 Initial	*Competent people and heroics*	

ISO/IEC 15504 -- Software Process Assessment

ISO/IEC 15504 is intended to harmonize the many different approaches to software process assessment. It has nine parts:

Part 1 : *Concepts and introductory guide*
Part 2 : *A reference model for processes and process capability*
Part 3 : *Performing an assessment*
Part 4 : *Guide to performing assessments*
Part 5 : *An assessment model and indicator guidance*
Part 6 : *Guide to competency of assessors*
Part 7 : *Guide for use in process improvement*
Part 8 : *Guide for use in determining supplier process capability*
Part 9 : *Vocabulary*

The reference model in Part 2 documents the set of universal software engineering processes that are fundamental to good software engineering and that cover best practice activities. It describes processes that an organization may perform to acquire, supply, develop, operate, evolve and support software and the process attributes that characterize the capability of those processes. The purpose of the reference model is to provide a common basis for different models and methods for software process assessment, ensuring that results of assessments can be reported in a common context.

The reference model architecture is two dimensional. The *process dimension* is characterized by process purpose statements, which are the essential measurable objectives of a process. The processes are listed in Table 4.

The *process capability dimension* is characterized by a series of process attributes, applicable to any process, which represent measurable characteristics necessary to manage a process and improve its capability to perform. Each process attribute describes an aspect of the overall capability of managing and improving the effectiveness of a process in achieving its purpose and contributing to the business goals of the organization. There are nine process attributes, which are grouped into capability levels, one at capability level 1 and two each at levels 2-5. Capability levels constitute a rational way of progressing through improvement of the capability of any process. The underlying principles are the same conceptually as the Software CMM maturity levels, although targeted to the process rather than the organization. The six capability levels are described in Table 3.

Table 3. The Capability Levels in ISO/IEC 15504-2.

Capability Level	ISO/IEC 15504-2 Capability Level Description
Level 0 *Incomplete*	There is general failure to attain the purpose of the process. There are little or no easily identifiable work products or outputs of the process.
Level 1 *Performed*	The purpose of the process is generally achieved. The achievement may not be rigorously planned and tracked. There are identifiable work products for the process, and these testify to the achievement of the purpose.
Level 2 *Managed*	The process delivers work products according to specified procedures and is planned and tracked. Work products conform to specified standards and requirements.
Level 3 *Established*	The process is performed and managed using a defined process based upon good software engineering principles. Individual implementations of the process use approved, tailored versions of standard, documented processes to achieve the process outcomes.
Level 4 *Predictable*	The defined process is performed consistently in practice within defined control limits, to achieve its defined process goals.
Level 5 *Optimizing*	Performance of the process is optimized to meet current and future business needs, and the process achieves repeatability in meeting its defined business goals.

The process attributes are defined in ISO/IEC 15504-2 and elaborated in ISO/IEC 15504-5 by process indicators, called generic practices in earlier drafts of the evolving standard.

Relating ISO/IEC 15504 Processes to CMM Key Process Areas

The mapping in Table 4 shows how the topics in ISO/IEC 15504 relate to the equivalent topics in the Software CMM. Topics are typically not isomorphic but are highly correlated. Anyone adequately implementing, for example, the Configuration Management Process in ISO/IEC 15504 could reasonably expect to have satisfied the Software Configuration Management key process area in the Software CMM. Topics are not usually isomorphic because of extensions that may have been added or different levels of abstraction that may have been chosen (e.g., the Development Process in ISO/IEC

12207 addresses the same set of concerns as the Software Product Engineering key process area in the Software CMM). "Subprocesses" and activities in Table 4 are in italics to emphasize that they are components of a larger construct. Where the relationship is indirect, the Software CMM component is in parentheses to highlight the difference in scope.

Table 4. Mapping Between ISO/IEC 15504-2 Processes and Software CMM Key Process Areas.

ISO/IEC 15504 Processes	**Software CMM v1.1**
CUS.1 Acquisition	Software Subcontract Management
CUS.1.1 Acquisition preparation	*Software Subcontract Management, Activity 1*
CUS.1.2 Supplier selection	*Software Subcontract Management, Activity 2*
CUS.1.3 Supplier monitoring	*Software Subcontract Management, Activities 5 and 7-11*
CUS.1.4 Customer acceptance	*Software Subcontract Management, Activity 12*
CUS.2 Supply[2]	(Software Project Planning; Software Project Tracking & Oversight; Software Product Engineering)
CUS.3 Requirements elicitation	
CUS.4 Operation	
CUS.4.1 Operational use	
CUS.4.2 Customer support	
ENG.1 Development	Software Product Engineering
ENG.1.1 System[3] requirements analysis and design	
ENG.1.2 Software requirements analysis	*Software Product Engineering, Activity 2*
ENG.1.3 Software design	*Software Product Engineering, Activity 3*
ENG.1.4 Software construction	*Software Product Engineering, Activity 4*
ENG.1.5 Software integration	*Software Product Engineering, Activity 6*

[2] The Supply Process deals with providing software to the customer that meets the agreed requirements. Establishing a contract, developing the software, and delivering it to the customer, which are the issues for this process, are addressed in various key process areas, although the Supply Process itself is not explicitly specified in the Software CMM.

[3] Systems engineering issues, e.g., requirements elicitation, system analysis, and system testing, were not considered within the scope of the Software CMM in version 1.1.

ISO/IEC 15504 Processes	**Software CMM v1.1**
ENG.1.6 Software testing	*Software Product Engineering, Activity 7*
ENG.1.7 System integration and testing	*(Software Product Engineering, Activities 6 and 7)*
ENG.2 System and software maintenance	
SUP.1 Documentation	*Software Product Engineering, Activity 8*
SUP.2 Configuration management	Software Configuration Management
SUP.3 Quality assurance	Software Quality Assurance
SUP.4 Verification	(Peer Reviews; *Software Product Engineering, Activities 5 and 6*)
SUP.5 Validation	*Software Product Engineering, Activity 5*
SUP.6 Joint review	*Software Project Tracking & Oversight, Activity 13*
SUP.7 Audit	(Software Quality Assurance)[4]
SUP.8 Problem resolution	*Software Configuration Management, Activity 5*
MAN.1 Management[5]	(Software Project Planning; Software Project Tracking & Oversight; Integrated Software Management)
MAN.2 Project management	Software Project Planning; Software Project Tracking & Oversight; Integrated Software Management
MAN.3 Quality management	Software Quality Management
MAN.4 Risk management	*Software Project Planning, Activity 13; Software Project Tracking & Oversight, Activity 10; Integrated Software Management, Activity 10*
ORG.1 Organizational alignment[6]	
ORG.4 Infrastructure	Organization Process Definition

[4] SQA covers both quality assurance and audits. Audits are an independent QA function. The SQA key process area can be implemented as an independent function or not; the requirement is objective verification rather than independent verification. SQA may, or may not, therefore cover the Audit Process in a particular environment.

[5] This is a generic planning and management process that is to be applied to any process, rather than project planning specifically.

[6] The purpose of the Organizational Alignment Process is to ensure that individuals share a common vision, culture, and understanding of business goals.

ISO/IEC 15504 Processes	Software CMM v1.1
ORG.2 Improvement	Organization Process Definition
ORG.2.1 Process establishment	Organization Process Definition
ORG.2.2 Process assessment	*Organization Process Focus, Activity 1*
ORG.2.3 Process improvement	Organization Process Focus; (Process Change Management)
ORG.3 Human resource management	Training Program
ORG.4 Infrastructure	
ORG.5 Measurement	*Measurement and Analysis (common feature)*
ORG.6 Reuse	
	Requirements Management
	Intergroup Coordination
	Peer Reviews[7]
	Quantitative Process Management[8]
	Defect Prevention
	Technology Change Management
	Process Change Management

Relating the Process Capability Dimension to Maturity Levels

The Software CMM is sometimes characterized as a *staged* model because it describes organizational capability in terms of maturity levels that represent evolutionary stages of capability, and the ISO/IEC 15504 model is sometimes, perhaps less accurately, described as a *continuous* model. The ISO/IEC 15504 model describes the terrain of software process maturity from the perspective of the individual process, where the Software CMM provides a roadmap for organizational improvement.

A *staged* model can be described as:

- an *organization-focused* model, since its target is the organization's process capability,
- a *descriptive* model, because it describes organizations at different levels of achieved capability,
- a *prescriptive* or normative model, since it prescribes how an organization should improve its processes.

[7] Indirectly covered by SUP.4 Verification process.
[8] Indirectly covered by ORG.5 Measurement process.

A staged architecture focuses on software process improvement and, in the case of the Software CMM, provides 500 pages of mostly informative material on software processes that has been prioritized by being in key process areas. The rating components, i.e., the key process areas and goals, are a comparatively small part of the document; there are 18 key process areas and 52 goals.

The term *continuous* is not a strictly accurate description since the ISO/IEC 15504 architecture is also based on (capability) levels. Other descriptive terms that could be used include:

- a *process-focused* model, since its target is process capability,
- a *terrain* model, from the analogy to a description of the software process terrain, and
- a *reference* model, since its primary use is in assessment as the reference for rating processes.

One of the objectives of ISO/IEC 15504 is to create a way of measuring process capability, while avoiding a specific approach to improvement such as the SEI's maturity levels, so that the many different kinds of assessment, model, and their results, can be meaningfully compared to one another. The approach selected is to measure the implementation and institutionalization of specific processes; a process measure rather than an organization measure. Maturity levels can be viewed as sets of process profiles using this approach [Paulk94, Paulk95a, Paulk96]. This addresses one of the deficiencies in the staged approach: lower maturity key process areas evolve with the organization's maturity. For example, there are organizational standards and required training for Software Configuration Management in a maturity level 3 organization, even though this is not explicitly stated in the Software CMM.

Differences Between ISO/IEC 15504 and the Software CMM

Both the staged and continuous perspectives have value, and they are conceptually compatible, but there is a fundamental philosophical difference between the two architectures. This philosophical difference implies strengths and weaknesses for both architectures.

Characteristic	**Staged Architecture (Software CMM)**	**Continuous Architecture (ISO/IEC 15504)**
Vital few	Attention is focused on the "vital few" issues in process improvement that are generally true for any organization.	Less important process issues can drown out the "vital few" issues when there are clashes over improvement priorities.
Organization capability	Organizational capability is explicitly described in terms of maturity levels.	Organizational capability is implicit; it can be intuitively understood by looking at the organizational processes, the process attributes, and their dependencies.
Process evolution	Key process areas are a	The evolution of processes

Characteristic	**Staged Architecture (Software CMM)**	**Continuous Architecture (ISO/IEC 15504)**
	snapshot of the evolving process.	from ad hoc to continuously improving is "fully" described.
Guidance	Extensive guidance in the key practices and subpractices provides significant help in understanding what a key practice or goal means, although it is typically oriented towards the practices of large organizations and projects in a contracting environment.	Abstract processes and process attributes can be difficult to interpret. No particular organizational improvement path is prescribed.
Extendibility	It may be difficult for the non-expert to extend the CMM principles to new disciplines or focus areas.	Adding processes and integrating with other models is a relatively straightforward definition, with the application of the capability dimension for rating the processes.

Some processes are “invisible” in a staged model, until the point that focusing on their improvement becomes critical to organizational maturity. Engineering processes, for example, are not a focus of maturity level 2, so they “suddenly appear” at level 3. Level 1 organizations perform engineering processes, but they are not represented in the Software CMM until level 3. This is intrinsic to the way the maturity levels are defined: the critical problems for level 1 organizations are managerial, not technical, so the improvement focus is not on the engineering processes at level 2.

This focus on the "vital few" processes at each maturity level for building organizational capability becomes a challenge when layering a staged model on top of a continuous architecture. For example, should every process described in the continuous model be placed under quantitative or statistical control if the organization is to be characterized as level 4 or higher? The staged model lets the decision of what processes should be quantitatively or statistically controlled be driven by business objectives. No rule, other than "all processes," has been articulated yet for aggregating process capability levels to achieve an organizational maturity level when using a continuous model. Should all processes be standardized? Under statistical control? Optimal?

This is both a strength and a weakness of the layering approach to integrating staged and continuous models. It highlights standardizing and tailoring (capability level 3) as issues for level 2 key process areas in a maturity level 3 organization, but flexibility in applying these principles to all (versus critical) processes is desirable from a business objective perspective. Conversely, the improvement priorities in the staged model describe an "80% solution" to effective process improvement; organizations have unique improvement needs driven by their business needs and environment.

Key process areas are not processes. A process changes over time and hopefully matures. A process is dynamic. A key process area is a static description of essential attributes (i.e., key practices) of a process when that process is fully realized, and it does not tell how the process is performed.

Usability was an issue in early drafts of ISO/IEC 15504 when 26 generic practices were rated on the capability dimension rather than nine process attributes. This lead to over 1,000 rating decisions during an assessment, and early trials indicated that assessments could be quite lengthy [Woodman96]. The solution to this problem was to "raise the level of abstraction" in rating to process attributes.

The Systems Engineering CMM [Bate95] uses the 26 generic practices in its instantiation of the continuous architecture (with a different rating philosophy). EIA Interim Standard 731 ("Systems Engineering Capability, Part 1: Model") defines 12 generic practices in its variation of a continuous architecture.

For Software CMM v2, we proposed addressing this concern via a goal for each key process area to capture the institutionalization of the process. The institutionalization goal would have captured the topics addressed by ISO/IEC 15504 process attributes via key practice templates for planning, training, tailoring, etc., as appropriate for the maturity level. Practices in the institutionalization common features – Commitment to Perform, Ability to Perform, Measurement and Analysis, and Verifying Implementation – would have mapped to this goal. This would have clarified that institutionalization is a critical part of satisfying a key process area and separated institutionalization and implementation for purposes of rating key process areas.

The difference in focus – organization versus process – thus leads to some subtle challenges in integrating the staged and continuous architectures. The similarities between capability and maturity levels make the relationships conceptually straightforward; the philosophical difference makes the operational details tricky.

Conclusions

The SEI is continuing to evolve the CMM concepts, primarily in its current work on CMM integration, which addresses software, systems engineering, and integrated process and product development. ISO/IEC JTC1/SC7/WG10 is continuing to refine ISO/IEC 15504 and is currently targeting 2001 for release of the international standard.

One of the challenges in harmonizing the CMM and ISO/IEC 15504 is defining the rating components for the process capability dimension. Mapping between one "goal" per level and two "process attributes" per level may lead to unacceptable granularity problems. The other major challenge is developing an acceptable operational mapping between organizational and process capability. Both SEI and WG10 continue to discuss these and other issues.

References

Bate95	Roger Bate, Dorothy Kuhn, Curt Wells, et al, "A Systems Engineering Capability Maturity Model, Version 1.1," Software Engineering Institute, Carnegie Mellon University, CMU/SEI-95-MM-003, November 1995.
Paulk94	Mark C. Paulk and Michael D. Konrad, "Measuring Process Capability Versus Organizational Process Maturity," **Proceedings of the 4th International Conference on Software Quality**, ASQC, Washington, DC, 3-5 October 1994.
Paulk95a	Mark C. Paulk, Michael D. Konrad, and Suzanne M. Garcia, "CMM Versus SPICE Architectures," IEEE Computer Society Technical Council on Software Engineering, Software Process Newsletter, No. 3, Spring 1995, pp. 7-11.
Paulk95b	Carnegie Mellon University, Software Engineering Institute (Principal Contributors and Editors: Mark C. Paulk, Charles V. Weber, Bill Curtis, and Mary Beth Chrissis), **The Capability Maturity Model: Guidelines for Improving the Software Process**, Addison-Wesley Publishing Company, Reading, MA, 1995.
Paulk95c	Mark C. Paulk, "The Evolution of the SEI's Capability Maturity Model for Software," Software Process: Improvement and Practice, Vol. 1, Pilot Issue, Spring 1995, pp. 3-15.
Paulk96	Mark C. Paulk, "Process Improvement and Organizational Capability: Generalizing the CMM," **Proceedings of the ASQC's 50th Annual Quality Congress and Exposition**, Chicago, IL, 13-15 May 1996, pp. 92-97.
Woodman96	Ian Woodman and Robin Hunter, "Analysis of Assessment Data from Phase One of the SPICE Trials," Software Process Newsletter, IEEE Computer Society Technical Council on Software Engineering, No. 6, Spring 1996, pp. 5-9.

APPLYING SPC TO THE PERSONAL SOFTWARE PROCESS

Mark C. Paulk
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA, USA

ABSTRACT

In recent years, a growing number of software organizations have begun to focus on applying the concepts of statistical process control (SPC) to the software process, usually as part of an improvement program based on the Software CMM.® There are a number of technical challenges to the successful use of these statistical techniques, primarily centered on the issues associated with high variation between individual software professionals. A growing number of organizations, however, are demonstrating that SPC techniques can be applied to the software process, even if questions remain on the specific processes, measures, and statistical techniques that will provide significant business value. This paper illustrates the application of the XmR control chart to the Personal Software Process. SM

Introduction

During the last decade, the focus of software process improvement has been on fundamental project management and organizational learning issues. In recent years, more mature organizations have begun to focus on applying the concepts of statistical process control (SPC) to the software process, specifically control charts, along with other statistical and process modeling techniques. There are a number of technical challenges to the successful use of these statistical techniques, primarily centered on the issues associated with high variation between individual software professionals. Studies of the differences between software professionals have ranged from 10:1 up to 100:1, with 20:1 differences being fairly common (Curtis, 1990; DeMarco, 1999). For this and other reasons, some software professionals have questioned the validity and business value of SPC in the software arena (Ould, 1996; Carleton, 1999; Kan, 1995).

® Capability Maturity Model and CMM are registered with the U.S. Patent and Trademark Office.
SM Personal Software Process and PSP are service marks of Carnegie Mellon University.
The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

A growing number of organizations, however, are demonstrating that SPC techniques can be applied to the software process, even if questions remain on the specific processes, measures, and statistical techniques that will provide significant business value (Florac, 1999; Paulk, 2000; Florac, 2000; Weller, 2000).

The term *software process* refers not just to an organization's overall software process, but to any process or subprocess used by a software project or organization. The Software CMM suggests it is at the subprocess level of activities and tasks that true quantitative management of the process occurs. At the macro level that management focuses on, the emphasis is more on risk management in an uncertain world than process control in a statistical sense, where software professionals focus on the day-to-day work. Thus, the concept of software process can be viewed as applying to any identifiable activity that is undertaken to produce or support a software product or service. This includes planning, estimating, designing, coding, testing, inspecting, reviewing, measuring, and controlling, as well as the subtasks and activities that comprise these undertakings.

The purpose of this paper is to apply a specific SPC technique, the XmR chart, to data from the Personal Software Process ^{SM} (PSP^{SM}). This is intended to illustrate the use of the XmR chart and to test the stability and predictability of the PSP processes.

The Personal Software Process

PSP is taught as a one-semester university course or a multi-week industry training course. It typically involves the development of ten programs, using increasingly sophisticated processes. PSP is a measurement-driven approach, where the student's own data drives the learning process as incrementally more powerful processes are adopted. SPC is not taught as part of PSP or TSP; the primary statistical technique used is prediction intervals for estimating size, effort, and defects.

There are seven process levels used to introduce the PSP; each level builds on the prior level by adding a few process steps to it. This minimizes the impact of process change on the engineer, who needs only to adapt the new techniques into an existing baseline of practices.

The processes are used on the last four PSP programming assignments and can be considered both rigorous and mature software processes. In an earlier analysis of the PSP data (Hayes, 1997), the data was separated for analysis into three "bands" for PSP0, PSP1, and PSP2. In this analysis, we will focus on the last three PSP processes: PSP2, PSP2.1, and PSP3. Design review and code review are introduced at PSP2 to help engineers achieve 100% yield: all defects removed before compiling the program. These are personal reviews conducted by an engineer on his or her own design or code. They are structured, data-driven review processes that are guided by personal review checklists derived from the engineer's historical defect data.

There are three basic measures in the PSP: development time, defects, and size. All other PSP measures are derived from these three basic measures. Minutes are the unit of measure for development time. A defect is defined as any change that must be made to the design or code in order to get the program to compile or test correctly.

Lines of code were chosen as the size measure for PSP because they can be automatically counted, precisely defined, and are well correlated with development effort. Size is also used to normalize other data, such as productivity (LOC per hour) and defect density (defects per KLOC). Each PSP program involves some amount of new development, enhancement, and/or reuse. The total LOC in a program will have several different sources, including some new LOC, some existing LOC that may have been modified, and some reused LOC. Developing new or modified LOC represents most of the programming effort in the PSP course; consequently, new and changed LOC is the basis for most size measurement in the PSP.

Statistical Process Control

What is statistical process control? What is implied by SPC? Operationally, statistical process control implies the use of seven basic tools (Ishikawa, 1986):

- flow charts
- scatter diagrams
- histograms
- Pareto analysis
- cause-and-effect (fishbone) diagrams
- run (trend) charts
- control charts

While many other techniques are associated with SPC, SPC always implies the use of control charts. A control chart is a run chart with limits added that indicate the normal execution of the process. The control limits are normally based on 3-sigma boundaries for the underlying common cause system that the process represents. Control charts provide a statistical method for distinguishing between variation caused by normal process operation and variation caused by anomalies in the process.

The basis for control charts is recognition of two types of variation – common cause variation and assignable cause variation. Common cause variation is variation in process performance due to normal or inherent interaction among the process components (people, machines, material, environment, and methods). Common cause variation of process performance is characterized by a stable and consistent pattern over time. Variation in process performance due to common cause is thus random, but will vary within predictable bounds. When a process is stable, the random variations that we see all come from a constant system of chance causes. The variation in process performance is predictable, and unexpected results are extremely rare.

Predictable is synonymous with "in control" or "stable." The other type of variation in process performance is due to assignable causes, also known as special causes. Assignable cause variation has marked impacts on product characteristics and other measures of process performance. These impacts create significant changes in the patterns of variation. Assignable cause variations arise from events that are not part of the normal process. They represent sudden or persistent abnormal changes to one or more of the process components. These changes can be in things such as inputs to the process, the environment, the process steps themselves, or the way in which the process steps are executed.

When all assignable causes have been removed and prevented from recurring in the future so that only a single, constant system of chance causes remains, we have a stable and predictable process.

The simplest rule for detecting a signal (a possible assignable cause) is when a point falls outside the 3-sigma control limits. Many other sets of detection rules have been proposed, which both make the control chart more sensitive to signals and also leads to a greater number of false alarms. The decision on which detection rules to use should be based on the economic trade-off between sensitivity and unnecessary work. Four common detection rules (Wheeler, 1992) are:

Detection Rule One	A lack of control is indicated whenever a single point falls outside the (three-sigma) control limits.
Detection Rule Two	A lack of control is indicated whenever at least two out of three successive values fall on the same side of, and more than two sigma units away from, the central line.
Detection Rule Three	A lack of control is indicated whenever at least four out of five successive values fall on the same side of, and more than one sigma unit away from, the central line.
Detection Rule Four	A lack of control is indicated whenever at least eight successive values fall on the same side of the central line.

When a process is stable, or nearly so, the 3-sigma limits determine the amount of variation that is normal or natural to the process. This is the "voice of the process" or the process telling us what it is capable of doing. This may or may not be satisfactory to the customer. If the performance is satisfactory, the process is "capable;" if the performance is not satisfactory, then the process must be changed since the variation is intrinsic to the process.

Many control charts use homogeneous, or rational, subgroups when measuring. We try to group the data so that assignable causes are more likely to occur between subgroups than within them -- thus "homogeneous" or "common cause" are useful adjectives. Control limits become wider and control charts less sensitive to assignable causes when containing non-homogeneous data. For many software processes, however, it will be more desirable to plot individual data points rather than the averages of subgroups. The most commonly used chart for individual data is the XmR chart, although other charts that take advantage of knowledge about the statistical distribution of the data can also be used when appropriate. Examples include the u-chart for Poisson data and the p-chart for binomial data.

For XmR charts, the upper and lower control limits (or natural process limits) are calculated by

$UNPL_X = X_{bar} + 2.66\ (mR_{bar})$

$UNPL_X = X_{bar} - 2.66\ (mR_{bar})$

$UCL_R = 3.268\ (mR_{bar})$

where the "subscript bar" denotes the average of the individual X values and the moving ranges. The lower control limit for the moving range chart is always zero.

Challenges in Using Control Charts

When analyzing process performance data, it is crucial to identify all sources of variation in the process. If a conscious effort is not made to account for the potential sources of variation, variation may be inadvertently hidden or obscured that will help improve the process. Overly aggregated data come about in many ways, but the most common causes are

- poorly formulated operational definitions of product and process measures
- inadequate description and recording of context information
- working with data whose elements are combinations (mixtures) of values from non homogeneous sources or different cause systems

When measured values of continuous variables have insufficient granularity (i.e., are coarse and imprecise), the discreteness that results can mask the underlying process variation. Computations for X_{bar} and σ can then be affected, and individual values that are rounded or truncated in the direction of the nearest control limit can easily give false out-of-control signals.

"Chunky data" can result in points that fall outside the limits even when the underlying process is predictable (Wheeler, 1998). Excessive round-off is one way to create chunky data. Using measurement units that are too large is another way. If the number of possible range values, including zero, within the limits is three or less, then you have chunky data. When rates based on counts are placed on an XmR chart, the average count must be greater than 1.0 for the moving ranges to yield appropriate limits.

Because of the non-repetitive nature of software products and processes, some believe it is difficult to achieve homogeneity with software data. When more than two data values are placed in a subgroup, a judgment is being made that these values are measurements taken under essentially the same conditions and that any difference between them is due to natural or common variation. The primary purpose of homogeneity is to limit the amount of variability within the subgroup data. When unlike things are grouped together, the subgroups are said to be stratified. One way of addressing the homogeneity concern is by having subgroups of size one using the XmR chart.

Perhaps the greatest challenge to the use of control charts for controlling software processes is that different work products are produced by different members of the project team. The result is software process data that is typically aggregated across many individuals. The equivalent situation in a manufacturing environment, of placing data from different machines on the same control chart, would not be acceptable since this leads to mixing and stratification. This also leads to problems in preserving time sequence in the data, although this is arguably a minor issue in conjuction with combining data from multiple team members. Collecting data on an individual basis would address this, but could have severe consequences if there were any chance of motivational use of the data, e.g., during performance appraisals, which would lead to dysfunctional behavior if the measurement system is not comprehensive (Austin, 1996). Collecting data at the individual level would also significantly decrease the amount of the data available for any specific statistical analysis. Disaggregating process data by individual, by defect type, or by other categories may be critical to obtaining insight into separate common cause systems (Florac, 2000), and this may imply severe practical limits to the value of SPC for software processes.

On the other hand, it is not unrealistic to think that that processes and systems are subject to hundreds of cause-and-effect relationships (Wheeler, 1998). When every process is subject to many different cause-and-effect relationships, predictable processes are those where the net effect of the multiple causes is a sort of statics equilibrium, which can be characterized as the common cause system. Pyzdek comments that even companies producing one-of-a-kind products usually do so with the same equipment, employees, and facilities, and the key to controlling the quality of single parts is to concentrate on process elements rather than on products features, i.e., the repeatable (if not repetitive) process (Pyzdek, 1993). Group thinking has been characterized as being usually better, less variable, and more precise than individual thinking in statistical terms (Hare, 1995). How much of a problem these disaggregation issues will be in practice remains an area for empirical research.

"Informally Stabilizing" the Software Process

When looking at their process data, organizations typically discover that the defined processes used by the projects are not as consistently implemented or measured as believed (Paulk, 1999). When a process is being placed under statistical process control in a rigorous sense, it is "stabilized" by removing assignable causes of variation. "Informal stabilization" occurs simply by examining the data (graphically) before placing it on a control chart, as patterns in the data suggestive of mixing and stratification are observed.

Informally stabilizing the process can be characterized as an exercise in exploratory data analysis, which is a precursor to true quantitative management or statistical process control. Shewhart expressed this initial stage of investigation as the point where control had not yet been attained and there was as yet no statistical universe to examine (Shewhart, 1939). The software processes that are first stabilized tend to be design, code, and test, since there is usually an adequate amount of inspection and test data to apply statistical techniques in a fairly straightforward manner.

This can be expressed as refining the operational definitions of both the measures and the processes to be repeatable. One of the first concerns about process performance is compliance: is the process being executed properly, are the personnel trained, right tools available, etc. If the process is not in compliance, there is little chance of performing consistently or satisfactorily. "Measurement error" is a significant concern, frequently resulting from problems in building good operational definitions.

If a process is compliant, the next question is whether the process performance is reasonably consistent over time. Are the effort, cost, elapsed time, delivery, and quality consumed and produced by executing the process consistent? Realizing that variation exists in all processes, is the variation in process performance predictable?

The use of measurement data for evaluating the performance of employees is an ongoing concern for high maturity organizations. Deming was a strong advocate of statistical techniques and strongly averse to performance evaluations (Deming, 1986), declaring performance measurement "the most powerful inhibitor to quality and productivity in the Western world." This is arguably the most controversial of Deming's points. Austin has shown that the potential for dysfunction arises when any critical dimension of effort expenditure is not measured, and unless the latitude to subvert measures can be eliminated, i.e, measures can be made perfect, or a means established for preventing the motivational use of data, dysfunction is destined to accompany organizational measurement (Austin, 1996).

Empirical data indicates that high maturity organizations know what business they are in; establishing product lines is one important mechanism for capturing that vision (Besselman, 1995; Paulk, 2000). Product lines also offer a way to stratify the data that supports the ability to compare process and product measures meaningfully. Although there are other confounding factors, the differences between application domain clearly cause significant performance variation. Product lines, therefore, can be used to minimize the variability of the process data, that is, removing inappropriate aggregation. Other issues, such as complexity and size, will also be dealt with by the organization. The successful use of control charts that add insight and value is predicated on identifying comparable data.

There is a tension between having stable process and continual process improvement. High maturity organizations use "quantitative management" of their processes, but they are continually improving those processes. The process data for the previous process may not be valid for the new process, and new control limits may need to be recalculated on an ongoing basis. Even when most changes are incremental, compounded small changes can lead to dramatic improvements. Wheeler observes that continual improvement is a journey consisting of frequent, intermittent improvements, interspersed with alternating periods of predictable and unpredictable performance (Wheeler, 1998).

Using XmR Charts on PSP Data

Control charts can be a useful tool in the later stages of "informally stabilizing" the software process. For process consistency, for example, it is important that the inspection process be a stable one, therefore placing review rates on a control chart can identify atypical inspections. A similar step could be taken for the production process, but the data used in this analysis is the same data set as used in the 1997 analysis (Hayes, 1997), and the information on reused LOC was not saved for that data set. Since the amount of reuse is an important attribute of the production process, we will focus on the inspection process in this analysis.

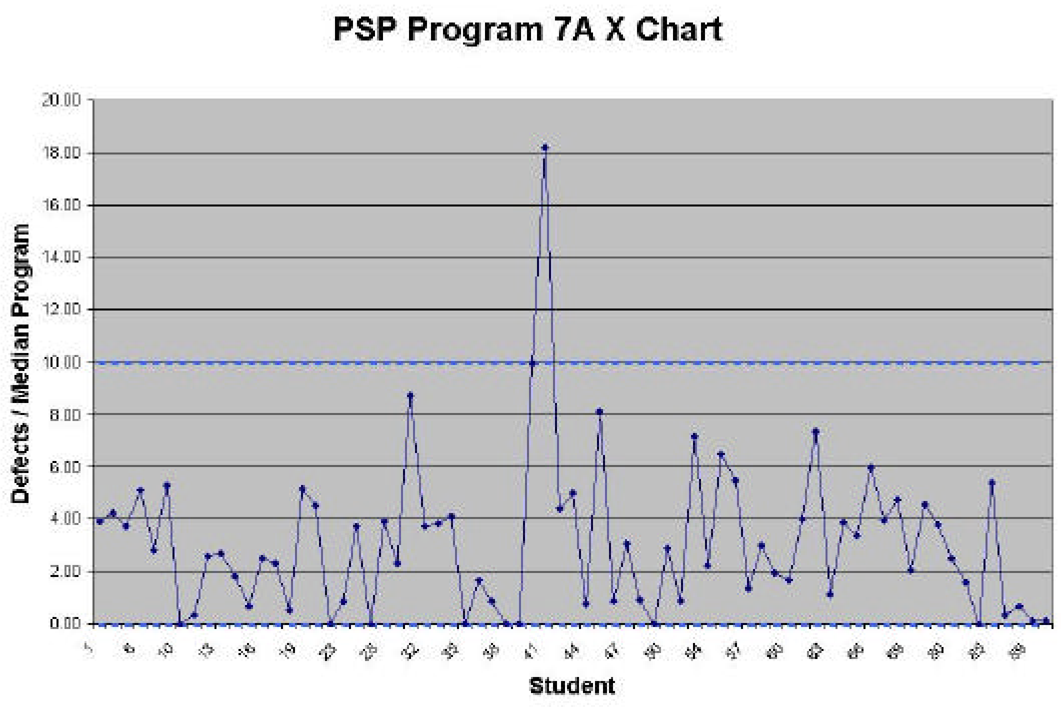
Two other confounding facts were considered: the programming language used and the experience of the programmer. Of the 298 students whose data is contained in the 1997 PSP data set, 71 usable sets of data for assignments programmed in C were identified, and that is the data set analyzed in this paper. Since experience does not seem to be a major factor for performance in the context of these small programming assignments (Hayes, 1997), experience of the programmer is not addressed in this analysis. This would not be recommended in general.

In the interests of brevity, only code inspections for program 7A are reported in this paper. Data were normalized by defect density rate times the median program size. XmR charts were calculated for code review rate and typical defect density; Table 1 summarizes the data.

Table 1. Summary of the PSP Data.

	Program 7A Review Rate **(LOC/min)**	**Program 7A Defect Density** **(defects/median program)**
n	37	62
X_{bar}	7.47	3.19
mR_{bar}	4.38	2.55
$UNLP_X$	19.12	9.98
$LNPL_X$	0	0
UCL_R	14.31	8.34

It should be noted that program 7A is when code and design reviews are first introduced in PSP, and it is apparent in the data that some students either did not perform the reviews or did not record the code review data. For program 7A, seven signals were iteratively identified in the code review rates for points 19, 23, 25, 43, 63, 69, and 88. Since these points suggest that an unstable inspection process was used, these points were discarded for the review rate data in Table 1 and the defect density analysis. Two signals were identified for the typical defect density at points 40 and 41. Graphically, this is captured in Figure 1. The X and mR charts for the defect density use all the data points, but the limits are calculated after the signals are removed. What the signal may indicate is unknown; it is generally recommended that a causal analysis be performed before deciding that a signal corresponds to an assignable cause, but that is impractical in this case.



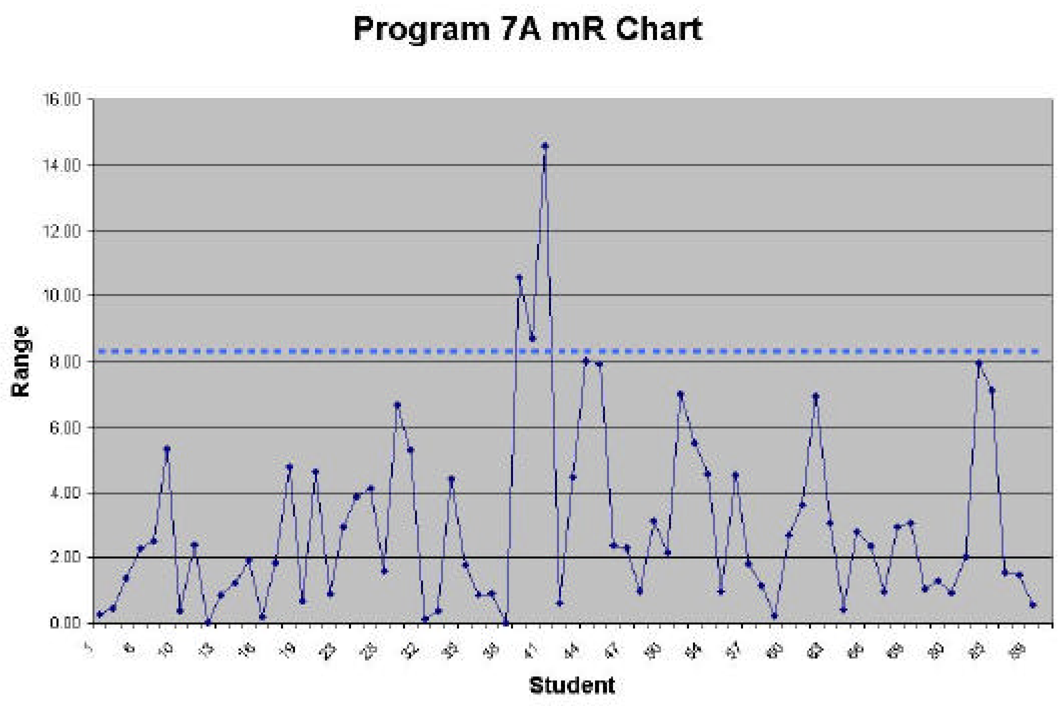


Figure 1. XmR Charts for PSP Program 7A Defect Density - Full Data Set.

Note that some points were identified as assignable causes based on the code review rate charts, which are not shown, but the defect density XmR chart illustrates both the use of the control chart and the fact that even in a PSP process there may be "signals" of extraordinary events -- especially when adopting a new technique, such as code reviews.

Conclusions

It may be unreasonable to expect that every major software process should be managed using control charts. There are other rigorous statistical techniques, such as confidence intervals and prediction intervals that could be superior in some situations. In some instances rigorous statistical techniques may not be economically justifiable. It is reasonable to expect, however, that SPC techniques will be in the tool kit of high maturity software organizations and used where they provide value.

The analysis reported here is only a brief introduction to the ongoing analyses of the PSP data that we hope to perform. The PSP and Team Software Process SM data provides insight into the potential performance of a disciplined process. Later analyses will examine the impact of experience, differing programming languages, and stability and predictability across assignments.

References

Robert D. Austin, **Measuring and Managing Performance in Organizations**, Dorset House Publishing, New York, NY, 1996.

Joe Besselman and Stan Rifkin, "Exploiting the Synergism Between Product Line Focus and Software Maturity," **Proceedings of the 1995 Acquisition Research Symposium**, Washington, D.C., pp. 95-107.

Anita D. Carleton, Mark C. Paulk, et al, "Panel Discussion: Can Statistical Process Control Be Usefully Applied to Software?" **The 11th Software Engineering Process Group (SEPG) Conferenc**e, Atlanta, Georgia, 8-11 March 1999 and **European SEGP 99**, 7-10 June 1999, Amsterdam, Netherlands. See http://www.sei.cmu.edu/cmm/slides/slides.html#spc-panel.

Bill Curtis, "Managing the Real Leverage in Software Productivity and Quality," American Programmer, Vol. 3, No. 7, July/August 1990, pp. 4-14.

Tom DeMarco and Timothy Lister, **Peopleware, 2nd Edition**, Dorset House, New York, NY, 1999.

W. Edwards Deming, **Out of the Crisis**, MIT Center for Advanced Engineering Study, Cambridge, MA, 1986.

William A. Florac and Anita D. Carleton, **Measuring the Software Process: Statistical Process Control for Software Process Improvement**, Addison-Wesley, Reading, MA, 1999.

William A. Florac, Anita D. Carleton, and Julie Barnard, "Statistical Process Control: Analyzing a Space Shuttle Onboard Software Process," IEEE Software, Vol. 17, No. 4, July/August 2000, pp. 97-106.

Lynne B. Hare, Roger W. Hoerl, John D. Hromi, and Ronald D. Snee, "The Role of Statistical Thinking in Management," ASQC Quality Progress, Vol. 28, No. 2, February 1995, pp. 53-60.

Will Hayes and James W. Over, "The Personal Software Process (PSP): An Empirical Study of the Impact of PSP on Individual Engineers," Software Engineering Institute, Carnegie Mellon University, CMU/SEI-97-TR-001, December 1997.

K. Ishikawa, **Guide to Quality Control**, Asian Productivity Organization, Tokyo, Japan, (available from Unipub - Kraus International Publications, White Plains, NY) 1986.

Stephen H. Kan, **Metrics and Models in Software Quality Engineering**, Addison-Wesley, Reading, MA, February 1995.

M.A. Ould, "CMM and ISO 9001," Software Process: Improvement and Practice, Vol. 2, Issue 4, December 1996, pp.281-289.

Mark C. Paulk, "Toward Quantitative Process Management With Exploratory Data Analysis," **Proceedings of the Ninth International Conference on Software Quality**, Cambridge, MA, 4-6 Oct 1999, pp. 35-42.

Mark C. Paulk, Dennis Goldenson, and David M. White, "The 1999 Survey of High Maturity Organizations," Software Engineering Institute, Carnegie Mellon University, CMU/SEI-2000-SR-002, February 2000.

Thomas Pyzdek, "Process Control for Short and Small Runs," ASQC Quality Progress, Vol. 26, No. 4, April 1993, pp. 51-60.

Walter A. Shewhart, **Statistical Method from the Viewpoint of Quality Control**, Dover Publications, Mineola, NY, 1939, republished 1986.

Edward F. Weller, “Practical Applications of Statistical Process Control,” IEEE Software, Vol. 17, No. 3, May/June 2000, pp. 48-55.

Donald J. Wheeler and David S. Chambers, **Understanding Statistical Process Control, Second Edition**, SPC Press, Knoxville, TN, 1992.

Donald J. Wheeler and Sheila R. Poling, **Building Continual Improvement: A Guide for Business**, SPC Press, Knoxville, TN, 1998.

Applying SPC to the Personal Software Process^SM

Mark C. Paulk

mcp@sei.cmu.edu -or- Mark.Paulk@ieee.org
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Statistical Process Control

The use of *statistical* tools and techniques

to analyze a *process*
or its *outputs*

to control, manage, and improve
the *quality* of the output
or the *capability* of the process.

SPC for Software Premises

The software process is performed by people, not machines.

The software process is (or can be) repeatable, but not repetitive.

The act of measuring and analyzing will change behavior – potentially in dysfunctional ways.

Seven Basic SPC Tools

Scatter diagrams

Run charts

Cause-and-effect diagrams

Histograms

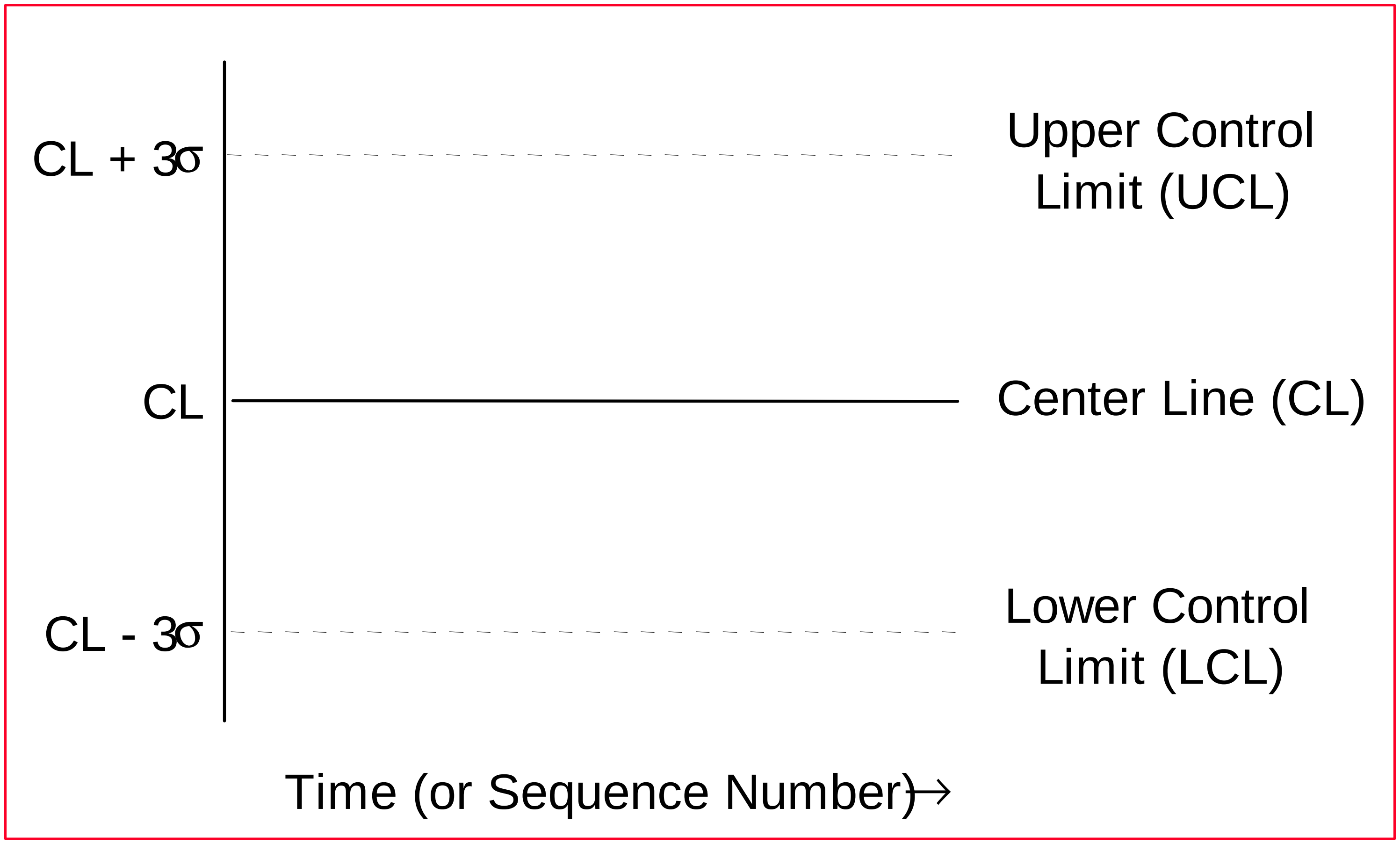
Bar charts

Pareto charts

Control charts

"SPC" implies control charts!

Control Chart Basics


CL + 3σ
CL
CL - 3σ
Upper Control Limit (UCL)
Center Line (CL)
Lower Control Limit (LCL)
Time (or Sequence Number) →

Process Variation

Shewhart's notion of dividing variation into two types:

- ***common cause*** **variation**
  - **variation in process performance due to normal or inherent interaction among process components (people, machines, material, environment, and methods).**
- ***assignable cause*** **variation**
  - **variation in process performance due to events that are not part of the normal process.**
  - **represents sudden or persistent abnormal changes to one or more of the process components.**

Equation Form

total variation

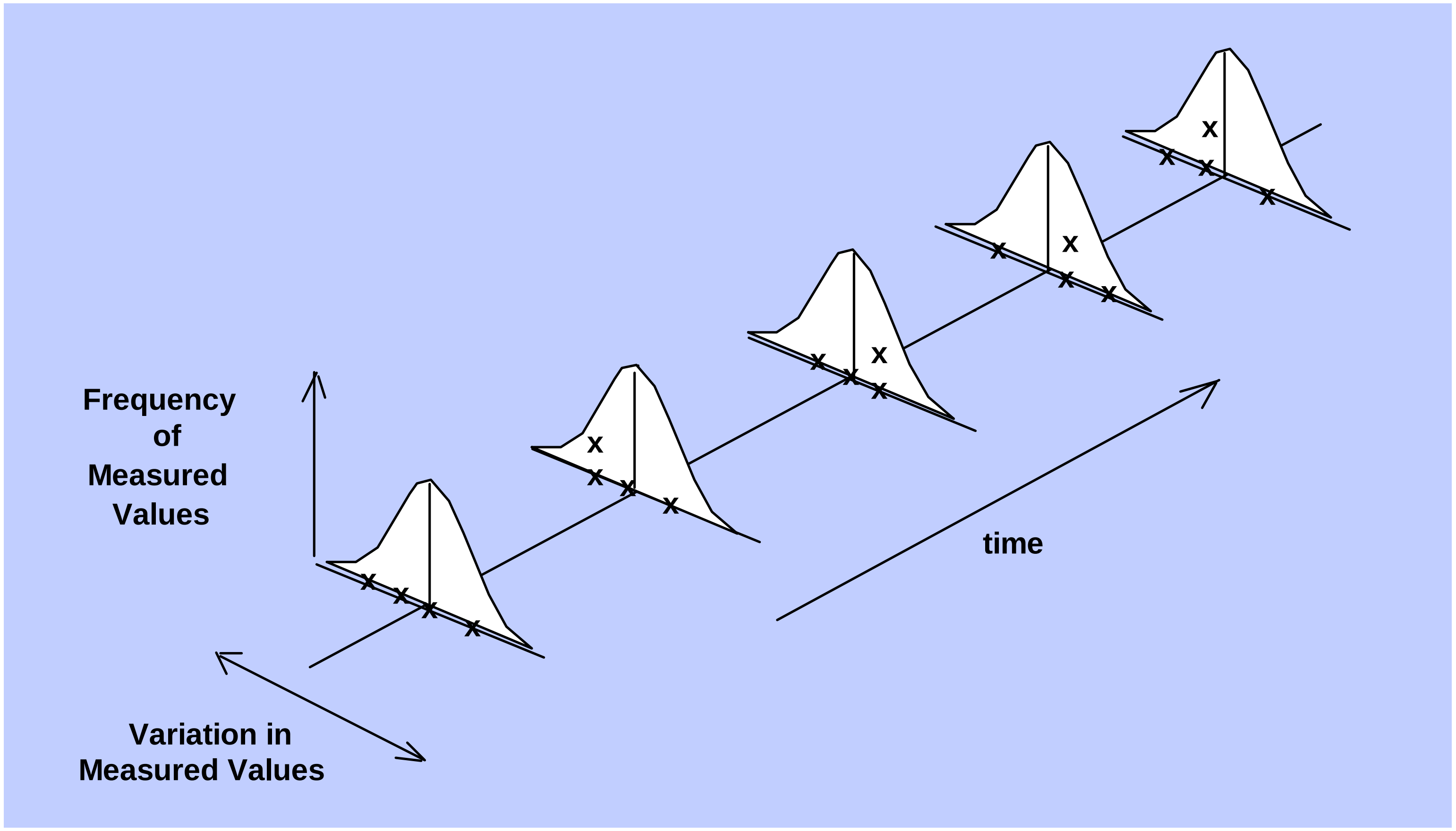
=

common cause variation

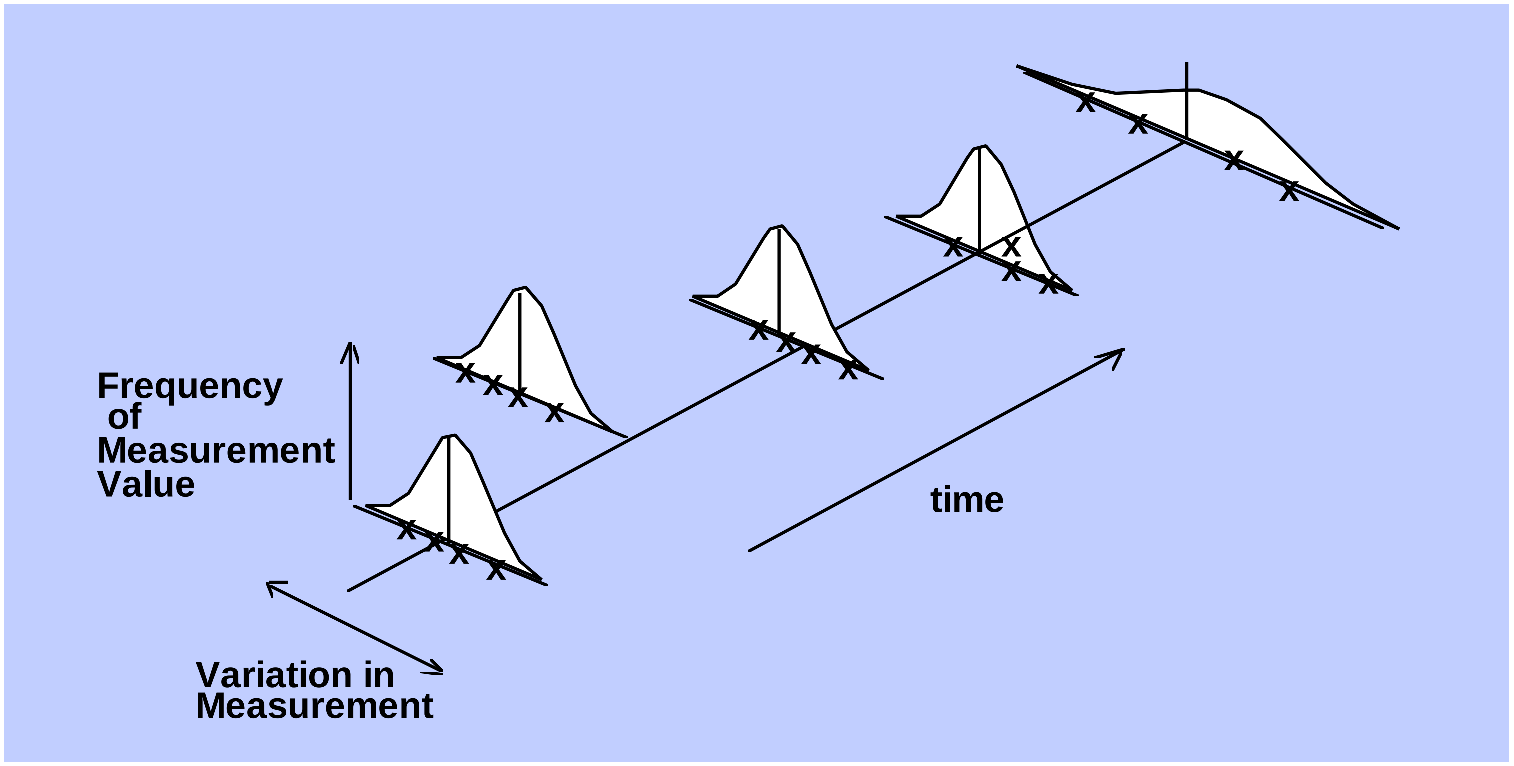
+

assignable cause variation

An Example Process in Statistical Control



An Example Out-of-Control Process



Useful Control Charts

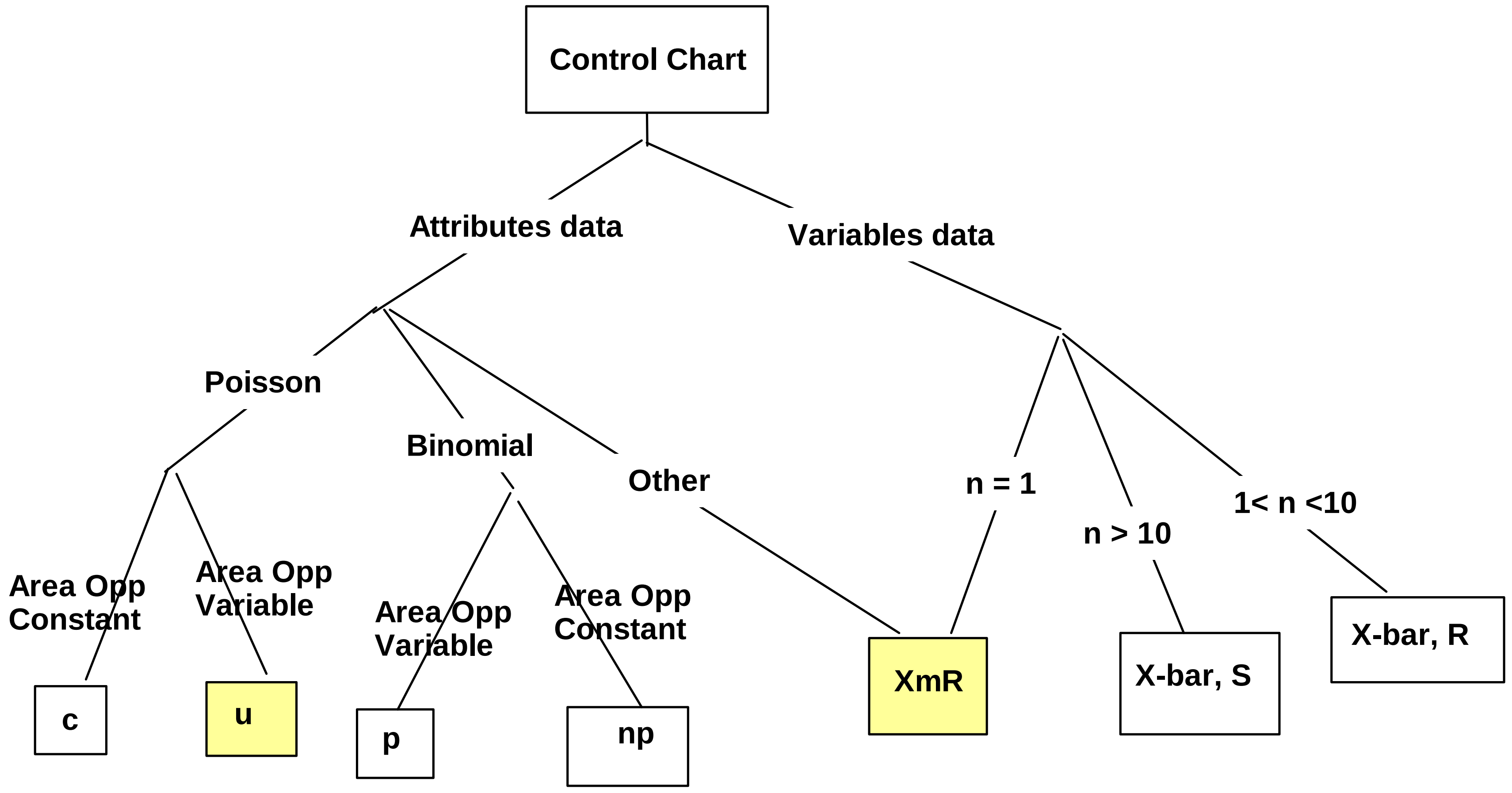
Hypothesis: begin with control charts for defect data.

Most likely to be of value for software processes

- **XmR chart for attributes data**
- **u-chart**
- **Z-chart**

Wheeler: XbarR and XmR charts should satisfy 99% or more of your control charting needs.

What Control Chart Should be Used?



Assumptions

Variable vs attribute

- **usually known, usually (for most software processes) attribute**
  - **defects, rates, etc.**

Statistical distributions, e.g., normal, Poisson

- **if a distributional assumption is made, it is wise to check whether the assumption is plausible**

Area of opportunity variable or constant

- **typically size measure (SLOC or FP), usually variable**

XmR Chart

Assumes population standard deviation (sigma) is constant across all observations

- **measuring a common-cause system!**

Does not assume a particular statistical distribution, e.g., Poisson or binomial data

Weakness: "chunky data"

- **range with less than four values**
- **average count less than 1**

XmR Chart Formulas

mR = moving Range
(difference between successive data points)

$UNPL_X = \overline{X} + 2.66\ \overline{(mR)}$

$CL_X = \overline{X}$

$LNPL_X = \overline{X} - 2.66\ \overline{(mR)}$

$URL_{mR} = 3.27\ \overline{(mR)}$

$CL_{mR} = \overline{mR}$

Notation: bar over a variable indicates "average" -- as does the word "bar," e.g., X_{bar}

Characterizing the PSP Data

Analyze the PSP 7A, 8A, 9A, and 10A programs

- **fairly stable, well-defined processes**
- **process still being refined in class (PSP2, PSP2.1, and PSP3)**
- **design and code reviews are introduced with PSP 7A**

Salient features of the data set

- **same functionality being implemented**
- **same disciplined process across individuals**
- **each program produced and reviewed by a different individual**
  - **focus on differences between individuals with a disciplined process**

Analyses Performed

Possible confounding factors

- **programming language**
- **experience of the student**

Use XmR charts to analyze process stability

- **design effort**
- **design review rate**
- **design review defects**
- **coding effort**
- **code review rate**
- **code review defects**
- **defect removal efficiency before test**

Issues

Separate analysis of

- **C programmers *versus* all programming languages**
- **experienced (5 or more years) programmers *versus* all programmers**

Reviews are introduced in PSP 7A

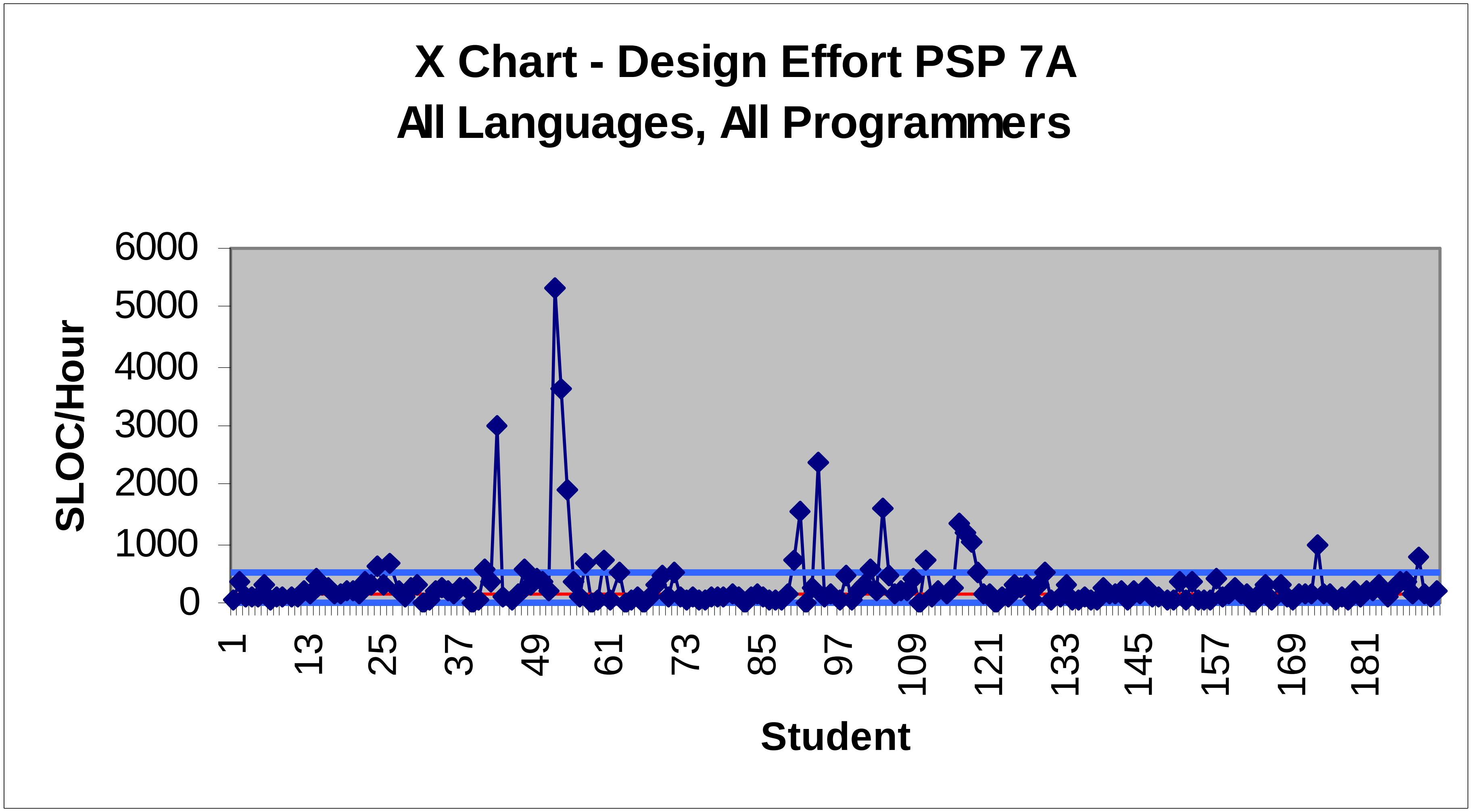
- **some programmers spent zero minutes in design and/or code reviews**

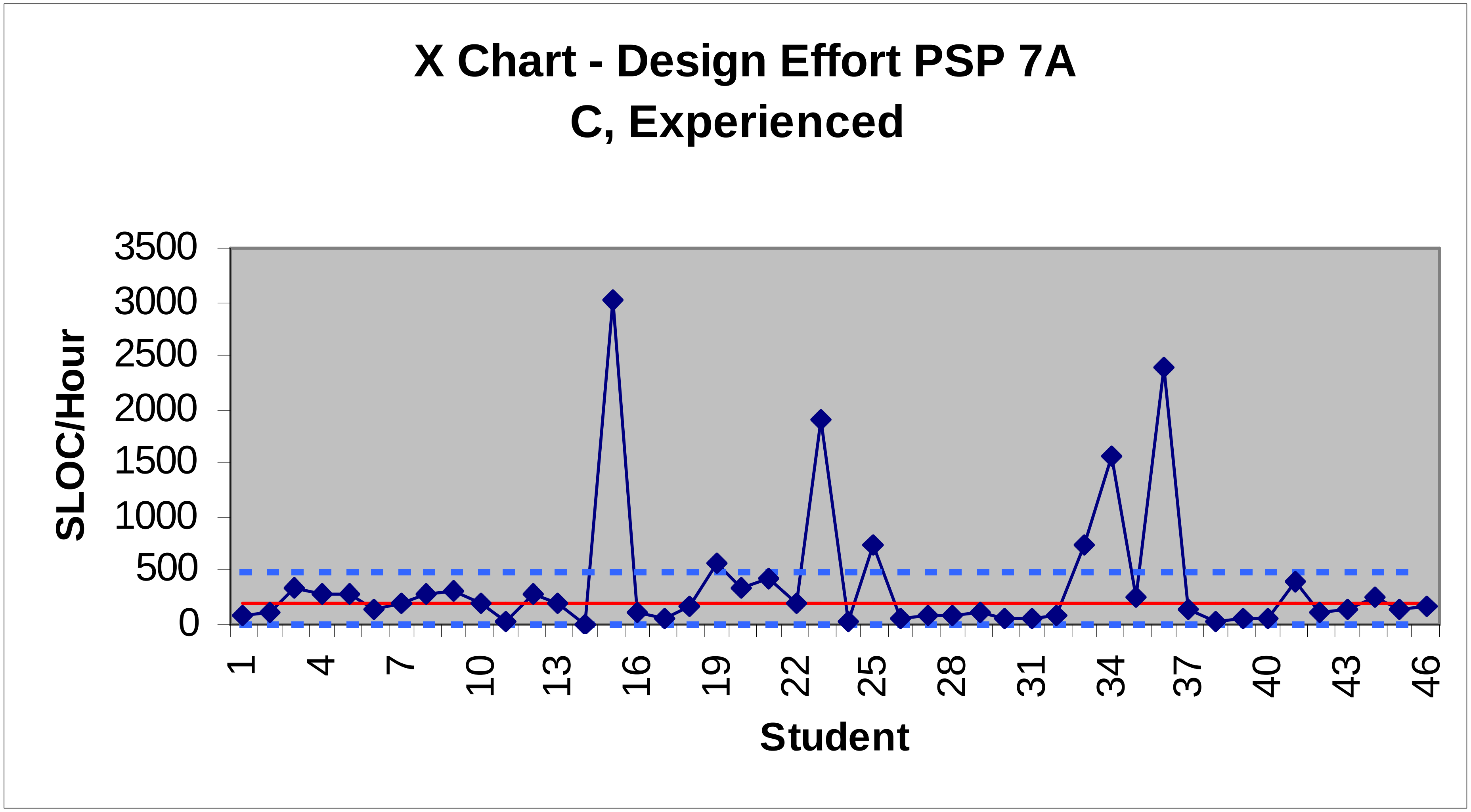
Control charts of percentages may have bounds above 100% or below 0% …

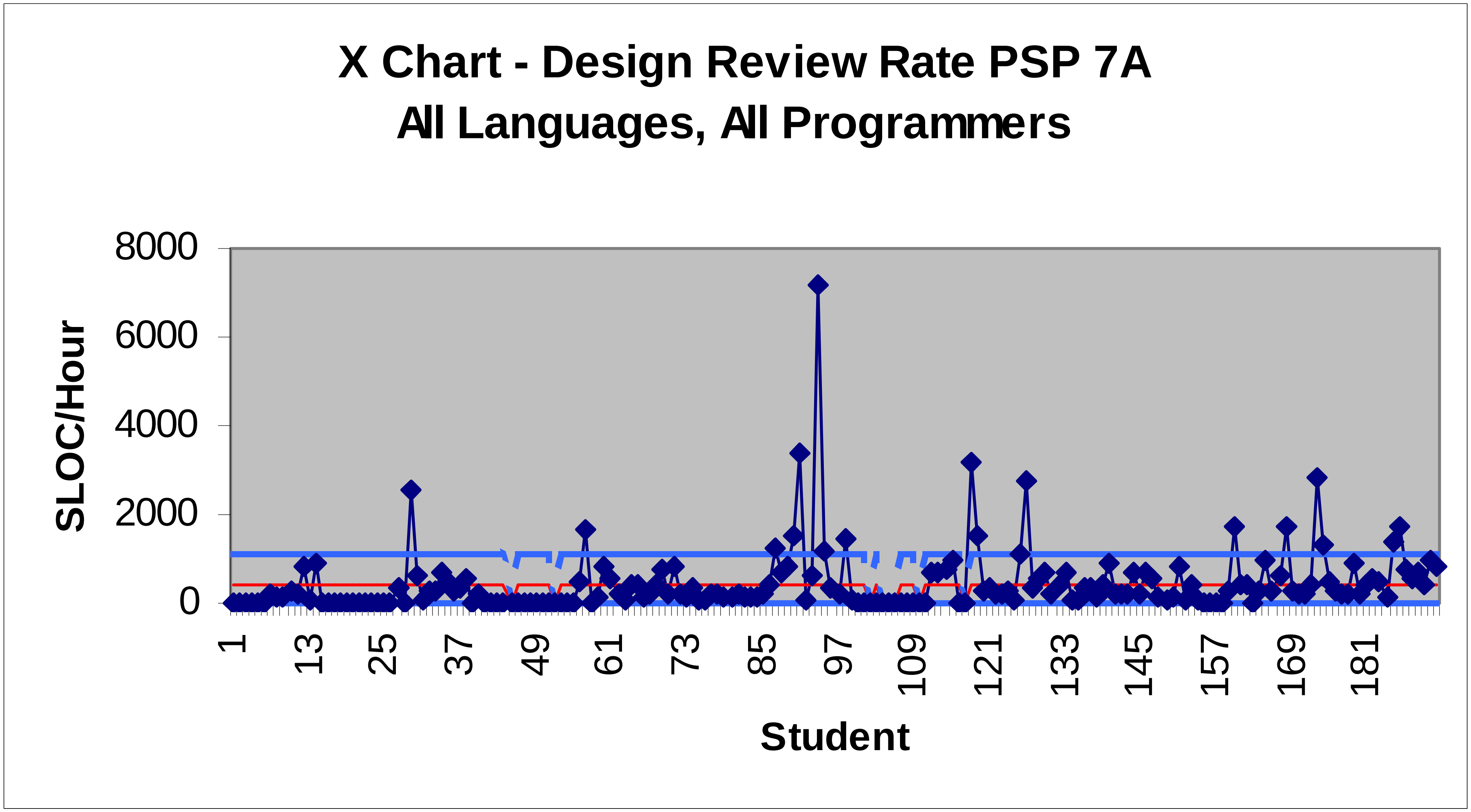
Iterative removal of “assignable causes”

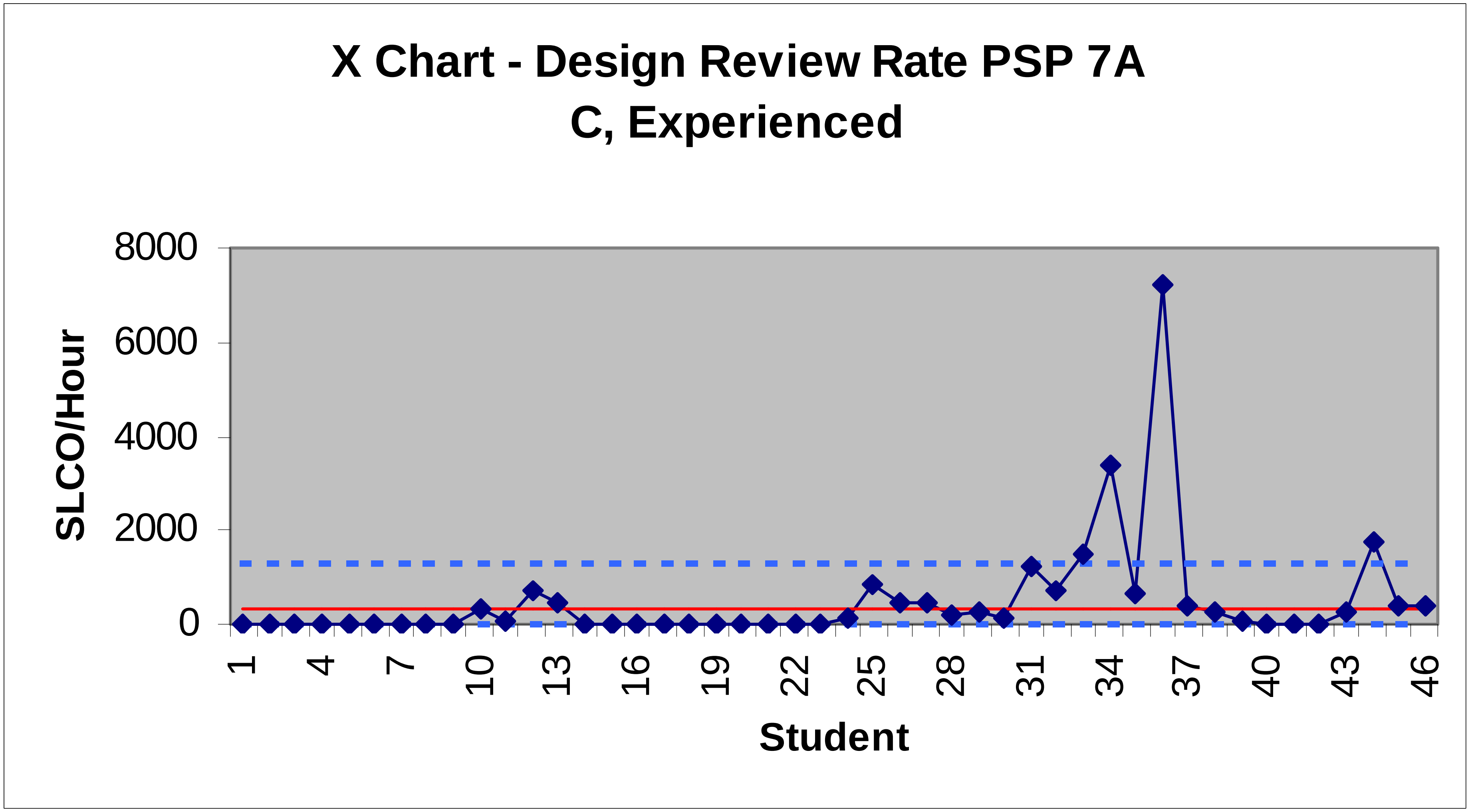
PSP 7A

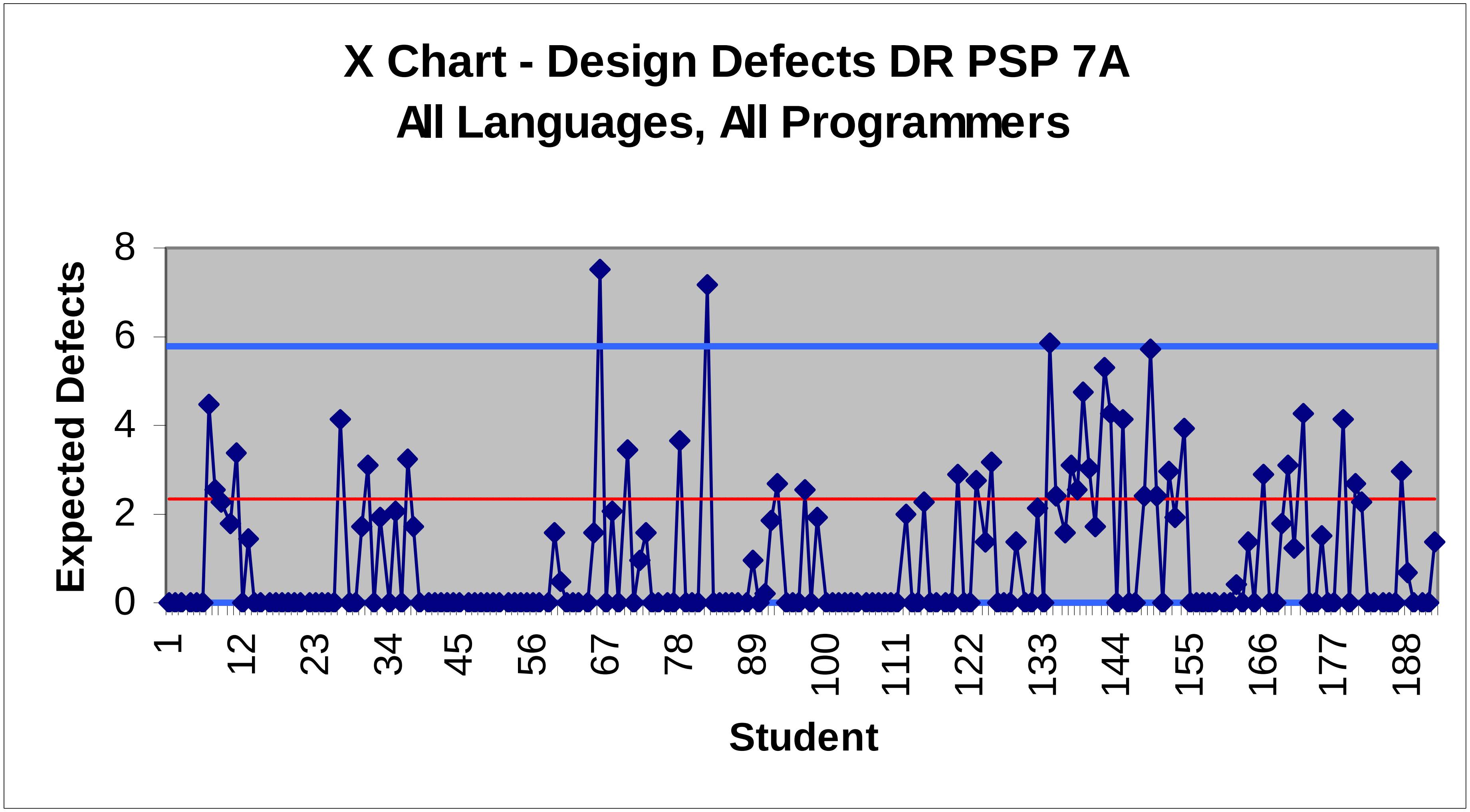
Process	**PSP2**
Characteristic	**Code and design reviews**
Population	**All programming languages** **All programmers** ***Experienced C programmers***

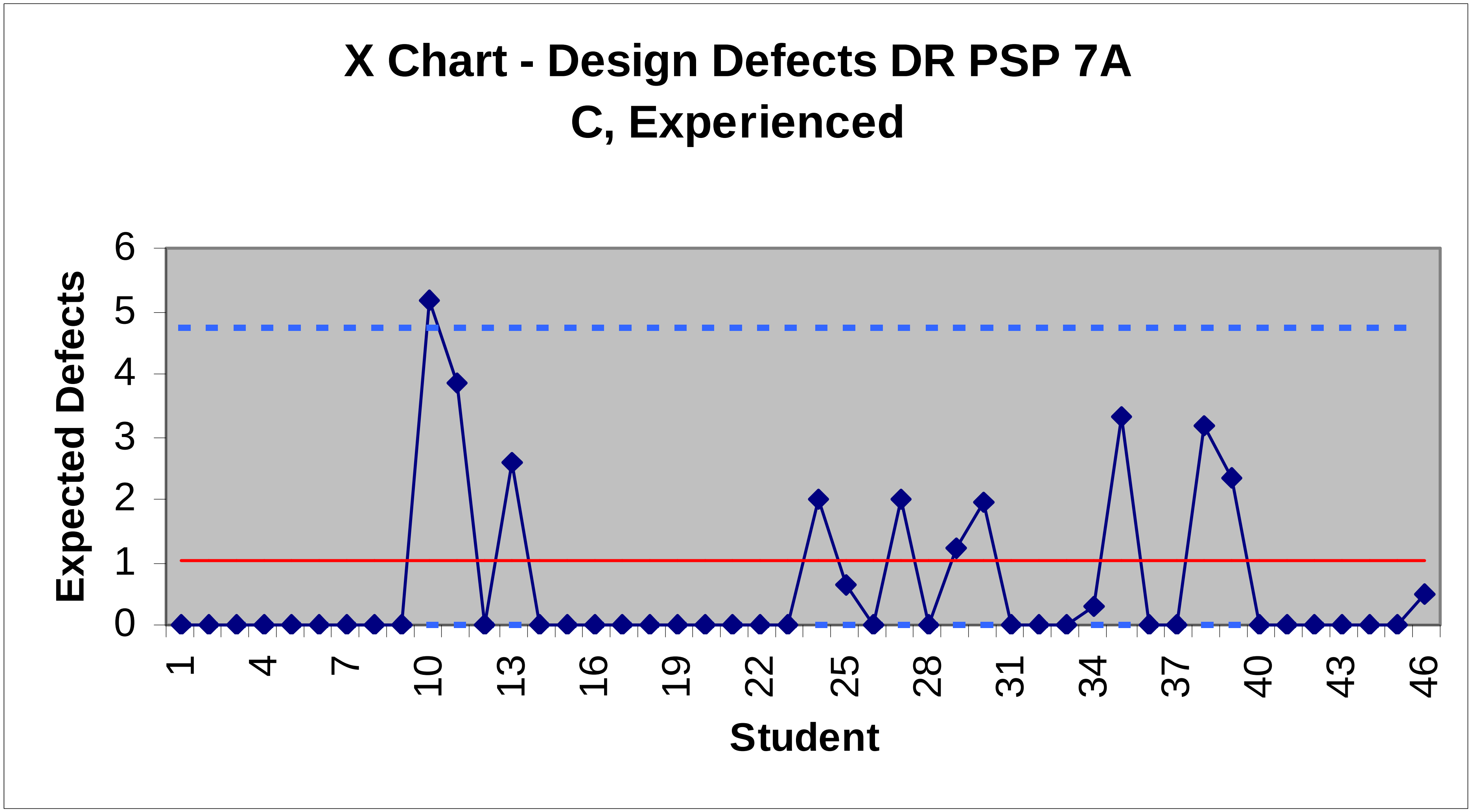

X Chart - Design Effort PSP 7A
All Languages, All Programmers
SLOC/Hour
6000
5000
4000
3000
2000
1000
0
1
13
25
37
49
61
73
85
97
109
121
133
145
157
169
181
Student

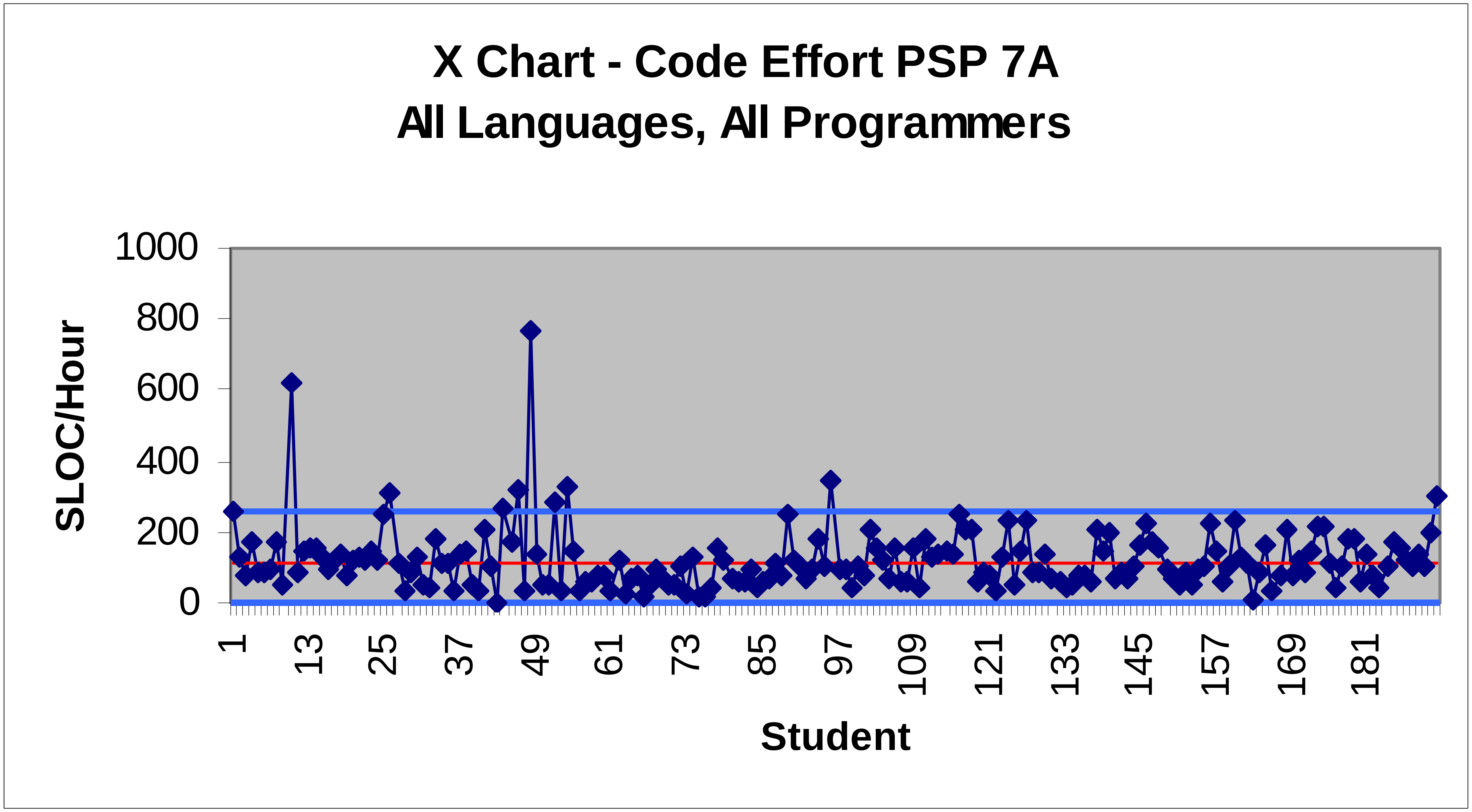

X Chart - Design Effort PSP 7A
C, Experienced
SLOC/Hour
3500
3000
2500
2000
1500
1000
500
0
1
4
7
10
13
16
19
22
25
28
31
34
37
40
43
46
Student


X Chart - Design Review Rate PSP 7A
All Languages, All Programmers
8000
6000
4000
2000
0
SLOC/Hour
1
13
25
37
49
61
73
85
97
109
121
133
145
157
169
181
Student

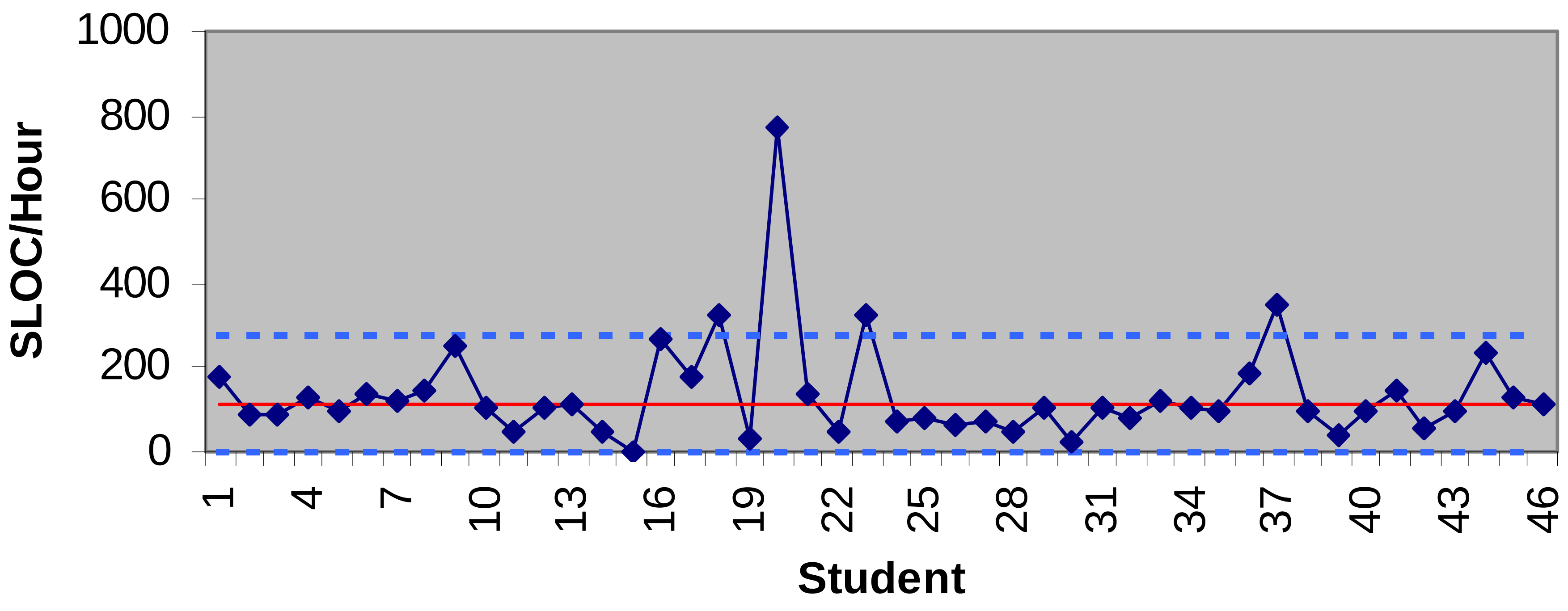

X Chart - Design Review Rate PSP 7A
C, Experienced
SLCO/Hour
8000
6000
4000
2000
0
1
4
7
10
13
16
19
22
25
28
31
34
37
40
43
46
Student

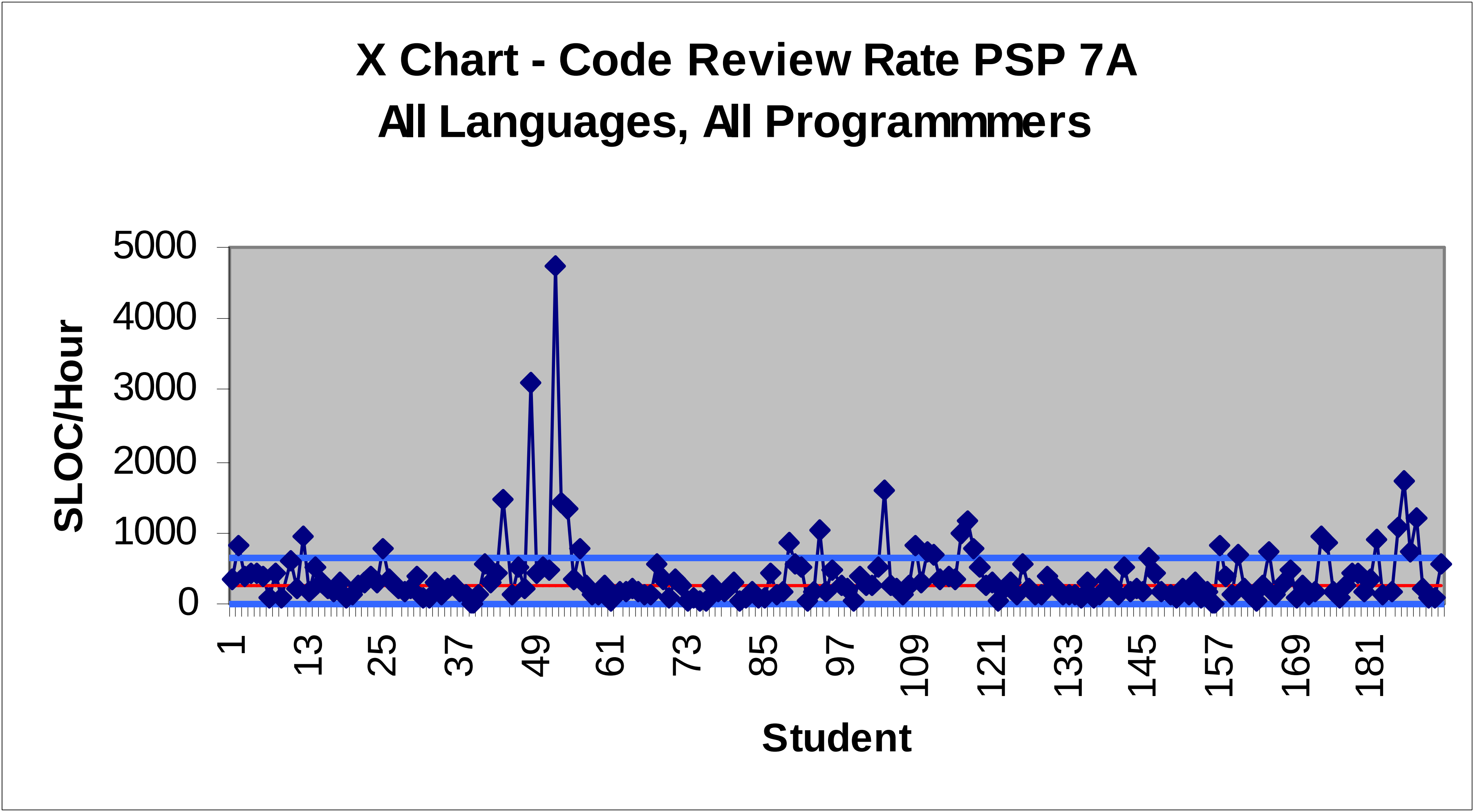

X Chart - Design Defects DR PSP 7A
All Languages, All Programmers
Expected Defects
8
6
4
2
0
Student
1
12
23
34
45
56
67
78
89
100
111
122
133
144
155
166
177
188


X Chart - Design Defects DR PSP 7A
C, Experienced
Expected Defects
0
1
2
3
4
5
6
1
4
7
10
13
16
19
22
25
28
31
34
37
40
43
46
Student

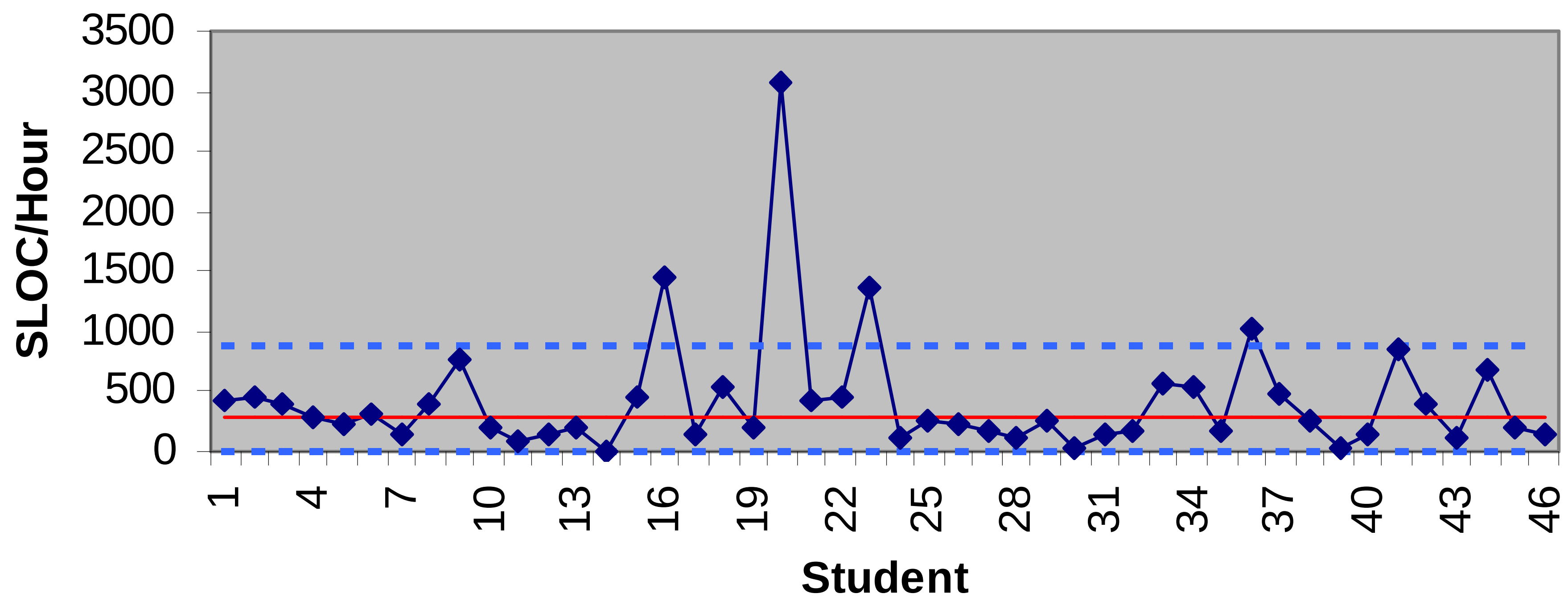

X Chart - Code Effort PSP 7A
All Languages, All Programmers
SLOC/Hour
1000
800
600
400
200
0
1
13
25
37
49
61
73
85
97
109
121
133
145
157
169
181
Student

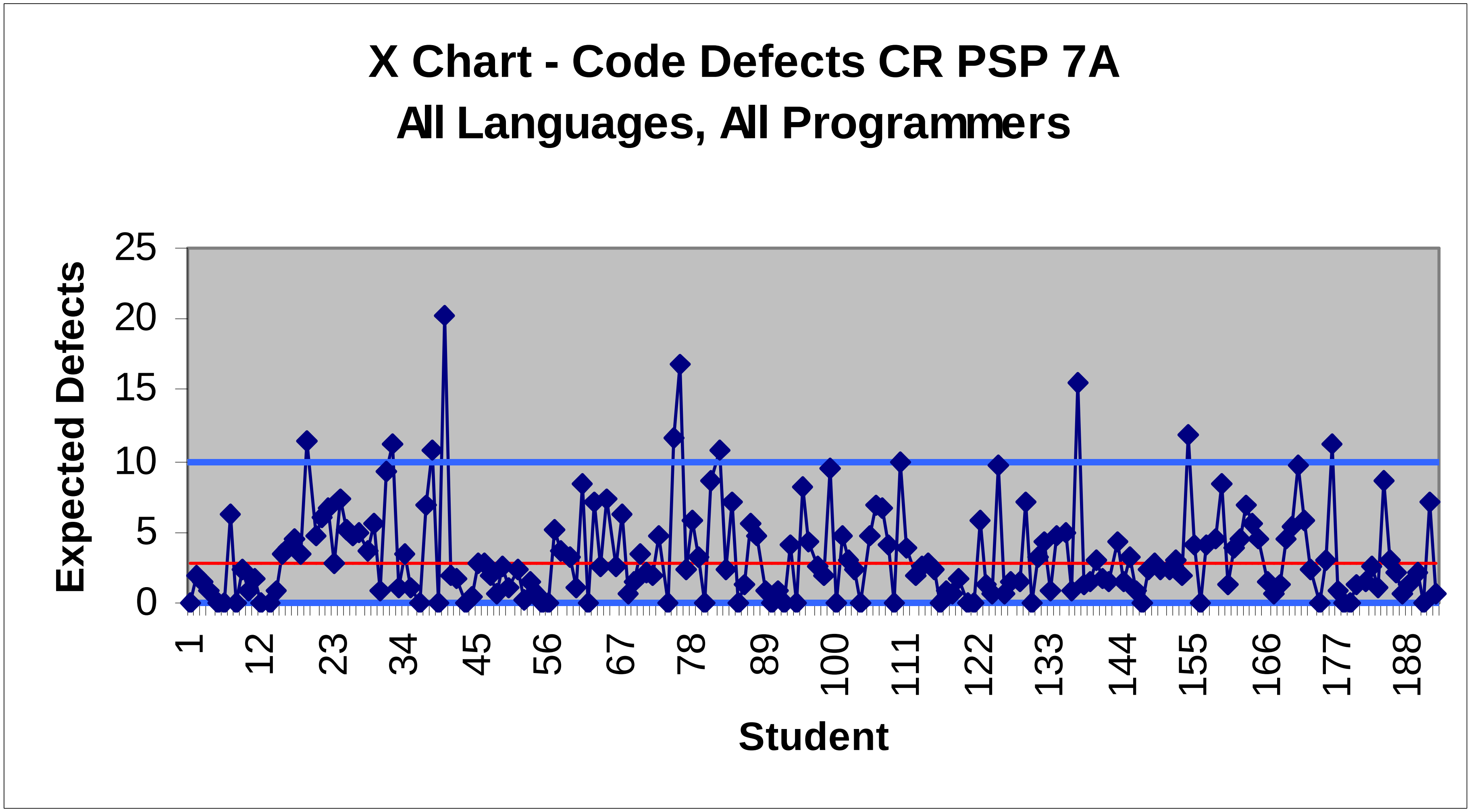
X Chart - Code Effort PSP 7A C, Experienced



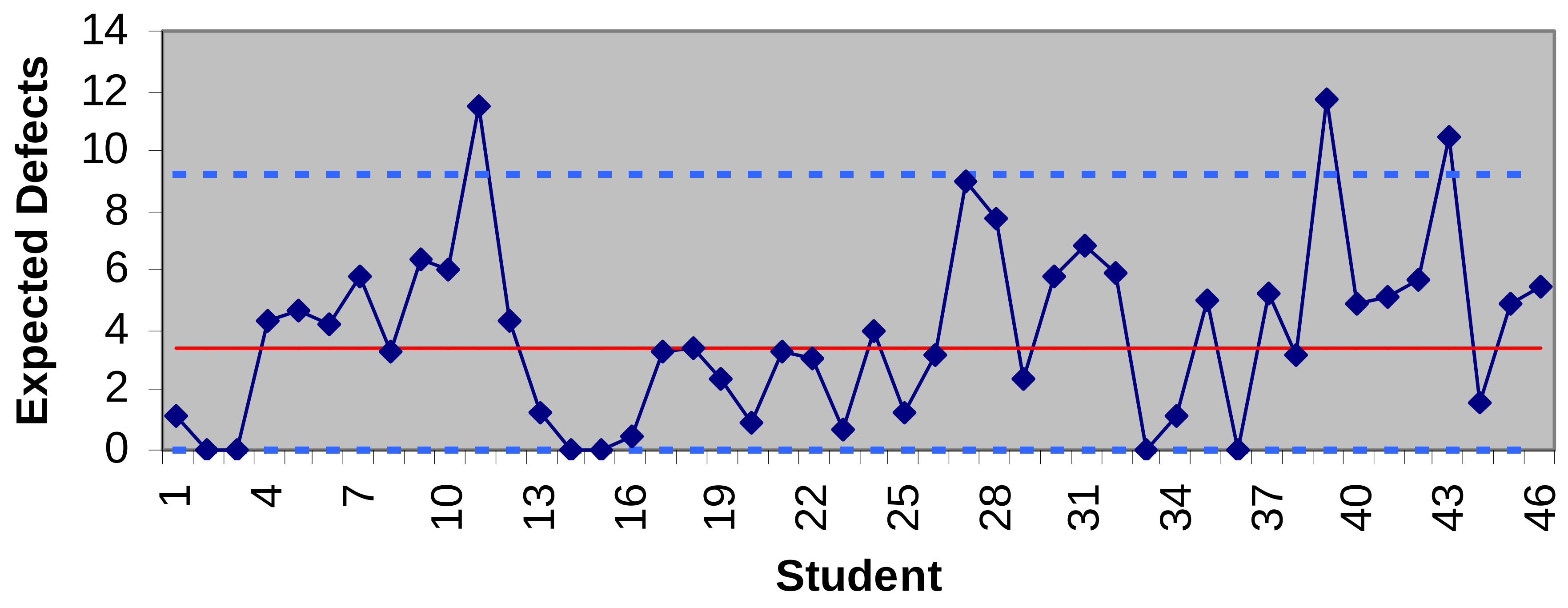

X Chart - Code Review Rate PSP 7A
All Languages, All Programmmers
SLOC/Hour
5000
4000
3000
2000
1000
0
1
13
25
37
49
61
73
85
97
109
121
133
145
157
169
181
Student

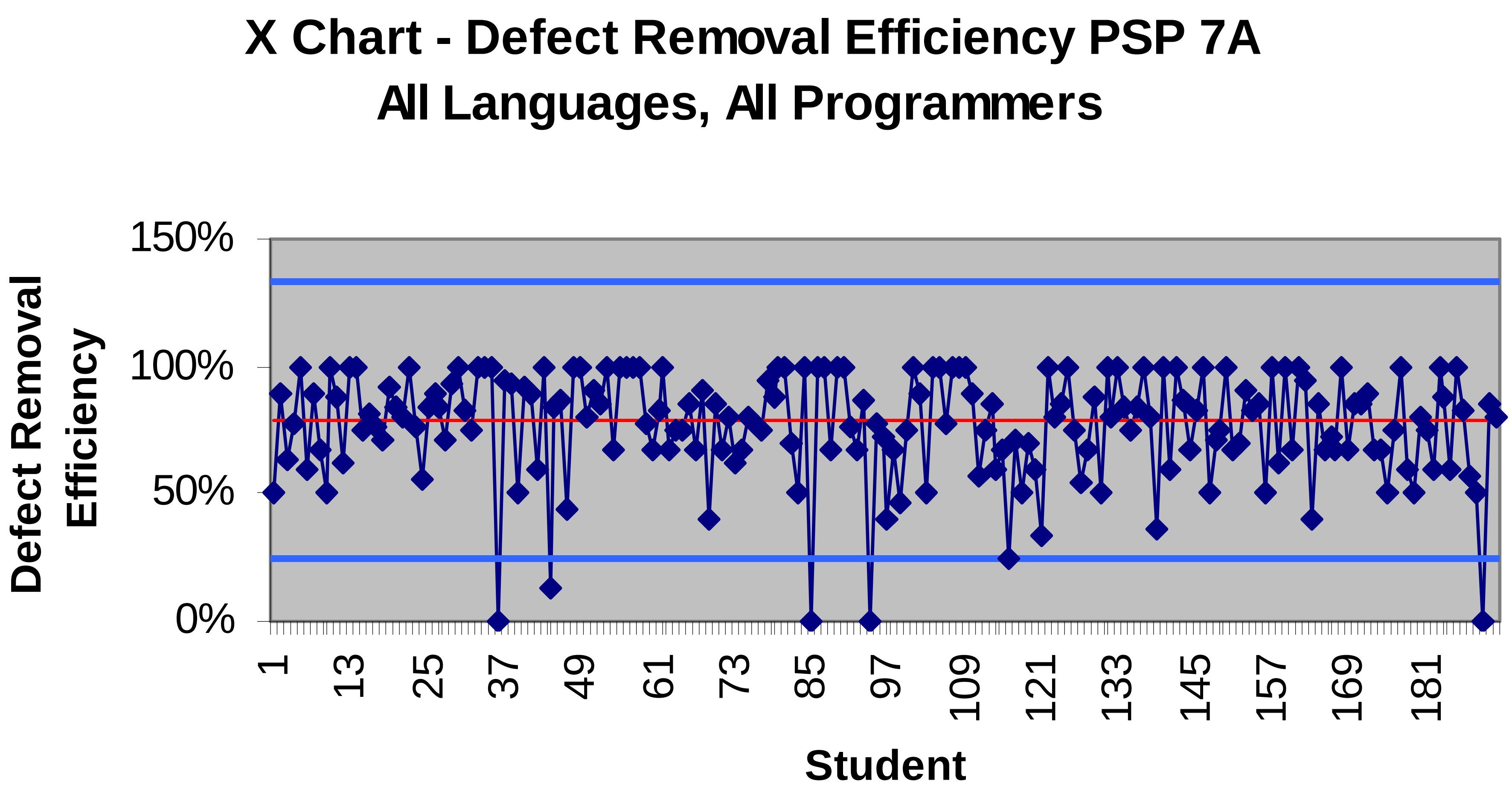
X Chart - Code Review Rate PSP 7A C, Experienced

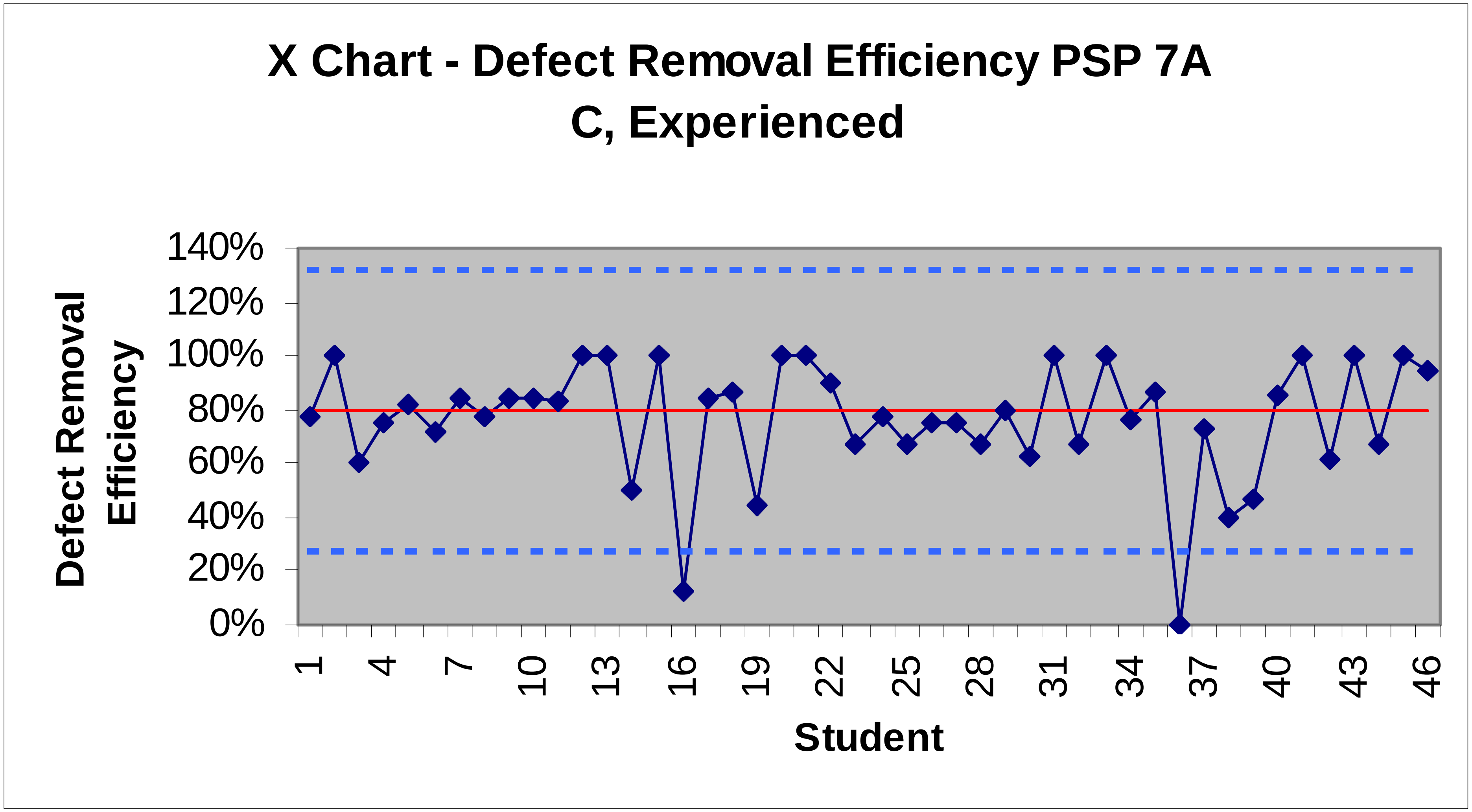



X Chart - Code Defects CR PSP 7A
All Languages, All Programmers
Expected Defects
25
20
15
10
5
0
1
12
23
34
45
56
67
78
89
100
111
122
133
144
155
166
177
188
Student

X Chart - Code Defects CR PSP 7A C, Experienced




X Chart - Defect Removal Efficiency PSP 7A
All Languages, All Programmers
Defect Removal Efficiency
150%
100%
50%
0%
1
13
25
37
49
61
73
85
97
109
121
133
145
157
169
181
Student


X Chart - Defect Removal Efficiency PSP 7A
C, Experienced
Defect Removal Efficiency
140%
120%
100%
80%
60%
40%
20%
0%
1
4
7
10
13
16
19
22
25
28
31
34
37
40
43
46
Student

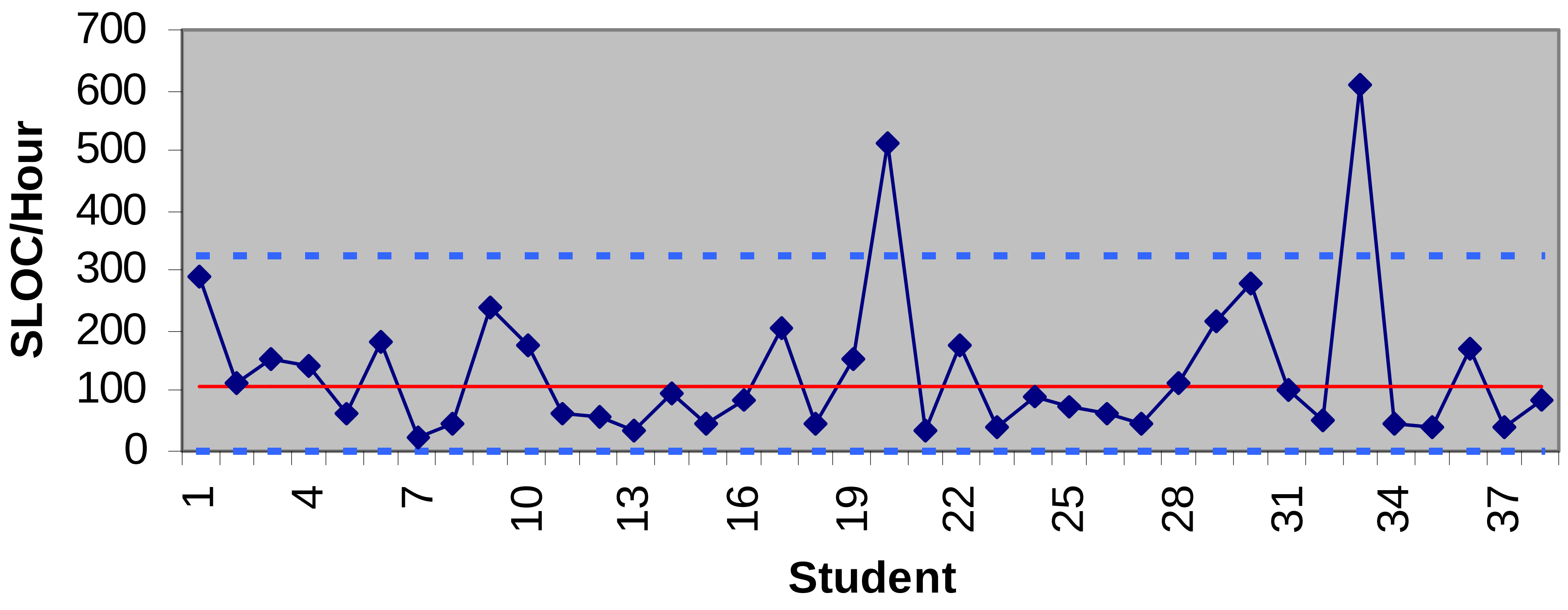
PSP 7A -- Summary of Results

Variable	**LNPL$_x$**	**X$_{bar}$**	**UNPL$_x$**	**R$_{bar}$**	**UCL$_R$**
Design effort	0 *0*	175 *203*	521 *470*	130 *100*	555 *428*
Design review rate	0 *0*	382 *333*	1132 *1279*	282 *356*	1204 *1518*
Design review defects	0 *0*	2.4 *1.0*	5.8 *4.7*	1.3 *1.4*	5.4 *5.9*
Coding effort	0 *0*	109 *110*	262 *273*	58 *61*	245 *262*
Code review rate	0 *0*	242 *298*	647 *884*	152 *220*	649 *941*
Code review defects	0 *0*	2.8 *3.4*	9.9 *9.2*	2.7 *2.2*	11.4 *9.3*
Defect removal efficiency	25% *27%*	79% *80%*	134% *132%*	20% *20%*	87% *85%*

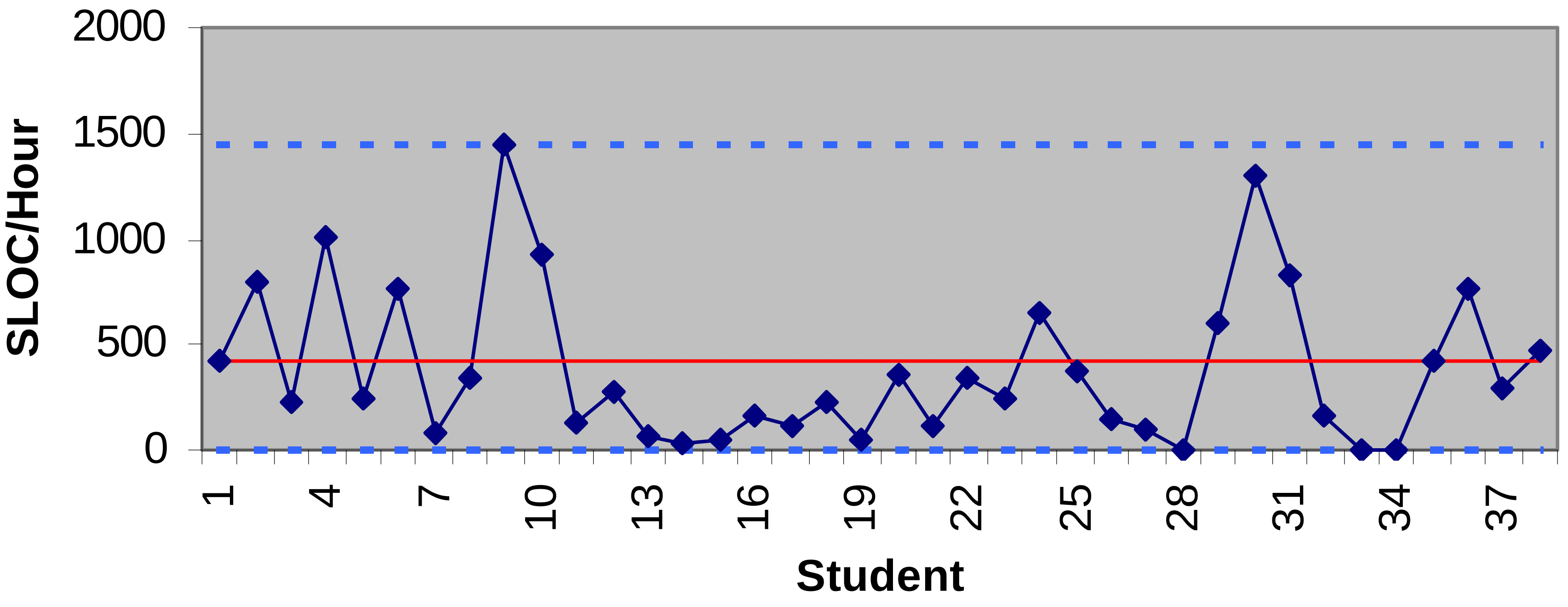
PSP 10A

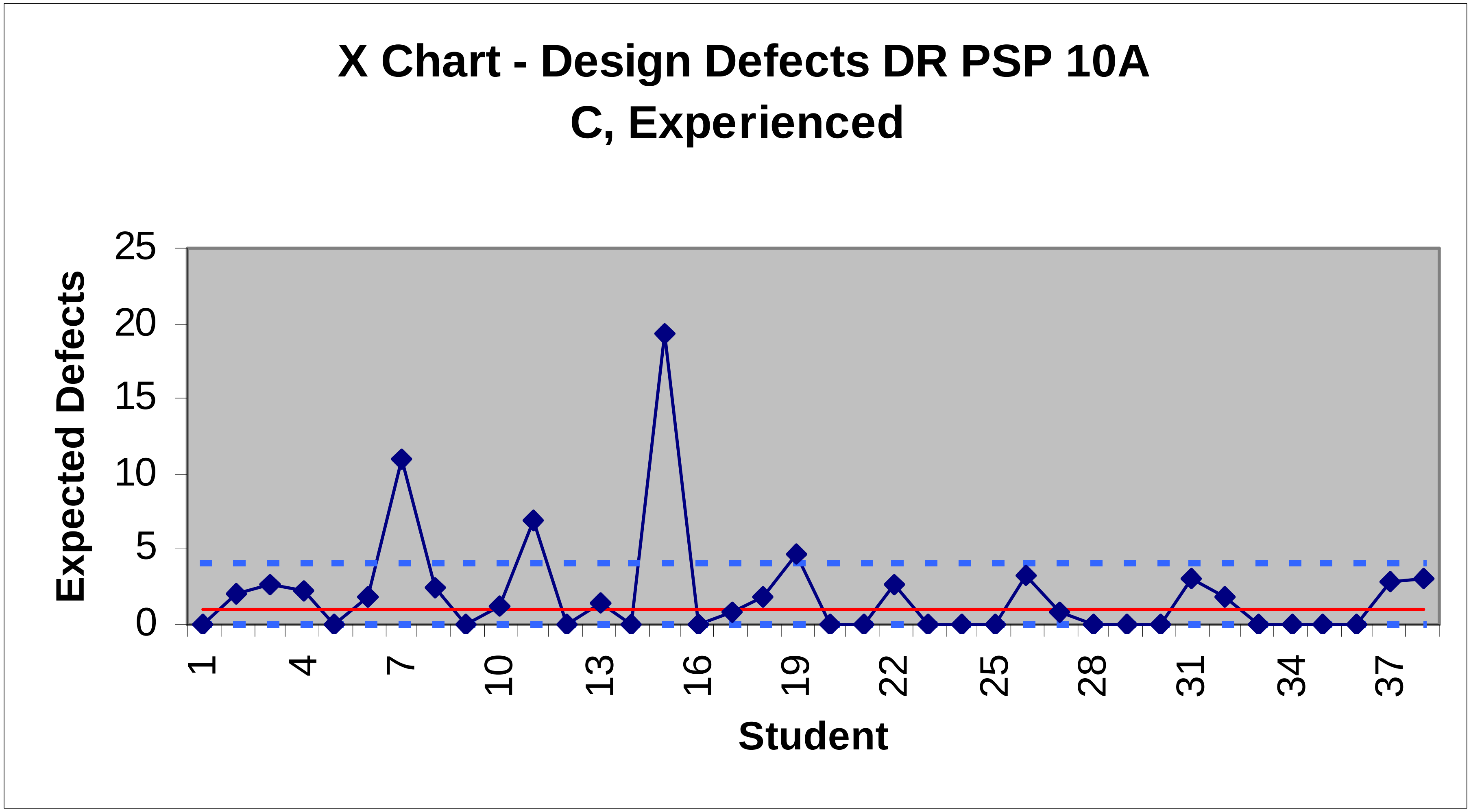
Process	**PSP3**
Characteristic	**Design and code reviews (2)** **Design templates (2.1)** **Cyclic development**
Population	***Experienced C programmers***

X Chart - Design Effort PSP 10A C, Experienced

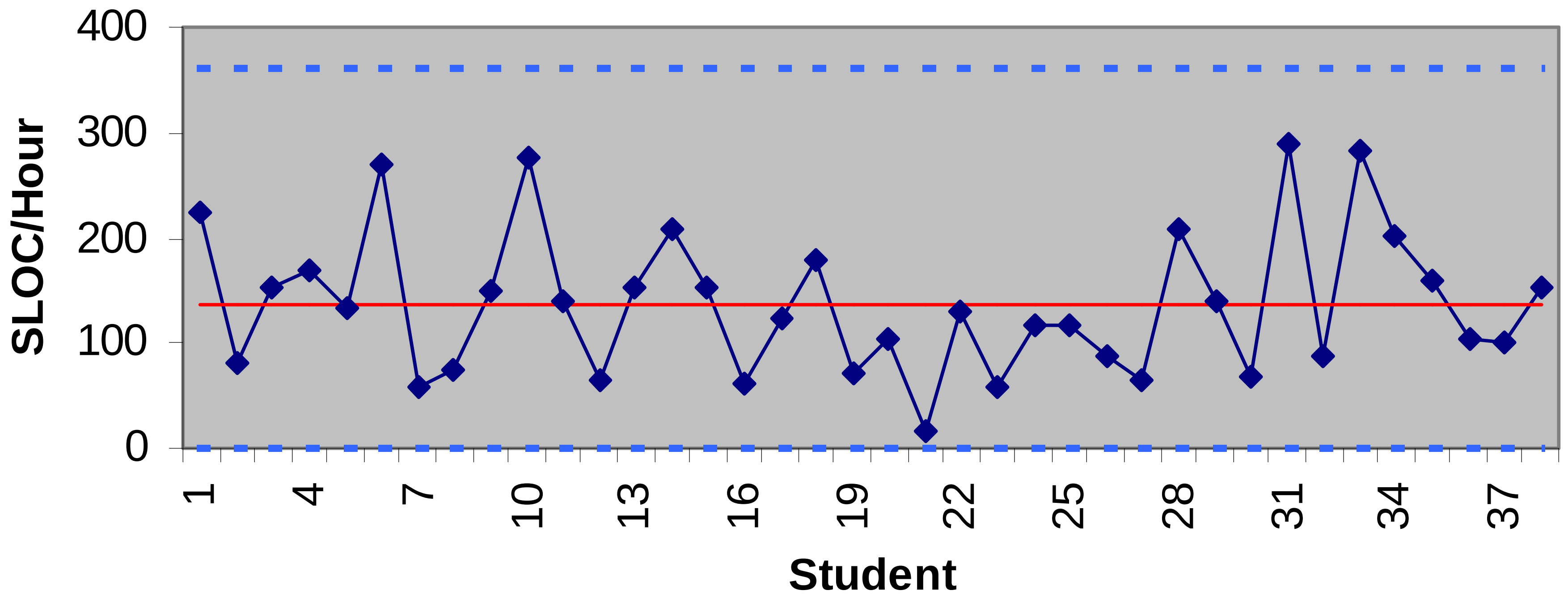


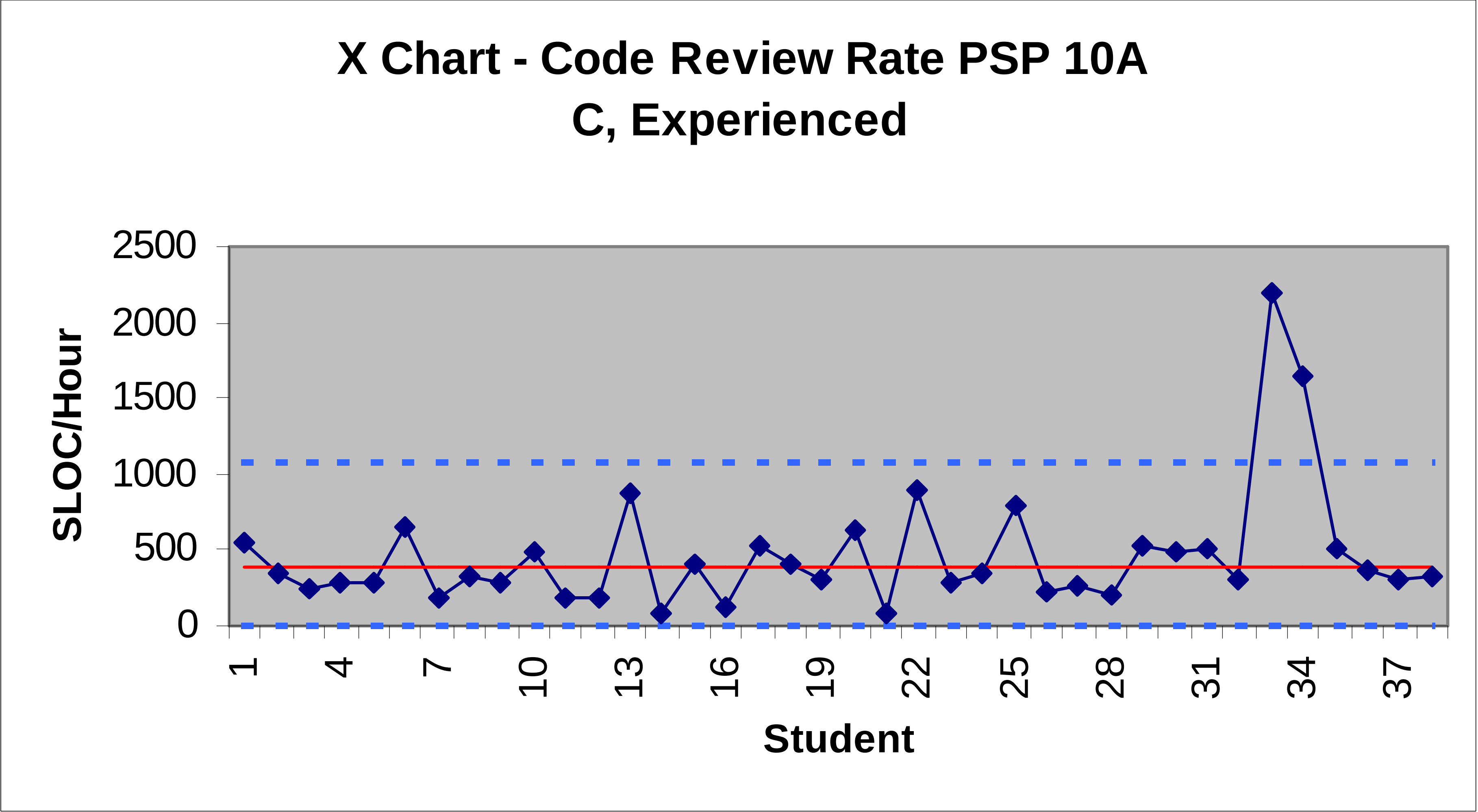
X Chart - Design Review Rate PSP 10A C, Experienced

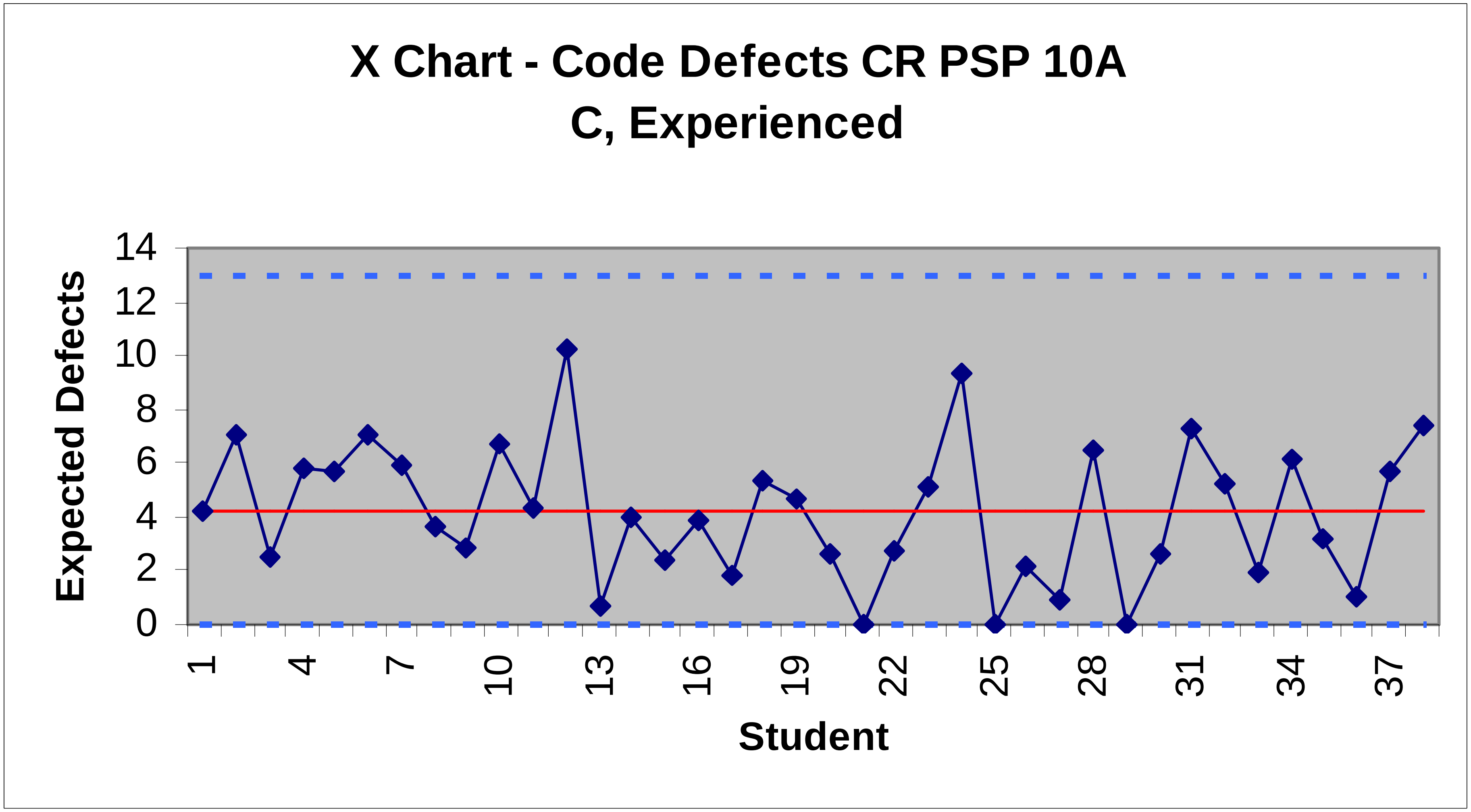



X Chart - Design Defects DR PSP 10A C, Experienced
Expected Defects
25
20
15
10
5
0
1
4
7
10
13
16
19
22
25
28
31
34
37
Student

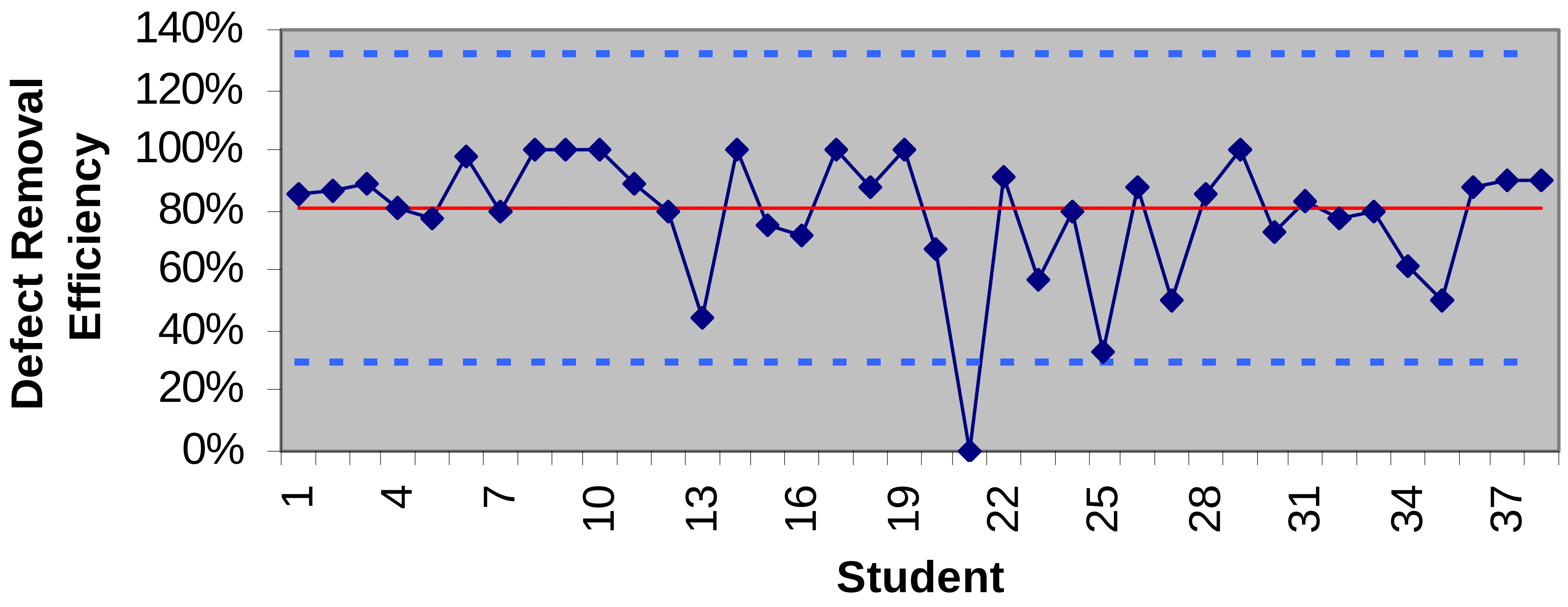
X Chart - Code Effort PSP 10A C, Experienced




X Chart - Code Review Rate PSP 10A C, Experienced
2500
2000
1500
1000
500
0
SLOC/Hour
1
4
7
10
13
16
19
22
25
28
31
34
37
Student


X Chart - Code Defects CR PSP 10A
C, Experienced
Expected Defects
0
2
4
6
8
10
12
14
1
4
7
10
13
16
19
22
25
28
31
34
37
Student

X Chart - Defect Removal Efficiency PSP 10A C, Experienced



PSP 10A -- Summary of Results

Variable	**$LNPL_x$**	**X_{bar}**	**$UNPL_x$**	**R_{bar}**	**UCL_R**
Design effort	*0*	*108*	*322*	*81*	*344*
Design review rate	*0*	*417*	*1453*	*390*	*1663*
Design review defects	*0*	*1.0*	*4.2*	*1.2*	*5.1*
Coding effort	*0*	*138*	*361*	*84*	*358*
Code review rate	*0*	*384*	*1075*	*260*	*1109*
Code review defects	*0*	*4.2*	*13.0*	*3.3*	*14.2*
Defect removal efficiency	*30%*	*81%*	*132%*	*19%*	*82%*

Observations

PSP 7A would not be a good time to drop out of the PSP class …

- **crucial learning is still occurring**

Between PSP 7A and 10A, a significant amount of process stabilization is occurring.

Both individuals (X) and moving range (mR) charts were generated in this analysis, but all signals on the mR chart corresponded to a signal in the X chart, therefore are not shown here.

Basis for High Variability

Potential for 10:1 difference in the performance of programmers (perhaps more)

- *200:1 difference in the performance of teams*

What is the impact of a disciplined process on consistency?

What is the impact of a "jelled" team on consistency?

What is the impact of a good working environment on consistency?

Three SPC Questions

What business value will SPC add to the organization?

- **important business objectives may include better quality, reduced cycle time, etc.**

What statistical techniques are appropriate?

- **correct implementation of appropriate techniques is crucial to success**

Will introducing SPC cause dysfunctional behavior?

- **measurement changes behavior**
- **can all critical factors be effectively measured?**

General SEI Information

SEI Customer Relations   +1 (412) 268-5800

SEI FAX number   +1 (412) 268-5758

Internet Address
customer-relations@sei.cmu.edu

Mailing Address
Customer Relations
Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh, PA 15213-3890

Internet Access to SEI

SEI Web pages

- ***www.sei.cmu.edu***
- ***www.sei.cmu.edu/cmm/***
- ***www.sei.cmu.edu/cmm/cmm.articles.html***
- ***www.sei.cmu.edu/cmm/slides/slides.html***

Extreme Programming from a CMM Perspective

Mark C. Paulk, *Software Engineering Institute*

Extreme Programming is an "agile methodology" that some people advocate for the high-speed, volatile world of Internet and Web software development. Although XP is a disciplined process, some have used it in arguments against rigorous software process improvement models such as the Software Capability Maturity Model.[1]

XP has good engineering practices that can work well with the CMM and other highly structured methods. The key is to carefully consider XP practices and implement them in the right environment.

In this article, I summarize both XP and the SW-CMM, show how XP can help organizations realize the SW-CMM goals, and then critique XP from a SW-CMM perspective.

The Software CMM

The Software Engineering Institute at Carnegie Mellon University developed the SW-CMM as a model for building organizational capability, and it has been widely adopted in the software community and beyond. As Table 1 shows, the SW-CMM is a five-level model that describes good engineering and management practices and prescribes improvement priorities for software organizations.

Although the SW-CMM is described in a book of nearly 500 pages, the requirements for becoming a Level 5 organization are concisely stated in 52 sentences—the 52 goals of the model's 18 key process areas (KPAs). The practices, subpractices, and examples that flesh out the model can guide software professionals in making reasonable, informed decisions about a broad range of process implementations.

The SW-CMM informative materials focus primarily on large projects and large organizations. With minor tailoring and common sense, however, the model can be applied in radically different environments, ranging from two- to three-person projects in small start-up companies to 500-person projects building hard real-time, life-critical systems.[2,3] The SW-CMM's rating components are intentionally abstract, capturing "universal truths" about high-performance software organizations. As a glance at Table 2 shows, the KPAs are clearly important to all types of software organizations.

With the exception of *software subcontract management*, which applies only to organizations that do subcontracting, the KPAs and their goals can apply to any software organization. Companies that focus on innovation more than operational excellence might downplay the role of consis-

Table 1

An overview of the Software CMM

Level	Focus	Key process areas
5: Optimizing	*Continual process improvement*	Defect prevention Technology change management Process change management
4: Managed	*Product and process quality*	Quantitative process management Software quality management
3: Defined	*Engineering processes and organizational support*	Organization process focus Organization process definition Training program Integrated software management Software product engineering Intergroup coordination Peer reviews
2: Repeatable	*Project management processes*	Requirements management Software project planning Software project tracking and oversight Software subcontract management Software quality assurance Software configuration management
1: Initial	*Competent people (and heroics)*	

tency, predictability, and reliability, but performance excellence is important even in highly innovative environments.

Extreme Programming

The XP method is typically attributed to Kent Beck, Ron Jeffries, and Ward Cunningham.[4,5] XP's target is small to medium-sized teams building software with vague or rapidly changing requirements. XP teams are typically colocated and have fewer than 10 members.

XP's critical underlying assumption is that developers can obviate the traditional high cost of change using technologies such as objects, patterns, and relational databases, resulting in a highly dynamic XP process. Beck's book is subtitled "Embrace Change," and XP teams typically deal with requirements changes through an iterative life cycle with short cycles.

The XP life cycle has four basic activities: coding, testing, listening, and designing. Dynamism is demonstrated through four values:

- continual communication with the customer and within the team;
- simplicity, achieved by a constant focus on minimalist solutions;
- rapid feedback through mechanisms such as unit and functional testing; and
- the courage to deal with problems proactively.

Principles in practice

Most of XP's principles—minimalism, simplicity, an evolutionary life cycle, user involvement, and so forth—are commonsense practices that are part of any disciplined process. As Table 3 summarizes, the "extreme" in XP comes from taking commonsense practices to extreme levels. Although some people may interpret practices such as "focusing on a minimalist solution" as hacking, XP is actually a highly disciplined process. Simplicity in XP terms means focusing on the highest-priority, most valuable system parts that are currently identified rather than designing solutions to problems that are not yet relevant (and might never be, given that requirements and operating environments change).

Although developers might use many different XP practices, the method typically consists of 12 basic elements:

- *Planning game*: Quickly determine the next release's scope, combining business priorities and technical estimates. The customer decides scope, priority, and dates from a business perspective, whereas technical people estimate and track progress.
- *Small releases*: Put a simple system into production quickly. Release new versions on a very short (two-week) cycle.
- *Metaphor:* Guide all development with a simple, shared story of how the overall system works.

Table 2

The Software CMM key process areas and their purpose

Key process area	Purpose
	Maturity Level 2: Repeatable
Requirements management	Establish a common understanding between the customer and software project team about the customer's requirements.
Software project planning	Establish reasonable plans for software engineering and overall project management.
Software project tracking and oversight	Provide adequate visibility into actual progress so that management can act effectively when the software project's performance deviates significantly from the software plans.
Software subcontract management	Select qualified software subcontractors and manage them effectively.
Software quality assurance	Provide management with appropriate visibility into the product and the software process.
Software configuration management	Establish and maintain the integrity of software products throughout the project's software life cycle.
	Maturity Level 3: Defined
Organization process focus	Establish organizational responsibility for software process activities that improve the organization's overall software process capability.
Organization process definition	Develop and maintain a usable set of software process assets that improve process performance across the projects and provide a basis for cumulative, long-term organizational benefits.
Training program	Develop individuals' skills and knowledge so they can perform their roles effectively and efficiently.
Integrated software management	Integrate the software engineering and management activities into a coherent, defined software process based on the organization's standard software process and related process assets.
Software product engineering	Consistently use a well-defined engineering process that integrates all the software engineering activities to produce correct, consistent software products effectively and efficiently.
Intergroup coordination	Establish a way for the software engineering group to participate actively with other engineering groups so that the project can effectively and efficiently satisfy customer needs.
Peer reviews	Remove defects from the software work products early and efficiently. An important corollary effect is to develop a better understanding of the software products and the preventable defects.
	Maturity Level 4: Managed
Quantitative process management	Quantitatively control the performance of the software project's process. Software process performance represents the actual results achieved from following a software process.
Software quality management	Quantify the quality of the project's software products and achieve specific quality goals.
	Maturity Level 5: Optimizing
Defect prevention	Identify the cause of defects and prevent them from recurring.
Technology change management	Identify new technologies (such as tools, methods, and processes) and introduce them into the organization in an orderly manner.
Process change management	Continually improve the organization's software processes with the goal of improving software quality, increasing productivity, and decreasing the product-development cycle time.

- *Simple design*: Design as simply as possible at any given moment.
- *Testing*: Developers continually write unit tests that must run flawlessly; customers write tests to demonstrate functions are finished. "Test, then code" means that a failed test case is an entry criterion for writing code.
- *Refactoring*: Restructure the system without changing its behavior to remove duplication, improve communication, simplify, or add flexibility.
- *Pair programming*: All production code is written by two programmers at one machine.
- *Collective ownership*: Anyone can improve any system code anywhere at any time.
- *Continuous integration*: Integrate and build the system many times a day (every time a task is finished). Continual regression testing prevents functionality regressions when requirements change.
- *40-hour weeks*: Work no more than 40 hours per week whenever possible; never work overtime two weeks in a row.
- *Onsite customer*: Have an actual user on the team full-time to answer questions.
- *Coding standards*: Have rules that emphasize communication throughout the code.

These basic practices work together to cre-

Table 3

The "extreme" in Extreme Programming

Commonsense	XP extreme	XP implementation practice
Code reviews	Review code all the time	Pair programming
Testing	Test all the time, even by customers	Unit testing, functional testing
Design	Make design part of everybody's daily business	Refactoring
Simplicity	Always work with the simplest design that supports the system's current functionality	The simplest thing that could possibly work
Architecture	Everybody works to refine the architecture all the time	Metaphor
Integration testing	Integrate and test several times a day	Continuous integration
Short iterations	Make iterations extremely short—seconds, minutes, and hours rather than weeks, months, and years	Planning game

ate a coherent method. XP characterizes the full system functionality using a pool of "stories," or short feature descriptions. For the planning game and small releases, the customer must select a subset of stories that characterize the most desirable work for developers to implement in the upcoming release. Because the customer can add new stories to the pool at any time, requirements are highly volatile. However, volatility is managed by implementing functionality in two-week chunks. Having a customer onsite supports this ongoing cycle of two-week releases.

XP developers generate a metaphor to provide the project's overarching vision. Although you could view this as a high-level architecture, XP emphasizes design, while at the same time minimizing design documentation. Some people have characterized XP as not allowing documentation outside code, but that is not quite accurate. Because XP emphasizes continual redesign—using refactoring whenever necessary—there is little value to detailed design documentation (and maintainers rarely trust anything other than the code anyway).

XP developers typically throw away design documentation after the code is written, although they will keep it if it's useful. They also keep design documentation when the customer stops coming up with new stories. At that point, it's time to put the system in mothballs and write a five- to 10-page "mothball tour" of the system. A natural corollary of the refactoring emphasis is to always implement the simplest solution that satisfies the immediate need. Requirements changes are likely to supersede "general solutions" anyway.

Pair programming is one of XP's more controversial practices, mainly because it has resource consequences for the very managers who decide whether or not to let a project use XP. Although it might appear that pair programming consumes twice the resources, research has shown that it leads to fewer defects and decreased cycle time.[6] For a jelled team, the effort increase can be as little as 15 percent, while cycle time is reduced by 40 to 50 percent. For Internet-time environments, the increased speed to market may be well worth the increased effort. Also, collaboration improves problem solving, and increased quality can significantly reduce maintenance costs. When considered over the total life cycle, the benefits of pair programming often more than pay for added resource costs.

Because XP encourages collective ownership, anyone can change any piece of code in the system at any time. The XP emphasis on continuous integration, continual regression testing, and pair programming protects against a potential loss of configuration control. XP's emphasis on testing is expressed in the phrase, "test, then code." It captures the principle that developers should plan testing early and develop test cases in parallel with requirements analysis, although the traditional emphasis is on black-box testing. Thinking about testing early in the life cycle is standard practice for good software engineering, though it is too rarely practiced.

The basic XP management tool is the metric, and the metric's medium is the "big visible chart." In the XP style, three or four measures are typically all a team can stand at one time, and those should be actively used and visible. One recommended XP metric is "project velocity"—the number of stories of a given size that developers can implement in an iteration.

Adoption strategies

XP is an intensely social activity, and not everyone can learn it. There are two conflicting attitudes toward XP adoption. XP is generally viewed as a system that demonstrates emergent properties when adopted as a whole. As the discussion thus far shows, there are strong dependencies between many XP practices, such as collective ownership and continuous integration.

Nonetheless, some people recommend adopting XP one practice at a time, focusing on the team's most pressing current problem. This is consistent with the attitude toward change that XP is "just rules" and the team can change the rules anytime as long as they agree on how to assess the change's effects. Beck, for example, describes XP practices as "etudes": They help developers master the techniques, but experienced users can modify them as necessary.

XP and the CMM

The SW-CMM focuses on both the management issues involved in implementing effective and efficient processes and on systematic process improvement. XP, on the other hand, is a specific set of practices—a "methodology"—that is effective in the context of small, colocated teams with rapidly changing requirements. Taken together, the two methods can create synergy, particularly in conjunction with other good engineering and management practices. I'll now illustrate this by discussing XP practices in relation to the CMM KPAs and goals outlined in Table 2.

XP and Level 2 practices

XP addresses Level 2's *requirements management* KPA through its use of stories, an onsite customer, and continuous integration. Although system requirements might evolve dramatically over time, XP integrates feedback on customer expectations and needs by emphasizing short release cycles and continual customer involvement. "Common understanding" is established and maintained through the customer's continual involvement in building stories and selecting them for the next release (in effect, prioritizing customer requirements).

XP addresses *software project planning* in the planning game and small releases. XP's planning strategy embodies Watts Humphrey's advice, "If you can't plan well, plan often." The first three activities of this KPA deal with getting the software team involved in early planning. XP integrates the software team into the commitment process by having it estimate the effort involved to implement customer stories; at the level of two-week releases, such estimates are typically quite accurate. The customer maintains control of business priorities by choosing which stories to implement in the next release with the given resources. By definition, the XP life cycle is both incremental and evolutionary. The project plan is not detailed for the project's whole life cycle, although the system metaphor does establish a vision for project direction. As a result, developers can identify and manage risks efficiently.

XP addresses *software project tracking and oversight* with the "big visual chart," project velocity, and commitments (stories) for small releases. XP's commitment process sets clear expectations for both the customer and the XP team at the tactical level and maximizes flexibility at the project's strategic level. The emphasis on 40-hour weeks is a general human factors concern; although CMM does not address it, having "rational work hours" is usually considered a best practice. XP also emphasizes open workspaces, a similar "people issue" that is outside CMM's scope. XP does not address *software subcontract management*, which is unlikely to apply in XP's target environment.

While an independent *software quality assurance* group is unlikely in an XP culture, SQA could be addressed by the pair-programming culture. Peer pressure in an XP environment can achieve SQA's aim of assuring conformance to standards, though it does not necessarily give management visibility into nonconformance issues. Dealing with process and product assurance using peer pressure can be extraordinarily effective in a small team environment. However, larger teams typically require more formal mechanisms for objectively verifying adherence to requirements, standards, and procedures. Also, peer pressure might be ineffective when the entire team is being pushed, just as a software manager might be vulnerable to external pressure. This vulnerability should be addressed at the organizational level when considering SQA.

Although not completely and explicitly addressed, *software configuration management* is implied in XP's collective ownership, small releases, and continuous integration. Collective ownership might be problematic for large systems, where more formal com-

Taken together, the two methods can create synergy, particularly in conjunction with other good engineering and management practices.

Table 4

XP satisfaction of key process areas, given the appropriate environment

Level	Key process area	Satisfaction
2	Requirements management	++
2	Software project planning	++
2	Software project tracking and oversight	++
2	Software subcontract management	—
2	Software quality assurance	+
2	Software configuration management	+
3	Organization process focus	+
3	Organization process definition	+
3	Training program	—
3	Integrated software management	—
3	Software product engineering	++
3	Intergroup coordination	++
3	Peer reviews	++
4	Quantitative process management	—
4	Software quality management	—
5	Defect prevention	+
5	Technology change management	—
5	Process change management	—

+	*Partially addressed in XP*
++	*Largely addressed in XP (perhaps by inference)*
—	*Not addressed in XP*

munication channels are necessary to prevent configuration management failures.

XP and Level 3 practices

At Level 3, XP addresses *organization process focus* at the team rather than organizational level. A focus on process issues is nonetheless implied in adopting XP one practice at a time, as well as in the "just rules" philosophy. Because XP focuses on software engineering process rather than organizational infrastructure issues, organizations adopting XP must address this and other organization-level processes, whether in a CMM-based context or not.

Similarly, the various XP-related books, articles, courses, and Web sites partially address the *organization process definition* and *training program* KPAs, but organizational assets are outside the scope of the XP method itself. As a consequence, XP cannot address *integrated software management* because there may not be any organizational assets to tailor.

Several XP practices effectively address *software product engineering*: metaphor, simple design, refactoring, the "mothball" tour, coding standards, unit testing, and functional testing. XP's de-emphasis of design documentation is a concern in many environments, such as hard real-time systems, large systems, or virtual teams. In such environments, good designs are crucial to success, and using the refactoring strategy would be high-risk. For example, if developers performed refactoring after a technique such as rate-monotonic analysis proved that a system satisfied hard real-time requirements, they'd have to redo the analysis. Such an environment invalidates XP's fundamental assumption about the low cost of change.

XP's emphasis on communication—through onsite customers and pair programming—appears to provide as comprehensive a solution to *intergroup coordination* as integrated product and process development. In fact, XP's method might be considered an effective IPPD approach, although the software-only context ignores multidiscipline environments.

Pair programming addresses *peer reviews*, and is arguably more powerful than many peer review techniques since it adopts preventive concepts found in code reading and literate programming. However, pair programming's relative lack of structure can lessen its effectiveness. Empirical data on pair programming is currently sparse but promising.[6] To make informed trade-off decisions, we'll need more empirical research that contrasts and compares pair programming and peer review techniques, especially more rigorous techniques such as inspections.

Beyond Level 3

XP addresses few of the Level 4 and 5 KPAs in a rigorous statistical sense, although feedback during rapid cycles might partially address *defect prevention*. Table 4 summarizes XP's potential to satisfy CMM KPAs, given the appropriate domain.

Many of the KPAs that XP either ignores or only partially covers are undoubtedly addressed in real projects. XP needs management and infrastructure support, even if it does not specifically call for it.

Discussion

As the earlier comparison shows, XP generally focuses on technical work, whereas the CMM generally focuses on management issues. Both methods are concerned with "culture." The element that XP lacks that is crucial for the SW-CMM is the concept of "institutionalization"—that is, establishing a culture of "this is the way we do things around here."

Table 5

XP and institutionalization practices

Common feature (in each KPA)	Practice	Satisfaction
Commitment to perform	Policy	—
	Leadership and sponsorship	—
Ability to perform	Organizational structures	+
	Resources and funding	+
	Training	+
Measurement and analysis	Measurement	+
Verifying implementation	Senior management oversight	—
	Project management oversight	++
	Software quality assurance	+

+	*Partially addressed in XP*
++	*Largely addressed in XP (perhaps by inference)*
—	*Not addressed in XP*

Although implicit in some practices, such as the peer pressure arising from pair programming, XP largely ignores the infrastructure that the CMM identifies as key to institutionalizing good engineering and management practices. Table 5 summarizes XP's coverage of institutionalization in its domain.

The CMM's KPAs share common features that implement and institutionalize processes. Each KPA's institutionalization practices map to the area's goals; a naïve XP implementation that ignored these infrastructure issues would fail to satisfy any KPA. XP ignores some of these practices, such as policies. XP addresses others, such as training and SQA, by inference. It addresses still others—project-specific practices such as management oversight and measurement—to a limited degree. As an implementation model focused on the development process, these issues are largely outside XP's focus, but they are arguably crucial for its successful adoption.

Size matters

Much of the formalism that characterizes most CMM-based process improvement is an artifact of large projects and severe reliability requirements, especially for life-critical systems. The SW-CMM's hierarchical structure, however, is intended to support a range of implementations through the 18 KPAs and 52 goals that comprise the requirements for a fully mature software process.

As systems grow, some XP practices become more difficult to implement. XP is, after all, targeted toward small teams working on small- to medium-sized projects. As projects become larger, emphasizing a good architectural "philosophy" becomes increasingly critical to project success. Major investment in product architecture design is one of the practices that characterizes successful Internet companies.[7]

Architecture-based design, designing for change, refactoring, and similar design philosophies emphasize the need to manage change systematically. Variants of the XP bottom-up design practices, such as architecture-based design, might be more appropriate in large-project contexts. In a sense, architectural design that emphasizes flexibility is the goal of any good object-oriented methodology, so XP and object orientation are well suited to one another. Finally, large projects tend to be multidisciplinary, which can be problematic given that XP is aimed at software-only projects.

Why explore XP?

Modern software projects should capture XP values, regardless of how radically their implementation differs from XP's. Organizations might call communication and simplicity by other names, such as coordination and elegance, but without these values, nontrivial projects face almost insurmountable odds.

XP's principles of communication and simplicity are also fundamental for organizations using the SW-CMM. When defining processes, organizations should capture the minimum essential information needed, structure definitions using good software design principles (such as information hiding and abstraction), and emphasize usefulness and usability.[2]

For real-time process control, rapid feedback is crucial. Previous eras have captured this idea in aphorisms such as "don't throw good money after bad"; in a quantitative sense, we can view this as the soul of the CMM's Level 4. One of the consequences of the cultural shift between Levels 1 and 2 is the need to demonstrate the courage of our convictions by being realistic about estimates, plans, and commitments.

False opposition

The main objection to using XP for process improvement is that it barely touches the management and organizational issues that the SW-CMM emphasizes. Implementing the kind of highly collaborative environment that XP assumes requires enlightened management and appropriate organizational infrastructure.

The argument that CMM's ideal of a rigorous, statistically stable process is antithetical to XP is unconvincing. XP has disciplined processes, and the XP process itself is clearly well defined. We can thus consider CMM and XP complementary. The SW-CMM tells organizations what to do in general terms, but does not say how to do it. XP is a set of best practices that contains fairly specific how-to information—an implementation model—for a particular type of environment. XP practices can be compatible with CMM practices (goals or KPAs), even if they do not completely address them.

Most of XP consists of good practices that all organizations should consider. While we can debate the merits of any one practice in relation to other options, to arbitrarily reject any of them is to blind ourselves to new and potentially beneficial ideas.

To put XP practices together as a methodology can be a paradigm shift similar to that required for concurrent engineering. Although its concepts have been around for decades, adopting concurrent engineering practices changes your product-building paradigm. XP provides a systems perspective on programming, just as the SW-CMM provides a systems perspective on organizational process improvement. Organizations that want to improve their capability should take advantage of the good ideas in both, and exercise common sense in selecting and implementing those ideas.

Should organizations use XP, as published, for life-critical or high-reliability systems? Probably not. XP's lack of design documentation and de-emphasis on architecture is risky. However, one of XP's virtues is that you can change and improve it for different environments. That said, when you change XP, you risk losing the emergent properties that provide value in the proper context. Ultimately, when you choose and improve software processes, your emphasis should be to let common sense prevail—and to use data whenever possible to offer insight on challenging questions.

Acknowledgments

I gratefully acknowledge Kent Beck, Steve McConnell, and Laurie Williams for their comments.

References

1. M.C. Paulk et al., *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison-Wesley, Reading, Mass., 1995.
2. M.C. Paulk, "Using the Software CMM with Good Judgment," *ASQ Software Quality Professional*, vol. 1, no. 3, June 1999, pp. 19–29.
3. D.L. Johnson and J.G. Brodman, "Applying CMM Project Planning Practices to Diverse Environments," *IEEE Software*, vol. 17, no. 4, July/Aug. 2000, pp. 40–47.
4. K. Beck, *Extreme Programming Explained: Embrace Change,* Addison-Wesley, Reading, Mass., 1999.
5. "eXtreme Programming Pros and Cons: What Questions Remain?" *IEEE Computer Soc. Dynabook,* J. Siddiqi, ed., Nov. 2000; www.computer.org/seweb/dynabook/index.htm (current 24 Sept. 2001).
6. L. Williams et al., "Strengthening the Case for Pair Programming," *IEEE Software*, vol. 17, no. 4, July/Aug. 2000, pp. 19–25.
7. A. MacCormack, "Product-Development Practices that Work: How Internet Companies Build Software," *MIT Sloan Management Rev.*, no. 42, vol. 2, Winter 2001, pp. 75–84.

About the Author

Mark Paulk is a senior member of the technical staff at the Software Engineering Institute. His current interests are in high-maturity practices and statistical control for software processes. He was "book boss" for Version 1.0 of the Capability Maturity Model for Software and project leader during the development of Software CMM Version 1.1. He is also involved with software engineering standards, including ISO 15504, ISO 12207, and ISO 15288. He received his bachelor's degree in mathematics and computer science from the University of Alabama in Huntsville and his master's degree in computer science from Vanderbilt University. Contact him at the Software Engineering Inst., Carnegie Mellon Univ., Pittsburgh, PA 15213; mcp@sei.cmu.edu.

List of High Maturity Organizations

Last updated October 2002

The following list of high maturity organizations was compiled as part of our Survey of High Maturity Organizations and High Maturity Workshop research. By popular request, it is being revived (at least temporarily). This Web page will probably be combined with the "List of Published Maturity Levels" at some point in the future. For further information on published maturity levels, see that list and the maturity profile presentation at **www.sei.cmu.edu/sema/welcome.html**. Note that an organization is not included in this list without its explicit permission.

The full set of 146 high maturity organizations, a subset of which is listed below, includes:

- 72 Level 4 organizations
- 74 Level 5 organizations

It is interesting to note that 87 of the high maturity organizations assessed are outside the United States.

Country	**Number of Maturity Level 4 Organizations**	**Number of Maturity Level 5 Organizations**
Australia	**2**	
Canada		**1**
China		**2**
France	**1**	
India	**27**	**50**
Ireland	**1**	
Israel	**1**	
Russia		**1**
Singapore	**1**	
USA	**39**	**20**

Please be aware of the following issues regarding this list.

- The SEI does not certify companies at maturity levels.
- The SEI does not confirm the accuracy of the maturity levels reported by the Lead Assessors or organizations.
- This list of Level 4 and 5 organizations is by no means exhaustive; we know of other high maturity organizations that have chosen not to be listed.
- The SEI did not use information stored within its Process Appraisal Information System to produce this document.
- The organizations listed gave explicit permission to publish this information.
- No information obtained in confidence was used to produce this list.

The following information is contained in this table, as reported by the organization:

- **Full name of the organization** (with acronyms defined), including city and state (or country)
- **Point of contact**: name and email address
- **Maturity level assessed**
- **Month and year of assessment**
- **Lead Assessor(s)** (Lead Evaluators are annotated with LE. Some appraisers are both LAs and LEs. Some Lead Assessors are now inactive (I) and no longer listed on the LA and LE lists.) (Include the form of assessment if different from CBA IPI with Lead Assessor.)

Name of Organization	**Country**	**Contact**	**Level**	**Date Assessed**	**Lead Assessor(s)**
Alitec, Laval	**France**	Jerome Barbier, jeb@alitec.net Jean Noel Martin, jnm@alitec.net	4	July 2000	Jean-Yves Le Goic
Atos Origin India (formerly Origin Information Technology India Limited), Mumbai	**India**	Darayus Desai, darayus.desai@atosorigin.com	5	Nov 2000	*(CAF-compliant Process Professional Assessment Method)* *(Cyril Dyer - Compita Assessor)*
BAE SYSTEMS, Communication, Navigation and Identification (CNI) Division, Wayne, NJ	**USA**	Peter J. Howard, peter.howard@baesystems.com	5	March 2002	Marilyn Bush
BFL Software Limited, Bangalore	**India**	Madhukumar P.S., Madhukumar.PS@bflsoftware.com	4	June 1999	Carolyn Swanson
Boeing Company, Aircraft & Missiles & Phantom Works Southern California, Long Beach, CA	**USA**	George H. Kasai, george.h.kasai@boeing.com	5	Dec 1997	Andy Felschow Jeff Facemire
Boeing Company, Military Aircraft & Missile Systems F/A-18 Mission Computer, St. Louis, MO	**USA**	Bruce A. Boyd, bruce.a.boyd@boeing.com Robert L. Allen, robert.l.allen3@boeing.com	4	Nov 1999	*(SCE)* *Roy Queen (LE)* *Jeff Perdue*
Boeing Company, Reusable Space Systems and Satellite Programs, Downey & Seal Beach, CA	**USA**	Don Dillehunt, donald.d.dillehunt@boeing.com	5	Oct 1999	Andy Felschow Jeff Facemire

Name of Organization	**Country**	**Contact**	**Level**	**Date Assessed**	**Lead Assessor(s)**
Boeing Company, Space Transportation Systems, Kent, WA	**USA**	Gary Wigle, gary.b.wigle@boeing.com	5	July 1996	Steve Masters Mark Paulk John Vu
CG-Smith Software Limited, Bangalore	**India**	G.N. Raghavendra Swamy, raghav@cgs.cgsmith.soft.net	5	Sept 1999	Richard Storch
Citicorp Overseas Software Limited (COSL), Mumbai	**India**	Makarand Khandekar, makarand.khandekar@citicorp.com	5	Oct 1999	John Sheckler
Cognizant Technology Solutions, Chennai	**India**	Emani BSP Sarathy, esarathy@chn.cts-corp.com	5	Sept 2000	V. Kannan
Computer Sciences Corporation (CSC), Aegis Program, , Moorestown, NJ	**USA**	Wendy Irion Talbot, wirionta@csc.com	5	March 2001	*Kathryn Gallucci (LE)*
Computer Sciences Corporation (CSC), Civil Group, Greenbelt, MD	**USA**	Mel Wahlberg, mwahlber@csc.com	4	Jan 2001	Paul Byrnes (LA & LE)
Computer Sciences Corporation (CSC), Civil Group, Systems, Engineering, and Analysis Support (SEAS) Center, Greenbelt, MD	**USA**	Frank McGarry, fmcgarry@csc.com Mel Wahlberg, mwahlber@csc.com	5	Nov 1998	*(SCE)* *Paul Byrnes (LA & LE)*
Computer Sciences Corporation (CSC), Defense Group Aerospace Information Technologies, Dayton, OH	**USA**	Cheryl Plak, cplak@csc.com	5	Feb 1999	*(SCE)* *Kathryn Gallucci (LE)*
Computer Sciences Corporation (CSC), Integrated Systems Division (ISD), Moorestown, NJ	**USA**	Bryan Cooper, bcooper1@csc.com	4	May 1998	*(SCE)* *Paul Byrnes (LA & LE)*
Computer Sciences Corporation (CSC), Tactical Systems Center (TSC), Moorestown, NJ	**USA**	Wendy Irion Talbot, wirionta@csc.com Jeff McGarry, jmcgarr1@csc.com	4	May 1998	*(SCE)* *Paul Byrnes (LA & LE)*
Covansys, San Francisco, CA	**USA**	Prasanth Kedarisetty, KPrasanth@Covansys.com	4	Jan 2001	Richard Knudson
DCM Technologies, DCM ASIC Technology Limited, New Delhi	**India**	Naresh C. Maheshwari, ncm@dcmds.co.in	5	April 2000	Richard Storch
DSQ Software, Chennai	**India**	K.N. Ananth, kna@md.in.dsqsoft.com	4	June 1998	Judy Bamberger
Eastern Software Systems Ltd., New Delhi	**India**	Rajyashree, r-agarwala@essindia.co.in	5	April 2002	Santhanakrishnan Srinivasan
Future Software Private Limited, Chennai	**India**	M.G. Thomas, thomasmg@future.futsoft.com	4	June 1999	Pradeep Udhas

Name of Organization	Country	Contact	Level	Date Assessed	Lead Assessor(s)
HCL Perot Systems, Noida and Bangalore	**India**	Rakesh Soni, rakesh.soni@hpsglobal.com	5	Feb 2000	Pradeep Udhas
HCL Technologies Limited, Applications Solutions Development Centre, Chennai	**India**	N. N. Jha, nnjha@msdc.hcltech.com	4	May 2000	V. Kannan
HCL Technologies Limited, Core Technologies Division, Chennai	**India**	K. R. Gopinath, krg@hclt.com	4	Dec 2000	Krishnamurthy Kothandaraman Raman
HCL Technologies Limited, Gurgaon Software Development Center, Gurgaon	**India**	Sanjeev Gupta, gsanjeev@ggn.hcltech.com	5	June 2001	V. Kannan
Hewlett Packard India Software Operations Limited, Bangalore	**India**	Kousthuba, kou@india.hp.com	5	June 2000	Richard Storch
Hexaware Technologies Limited, Mumbai and Chennai Operations, Chennai	**India**	Sulochana Ganesan, sulochana@hexaware.co.in	5	Dec 2000	V. Kannan
Honeywell International, Avionics Integrated Systems (formerly AlliedSignal, Guidance & Control Systems), Teterboro, NJ	**USA**	Steve Janiszewski, stephen.janiszewski@honeywell.com	4	Nov 1996	Larry Bramble (I)
Hughes Software Systems, Bangalore and Gurgaon	**India**	Gautam Brahma, gbrahma@hss.hns.com	4	Jan 2000	V. Kannan
IBM Global Services India, Bangalore	**India**	Asha Goyal, gasha@in.ibm.com Maya Srihari, smaya@in.ibm.com	5	Nov 1999	Richard Storch
i-flex solutions limited (formerly Citicorp Information Technology Industries Limited aka CITIL), Bangalore	**India**	Vivek V. Govilkar, vivek.govilkar@iflexsolutions.com	4	Dec 1995	Ken Dymond
i-flex solutions limited (formerly Citicorp Information Technology Industries Limited aka CITIL), Mumbai	**India**	Vivek Govilkar, vivek.govilkar@citicorp.com	4	Dec 1995	Cindi Wise Ken Dymond
i-flex solutions limited Data Warehouse Center of Excellence, Bangalore	**India**	Vivek V. Govilkar, vivek.govilkar@iflexsolutions.com	5	Nov 1999	Ken Dymond Santhanakrishnan Srinivasan Anand Kumar
i-flex solutions limited IT Services Division, Bangalore	**India**	Anand Kumar, anand.kumar@iflexsolutions.com	5	Dec 2000	Santhanakrishnan Srinivasan Anand Kumar

Name of Organization	**Country**	**Contact**	**Level**	**Date Assessed**	**Lead Assessor(s)**
i-flex solutions limited IT Services Division, Mumbai	**India**	Anand Kumar, anand.kumar@iflexsolutions.com	5	Dec 2000	Santhanakrishnan Srinivasan Anand Kumar Atul Gupta
Information Technologies (India) Ltd., New Delhi	**India**	Arvind Sinha, arvindks@itil.com	5	April 2001	Srinivas Thummalapalli
Information Technology (India) Ltd. , Delhi	**India**	Madhumita Poddar Sen, madhumitap@itil.com	4	April 2000	Pradeep Udhas
Intelligroup Asia Private Limited, Advanced Development Center, Hyderabad	**India**	G.V.S. Sharma, gvs.sharma@intelligroup.co.in	5	Oct 2000	Raghav S. Nandyal John Harding
ITC Infotech India Limited, Bangalore	**India**	Paresh Master, pareshmaster@vsnl.com	5	Aug 2000	Richard Storch
Kshema Technologies Limited, Bangalore	**India**	V. Bhaskar, vbhaskar@kshema.com	5	July 2002	Krishnamurthy Kothanda Raman
L & T Information Technology Limited, Chennai	**India**	Anil S. Pandit, anil.pandit@vashimail.ltitl.com	4	Feb 2000	V. Kannan
Litton Guidance and Control Systems, Woodland Hills, CA	**USA**	Roy Nakahara, nakaharr@littongcs.com	4	Dec 1998	Mark Amaya
Litton/PRC Inc., McLean, VA and Colorado Springs, CO	**USA**	Al Pflugrad, pflugrad_al@prc.com	5	March 2000	*(SCE)* *Joseph Morin (LE)*
Lockheed Martin Aeronautics Company (formerly Lockheed Martin Tactical Aircraft Systems - LMTAS), Fort Worth, TX	**USA**	Phil Gould, philip.c.gould@lmco.com	4	Dec 1999	Leia Bowers White
Lockheed Martin Air Traffic Management, Rockville, MD	**USA**	Jim Sandford, jim.sandford@lmco.com	4	Dec 1999	Carol Granger-Parker Jeff Facemire
Lockheed Martin Federal Systems, Owego, NY	**USA**	Ed Fontenot, ed.fontenot@lmco.com Warren A. Schwomeyer, warren.schwomeyer@lmco.com	5	Dec 1997	John Travalent Mary Busby
Lockheed Martin Information Systems, Orlando, FL	**USA**	Michael Ziomek, michael.ziomek@lmco.com	4	June 2000	Gene Jorgensen
Lockheed Martin Management & Data Systems, King of Prussia, PA	**USA**	M. Lynn Penn, mary.lynn.penn@lmco.com	5	Dec 2000	Andy Felschow Carol Granger-Parker Dennis Ring

Name of Organization	Country	Contact	Level	Date Assessed	Lead Assessor(s)
Lockheed Martin Mission Systems, Gaithersburg, MD	**USA**	Paul Weiler, paul.weiler@lmco.com Al Aldrich, al.aldrich@lmco.com	5	Oct 1999	*(SCE)* *Paul Byrnes (LA & LE)*
Lockheed Martin Naval Electronics & Surveillance Systems - Syracuse, Syracuse NY	**USA**	Peter Barletto, pete.barletto@lmco.com	5	Nov 1999	Carol Granger-Parker Andy Felschow
Lockheed Martin Naval Electronics & Surveillance Systems – Eagan, Eagan, MN	**USA**	John Travalent, john.travalent@lmco.com	4	Oct 1999	Mary Busby
Lockheed Martin Naval Electronics & Surveillance Systems – Manassas (formerly Undersea Systems), Manassas, VA	**USA**	Dana Roper, dana.roper@lmco.com	5	Feb 1999	Judah Mogilensky John Travalent Donald White
Lockheed Martin Naval Electronics & Surveillance Systems – Moorestown, Moorestown, NJ	**USA**	Nghia N. Nguyen, nghia.n.nguyen@lmco.com Jeff Tait, jeffery.a.tait@lmco.com	4	Dec 1999	Kevin Schaan Kent Johnson Dennis Ring
Lockheed Martin Space Electronics and Communications Systems – Manassas (formerly Loral Federal Systems), Manassas, VA	**USA**	Dana Roper, dana.roper@lmco.com	4	June 1995	Judah Mogilensky John Travalent Chris Manak (I)
Mastek Limited, Mumbai	**India**	P. Rajshekharan, rajshekhar@mastek.com	5	Sept 2000	Ron Radice
Motorola Australia Software Centre, Adelaide	**Australia**	Peter Dew, pdew@asc.corp.mot.com	4	Aug 1997	John Pellegrin (I)
Motorola China Software Center, Beijing & Nanjing	**China**	John Jun'an Yu, johny@sc.mcel.mot.com	5	Sept 2000	Dan Weinberger Patricia McNair
Motorola India Electronics Ltd. (MIEL), Bangalore	**India**	Sarala Ravishankar, sarala@miel.mot.com	5	Nov 1993	John Pellegrin (I)
Motorola, Asia Pacific Telecom Carrier Solutions Group (TCSG) Applied R&D Center, Beijing	**China**	Graham Hu, qch1422@email.mot.com	5	Dec 2000	*(CAF-compliant Motorola QSR Subsystem 10 Software Assessment)* *(Fathi Hakam -- Motorola Assessor)*

Name of Organization	**Country**	**Contact**	**Level**	**Date Assessed**	**Lead Assessor(s)**
Motorola, GSM (Global System for Mobile Communications) Systems Division, Network Systems Group, Arlington Heights, IL	**USA**	Barbara Hirsh, hirsh@cig.mot.com	5	Oct 1997	*(CAF-compliant Motorola QSR Subsystem 10 Software Assessment)* *(Ellen Pickthall -- Motorola Assessor)*
NCR Corporation, Teradata Development Division, Massively Parallel Systems, San Diego, CA	**USA**	Ron Weidemann, ron.weidemann@sandiegoca.ncr.com	4	Oct 1999	Ron Weidemann
NeST Information Technologies (P) Ltd., Kerala	**India**	Mary Roselind Michael, mary.roselind@nestinfotech.com M.I. Theodore, theodore@nestinfotech.com	5	Nov 2001	Jack Hilbing
Network Systems and Technologies (P) Ltd, Trivandrum	**India**	S K Pillai, skp@nestec.net	5	May 2000	Ron Radice
NIIT Limited, New Delhi	**India**	Bhaskar Chavali, BhaskarC@niit.com	5	Sept 1999	Richard Storch
Northrop Grumman Electronic Sensors and Systems Sector (ESSS), Baltimore, MD	**USA**	Eva M. Brandt, eva_m_brandt@md.northgrum.com	4	Oct 1999	John Blyskal
Northrop Grumman, Air Combat Systems, Integrated Systems and Aeronautics Sector, El Segundo, CA	**USA**	Leitha Purcell, purcele@mail.northgrum.com	4	Oct 1998	Don Dortenzo
Northrop Grumman, Integrated Systems & Aerostructures, AEW & EW Systems (formerly Surveillance & Battle Management), Bethpage, NY	**USA**	Dennis Carter, cartede@mail.northgrum.com	4	Oct 1998	Andy Felschow
Oracle Software India Limited, India Development Center, Bangalore	**India**	Ashish Saigal, asaigal@in.oracle.com	4	May 1999	Pradeep Udhas
Oracle Solution Services (India) Private Limited, Bangalore	**India**	Jomon Jose, Jomon.Jose@oracle.com	4	June 2001	K.K. Raman
Patni Computer Systems Ltd. (PCS), Mumbai, Navi Mumbai, Pune and Gandhinagar Facilities, Mumbai	**India**	Sunil Kuwalekar, sunil.kuwalekar@patni.com N A Nagwekar, nilendra.nagwekar@patni.com	5	Aug 2000	Pradeep Udhas

Name of Organization	Country	Contact	Level	Date Assessed	Lead Assessor(s)
Philips Software Centre Private Limited, Bangalore	**India**	Ramachandran KV, ramachandran.kv@philips.com	5	June 2000	Richard Knudson
Raytheon (formerly Raytheon E-Systems), Garland, TX	**USA**	Mary E. Howard, mary_e_howard@raytheon.com	4	Dec 1998	Neil Potter
Raytheon C3I Fullerton Integrated Systems, Command and Control Systems/Middle East Operations, Fullerton, CA	**USA**	Jane A. Moon, jmoon@west.raytheon.com Janet Bratton, jabratton@west.raytheon.com	5	Oct 1998	Paul Byrnes (LA & LE) Jane Moon Ronald Ulrich Ivan Flinn Bruce Duncil (LA & LE) Janet Bratton
Raytheon Electronic Systems, Air & Missile Defense Systems Surface Radar (AMDS/SR), Tewksbury, Bedford, Sudbury, MA	**USA**	Michael Campo, Michael_J_Campo@res.raytheon.com	4	June 2001	Janet Bratton Ivan Flinn
Raytheon Missile Systems, Software Engineering Center, Tucson, AZ	**USA**	Michael D. Scott, mscott1@west.raytheon.com	4	Oct 1998	John Ryskowski Michael Scott
Raytheon, Electronic Systems, Sensors Engineering, El Segundo, CA	**USA**	Paul Curry, pcurry@west.raytheon.com	4	Oct 2000	Janet Bratton Michael Scott Ivan Flinn
Satyam Computer Services Ltd	**India**	Prabhuu Sinha, prabhuu@satyam.com	5	March 1999	Richard Knudson
Siemens Information Systems Limited (SISL), Software Development Strategic Business Unit (SBU), Bangalore	**India**	T. Kathavarayan, kathavarayan.t@sisl.co.in	5	Sept 2001	Ajay Batra
Siemens Information Systems Limited (SISL), Telecom & Major Projects Strategic Business Unit (SBU), Gurgaon	**India**	Neelima Yadav, Neelima.Yadav@sisl.co.in	5	Aug 2001	Ajay Batra
Silverline Technologies Limited, Mumbai	**India**	S. Purushotham, sp@silverline.com	4	Dec 1999	V. Kannan
Syntel, Mumbai and Chennai	**India**	Jonathan James, jonathan_james@syntelinc.com	5	March 2001	Ajay Batra
Tata Consultancy Services, Ahmedabad	**India**	Gargi Keeni, gkeeni@mumbai.tcs.co.in Rosemary Hedge, rhedge@ahd.tcs.co.in	5	Nov 2000	Ron Radice P. Suresh
Tata Consultancy Services, Ambattur, Chennai	**India**	Gargi Keeni, gkeeni@mumbai.tcs.co.in	5	July 2000	Ron Radice

Name of Organization	**Country**	**Contact**	**Level**	**Date Assessed**	**Lead Assessor(s)**
Tata Consultancy Services, Bangalore	**India**	Gargi Keeni, gkeeni@mumbai.tcs.co.in Uma Rijhwani, umarijhwani@blore.tcs.co.in	5	Jan 2000	Ron Radice
Tata Consultancy Services, Calcutta	**India**	Gargi Keeni, gkeeni@mumbai.tcs.co.in Arunava Chandra, achandra@tcscal.co.in	5	Jan 2000	Ron Radice
Tata Consultancy Services, Global Engineering Development Center, Chennai	**India**	Gargi Keeni, gkeeni@mumbai.tcs.co.in M. Mala, mala@wst03.tata.ge.com	5	July 2000	John Harding
Tata Consultancy Services, Gurgaon II, New Delhi	**India**	Gargi Keeni, gkeeni@mumbai.tcs.co.in	5	Feb 2001	Ron Radice
Tata Consultancy Services, HP Centre, Chennai	**India**	Gargi Keeni, gkeeni@mumbai.tcs.co.in P. Vasu, pvasu@hp.india.com	5	July 1999	Ron Radice
Tata Consultancy Services, Hyderabad	**India**	Gargi Keeni, gkeeni@mumbai.tcs.co.in N V Jayaramakrishna, jayaram@hydbad.tcs.co.in	5	May 2000	John Harding Gargi Keeni
Tata Consultancy Services, Lucknow	**India**	Gargi Keeni, gkeeni@mumbai.tcs.co.in Nirmal Kumar, nirmal_kumar@lko.tcs.co.in	5	Jan 2000	John Harding Radhika Sokhi
Tata Consultancy Services, SEEPZ, Mumbai	**India**	Gargi Keeni, gkeeni@mumbai.tcs.co.in P. Suresh, p.suresh@seepz.tcs.co.in	5	Aug 1999	Ron Radice
Tata Consultancy Services, Shollinganallur, Chennai	**India**	Gargi Keeni, gkeeni@mumbai.tcs.co.in R. Ravishankar, rravisha@chennai.tcs.co.in	5	Nov 1999	Ron Radice
Tata Consultancy Services, US West, Chennai	**India**	Gargi Keeni, gkeeni@mumbai.tcs.co.in R. Umasankar, rumasan@uswest.com	5	April 1999	Ron Radice V. Muralidharan John Harding
Tata Elxsi Limited, Bangalore	**India**	M. Thangarajan, mtr@teil.soft.net	4	Aug 1999	Pradeep Udhas
Telcordia Technologies, Inc., Morristown, NJ	**USA**	Bill Pitterman, wpitterm@telcordia.com Ivan Handojo, Ihandojo@telcordia.com	5	May 1999	Pat O'Toole Bill Curtis Norm Hammock
U.S. Air Force, Ogden Air Logistics Center, Technology & Industrial Support Directorate, Software Engineering Division, Hill AFB, UT	**USA**	Jim Vanfleet, Jim.Vanfleet@Hill.af.mil	5	July 1998	Mark Paulk Brian Larman Donna Dunaway Bonnie Bollinger Millie Sapp Mike Ballard

Name of Organization	Country	Contact	Level	Date Assessed	Lead Assessor(s)
U.S. Air Force, Oklahoma City Air Logistics Center, Directorate of Aircraft Management, Software Division, Test Software and Industrial Automation Branches (OC-ALC/LAS), Tinker AFB, OK	**USA**	Kelley Butler, kelley.butler@tinker.af.mil	4	Nov 1996	Judah Mogilensky
U.S. Army Aviation & Missile Command, Software Engineering Directorate, Redstone Arsenal, Alabama, AL	**USA**	Jacquelyn Langhout, jackie.langhout@sed.redstone.army.mil	4	April 2000	David Zubrow
U.S. Army, Communications and Electronics Command (CECOM), Software Engineering Center (SEC), Fire Support Software Engineering (Telos), Fort Sill, OK	**USA**	Don Couch, couchdc@fssec.army.mil Phil Sperling, sperlips@fssec.army.mil	4	Nov 1997	Don Couch David Zubrow
U.S. Navy, F/A-18 Software Development Task Team (SWDTT), Naval Air Warfare Center Weapons Division (NAWCWD), China Lake, CA	**USA**	Claire Velicer, velicercm@navair.navy.mil	4	Feb 2000	Tim Olson Ralph Williams
U.S. Navy, Fleet Material Support Office, Mechanicsburg, PA	**USA**	Kathleen D. Chastain, kathleen_chastain@fmso.navy.mil	4	Oct 1998	John Smith Ann Roberts
United Space Alliance, Space Shuttle Onboard Software Project, Houston, TX	**USA**	Julie Barnard, julie.r.barnard@usahq.unitedspacealliance.com	5	Nov 1989	*(SCE -- before LA and LE programs)* *(Donald Sova)*
US Technology Resources India (US Software Pvt Ltd.), Trivandrum	**India**	Arun Narayanan, arun_narayan@usswi.com Sunil Balakrishnan, sunil_bala@usswi.com	5	Dec 2001	Anamika Chakravarty
Wipro GE Medical Systems, Bangalore	**India**	K. Puhazhendi, k.puhazhendi@geind.ge.com	5	Jan 1999	Richard Knudson C. Rama Rao
Wipro Technologies, Enterprise Solutions Division, Bangalore	**India**	T. V. Subbarao, subbarao.tangirala@wipro.com	5	Dec 1998	Richard Storch
Wipro Technologies, Global R & D (formerly Technology Solutions), Bangalore	**India**	V. Subramanyam, vsm@wipinfo.soft.net	5	June 1999	Richard Knudson Mark Paulk
Zensar Technologies Limited (formerly International Computers India Limited), Pune	**India**	Ashok Sontakke, a.r.sontakke@icil.co.in	5	Feb 1999	Richard Knudson

A Software Process Bibliography

Updated October 2002

This bibliography was developed for people interested in learning more about software process management. It is neither authoritative nor exhaustive and should not be construed as an endorsement for any of the books and papers that may be referenced. In some cases, papers listed present diametrically opposed perspectives and are included to provide a balanced view of the issues. Some papers take radical stances that may inspire useful discussion.

Please send suggestions and corrections to:

Mark C. Paulk
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
Email: mcp@sei.cmu.edu

The SEI's Web page is http://www.sei.cmu.edu/

The SEI's Capability Maturity Model® for Software (CMM®) Web page is
http://www.sei.cmu.edu/cmm/cmm.html

The SEI's CMM Integration Web page is
http://www.sei.cmu.edu/cmm/cmms/cmms.integration.html

Many of my papers are on-line at http://www.sei.cmu.edu/cmm/cmm.articles.html. This bibliograpy is online at http://www.sei.cmu.edu/activities/cmm/docs/biblio.html.

Web resources that you may find of interest (sources of many of the papers below) include

- ASQ Software Division — http://www.asq.org/sd/swqweb.html
- Crosstalk: The Journal of Defense Software Engineering — http://www.stsc.hill.af.mil/crosstalk/crostalk.html
- IEEE TCSE Software Process Newsletter — http://www-se.cs.mcgill.ca/process/spn.html
- Maturity Profile Presentation — http://www.sei.cmu.edu/sema/profile.html
- Maturity Questionnaire — http://www.sei.cmu.edu/publications/documents/94.reports/94.sr.007.html
- Practical Software Measurement — http://www.psmsc.com/
- Project Management Institute — http://www.pmi.org/
- Published Maturity Levels — http://www.sei.cmu.edu/sema/pub_ml.html
- Software Process Improvement Networks (SPINs) — http://www.sei.cmu.edu/collaborating/spins/spins.html
- Software Program Managers Network — http://spmn.com/

There are also a number of useful slide presentations at
http://www.sei.cmu.edu/activities/cmm/slides/slides.html

GENERAL MANAGEMENT, PROCESS, AND QUALITY

General Management Topics

Clayton M. Christensen, **The Innovator's Dilemma**, Harvard Business School Press, Cambridge, MA, 1997.

James C. Collins and Jerry I. Porras, **Built to Last**, HarperCollins Publishers, New York, NY, 1994.

Jim Collins, **Good to Great**, HarperCollins Publishers, New York, NY, 2001.

Eliyahu M. Goldratt, **Critical Chain**, North River Press, Great Barrington, MA, 1997.

Ann Langley, "Between 'Paralysis by Analysis' and 'Extinction by Instinct'," Sloan Management Review, Vol. 36, No. 3, Spring 1995, pp. 63-76.

John Micklethwait and Adrian Wooldridge, **The Witch Doctors: Making Sense of the Management Gurus**, Times Books, New York, NY, 1996.

Aaron J. Shenhar, Ofer Levy, and Dov Dvir, "Mapping the Dimensions of Project Success," Project Management Journal, Project Management Institute, Vol. 28, No. 2, June 1997, pp. 5-13.

Robert Simons, "Control in an Age of Empowerment," Harvard Business Review, March-April 1995, pp. 80-88.

Michael Treacy and Fred Wiersema, **The Discipline of Market Leaders**, Addison-Wesley, Reading, MA, 1997.

Karl E. Weick and Kathleen M. Sutcliffe, **Managing the Unexpected: Assuring High Performance in an Age of Complexity**, Jossey-Bass, San Francisco, CA, 2001.

Margaret J. Wheatley, **Leadership and the New Science**, Berrett-Koehler Publishers, San Francisco, CA, 1992.

Total Quality Management

Peter S. Bottcher and Robert W. Stoddard, "How Does Software Six Sigma Relate to the SEI CMM?" **Proceedings of the 7th International Conference on Software Quality**, Montgomery, Alabama, 6-8 October 1997, pp. 34-53.

P.B. Crosby, **Quality is Free**, McGraw-Hill, New York, NY, 1979.

W. Edwards Deming, **Out of the Crisis**, MIT Center for Advanced Engineering Study, Cambridge, MA, 1986.

W. Edwards Deming, **The New Economics for Industry, Government, Education, Second Edition**, MIT Center for Advanced Educational Services, Cambridge, MA, 1994.

David A. Garvin, “Competing on the Eight Dimensions of Quality,” Harvard Business Review, November-December 1987. Reprinted in IEEE Engineering Management Review, Vol. 24, No. 1, Spring 1996, pp. 15-23.

Michael Hammer and James Champy, **Reengineering the Corporation: A Manifesto for Business Revolution**, HarperCollins, New York, New York, 1993.

Mikel Harry and Richard Schroeder, **Six Sigma: The Breakthrough Management Strategy Revolutionizing the World’s Top Corporations**, Doubleday, New York, NY, 2000.

J.M. Juran, **Juran on Planning for Quality**, Macmillan, New York, NY, 1988.

Imai Masaaki, **Kaizen: The Key to Japan’s Competitive Success**, McGraw-Hill, New York, NY, 1986.

Daniel Niven, “When Times Get Tough, What Happens to TQM?” Harvard Business Review, May-June 1993, pp. 20-34.

Nelson P. Repenning and John D. Sterman, “Nobody Ever Gets Credit for Fixing Problems that Never Happened: Creating and Sustaining Process Improvement,” California Management Review, Vol. 43, No. 4, Summer 2001, pp. 64-88.

Peter M. Senge, **The Fifth Discipline: The Art & Practice of the Learning Organization**, Doubleday/Currency, New York, NY, 1990.

ISO 9000 Series

Tomoo Matsubara, “Does ISO 9000 Really Help Improve Software Quality?,” American Programmer, Vol. 7, No. 2, February 1994, pp. 38-45.

Mark C. Paulk, “How ISO 9001 Compares With the CMM,” IEEE Software, Vol. 12, No. 1, January 1995, pp. 74-83.

D. Stelzer, W. Mellis, and G. Herzwurm, “Software Process Improvement via ISO 9000? Results of Two Surveys Among European Software Houses,” Software Process: Improvement and Practice, Vol. 2, Issue 3, September 1996, pp. 197-210.

“TickIT: A Guide to Software Quality Management System Construction and Certification Using EN29001, Issue 2.0,” U.K. Department of Trade and Industry and the British Computer Society, 28 February 1992.

SOFTWARE ENGINEERING

General Software Engineering

Frederick P. Brooks, Jr, **The Mythical Man-Month: Essays on Software Engineering Anniversary Edition,** Addison-Wesley, Reading, MA, 1995.

Frederick Brooks, Jr, "No Silver Bullet: Essence and Accidents of Software Engineering," IEEE Computer, Vol. 20, No. 4, April 1987, pp. 10-19.

Tom DeMarco, **Why Does Software Cost So Much?**, Dorset House, New York, NY, 1995.

Norman Fenton, Shari Lawrence Pfleeger, and Robert Glass, "Science and Substance: A Challenge to Software Engineers," IEEE Software, Vol. 11, No. 4, July 1994, pp. 86-95.

Capers Jones, **Assessment and Control of Software Risks**, PTR Prentice-Hall, Inc., Englewood Cliffs, NJ, 1994.

Mary Shaw, "Prospects for an Engineering Discipline of Software," IEEE Software, Vol. 7, No. 6, November 1990, pp. 15-24.

Software Problems

"Report of the Defense Science Board Task Force on Military Software," Department of Defense, Office of the Under Secretary of Defense for Acquisition, Washington, D.C., September 1987.

Robert N. Charette, "No One Could Have Done Better," American Programmer, July 1995, pp. 21-28.

Jim Johnson, "Chaos: The Dollar Drain of IT Project Failures," Application Development Trends, January 1995, pp. 41-47.

Cem Kaner, "Legal Issues Related to Software Quality," Software Quality, ASQ Software Division, No. 2, 1997-98, pp. 1-10.

J.H. Paul and G.C. Simon, "Bugs in the Program: Problems in Federal Government Computer Software Development and Regulation," Staff Study for the House Committee on Science, Space, and Technology, September 1989.

Measurement

Robert D. Austin, **Measuring and Managing Performance in Organizations**, Dorset House Publishing, New York, NY, 1996.

Anita D. Carleton, Robert E. Park, et al., “Software Measurement for DoD Systems: Recommendations for Initial Core Measures,” Software Engineering Institute, Carnegie Mellon University, CMU/SEI-92-TR-19, 1992.

Michael K. Daskalantonakis, “A Practical View of Software Measurement and Implementation Experience With Motorola,” IEEE Transactions on Software Engineering, Vol. 18, No. 11, November 1992, pp. 998-1010.

Norman Fenton, “Software Measurement: A Necessary Scientific Basis,” IEEE Transactions on Software Engineering, Vol. 20, No. 3, March 1994, pp. 199-206.

Robert B. Grady, **Practical Software Metrics For Project Management and Process Improvement**, Prentice Hall, Englewood Cliffs, NJ, May 1992.

Capers Jones, **Applied Software Measurement, 2nd Edition**, McGraw Hill, New York, NY, 1997.

Shari L. Pfleeger, “Lessons Learned in Building a Corporate Metrics Program,” IEEE Software, May 1993, pp. 67-74.

Lawrence H. Putnam and Ware Myers, **Industrial Strength Software: Effective Management Using Measurement**, IEEE Computer Society Press, Los Alamitos, CA, 1997.

Edward R. Tufte, **The Visual Display of Quantitative Information**, Graphics Press, Cheswick, CT, 1983.

Gerald M. Weinberg, **Quality Software Management Vol. 2: First-Order Measurement**, Dorset House Publishing, New York, New York, 1993.

SOFTWARE PROCESS

General Software Process

R.D. Austin and D.J. Paulish, “A Survey of Commonly Applied Methods for Software Process Improvement,” Software Engineering Institute, Carnegie Mellon University, CMU/SEI-93-TR-27, 1993.

James Bach, “The Challenge of ‘Good Enough’ Software,” American Programmer, Vol. 8, No. 10, October 1995, pp. 2-11.

Kim Caputo, **CMM Implementation Guide: Choreographing Software Process Improvement**, Addison-Wesley, Reading, MA, April 1998.

Bill Curtis, Herb Krasner, and N. Iscoe, "A Field Study of the Software Design Process for Large Systems," Communications of the ACM, Vol. 31, No. 11, November 1988, pp. 1268-1287.

Khaled El Emam and Nazim H. Madhavji (editors), **Elements of Software Process Assessment and Improvement**, IEEE Computer Society Press, Los Alamitos, CA, 1999.

Robert B. Grady, **Successful Software Process Improvement**, Prentice Hall, Englewood Cliffs, NJ, May 1997.

Watts S. Humphrey, **A Discipline for Software Engineering**, Addison-Wesley Publishing Company, Reading, MA, 1995.

Watts S. Humphrey, **Introduction to the Team Software Process**, Addison-Wesley Longman Inc, Reading, MA, 1999.

Robin B. Hunter and Richard H. Thayer (editors), **Software Process Improvement**, IEEE Computer Society Press, Los Alamitos, CA, 2001.

Craig Kaplan, Ralph Clark, and Victor Tang, **Secrets of Software Quality**, McGraw-Hill, New York, NY, 1995.

Steve Maguire, **Debugging the Development Process**, Microsoft Press, Redmond, WA, 1994.

Steve McConnell, **Rapid Development: Taming Wild Software Schedules**, Microsoft Press, Redmond, WA, 1996.

Mark C. Paulk, "Software Process Proverbs," Crosstalk: The Journal of Defense Software Engineering, Vol. 10, No. 1, January 1997, pp. 4-7.

Karl Wiegers, **Creating a Software Engineering Culture**, Dorset House Publishing, New York, NY, 1996.

Laurie Ann Williams, "The Collaborative Software Process," PhD Dissertation, University of Utah, August 2000.

Agile Methodologies

Richard Baskerville, Linda Levine, et al., "How Internet Software Companies Negotiate Quality," IEEE Computer, Vol. 34, No. 5, May 2001, pp. 51-57.

Kent Beck, **Extreme Programming Explained: Embrace Change**, Addison-Wesley, Reading, MA, 1999.

Alistair Cockburn, **Agile Software Development**, Addison-Wesley, Boston, MA, 2002.

James A. Highsmith, **Adaptive Software Development: A Collaborative Approach to Managing Complex Systems**, Dorset House, New York, NY, 2000.

Alan MacCormack, “Product-Development Practices That Work: How Internet Companies Build Software,” MIT Sloan Management Review, Vol. 42, No. 2, Winter 2001, pp. 75-84.

Michele Marchesi, Giancarlo Succi, Don Wells, and Laurie Williams (ed), **Extreme Programming Perspectives**, Addison-Wesley, Boston, MA, 2002.

Mark C. Paulk, “Extreme Programming from a CMM Perspective,” IEEE Software, Vol. 18, No. 6, November/December 2001, pp. 19-26.

Ken Schwaber and Mike Beedle, **Agile Software Development with Scrum**, Prentice Hall, Upper Saddle River, NJ, 2002.

Stanley A. Smith and Michael A. Cusumano, “Beyond the Software Factory: A Comparison of ‘Classic’ and PC Software Developers,” Massachusetts Institute of Technology, Sloan School WP#3607-93\BPS, 1 September 1993.

Laurie Williams and Robert Kessler, **Pair Programming Illuminated**, Addison-Wesley, Boston, MA, 2002.

Capability Maturity Model for Software

Kenneth M. Dymond, **A Guide to the CMM**, Process Inc US, Annapolis, MD, 1995.

Mark Ginsberg and Lauren Quinn, “Process Tailoring and the Software Capability Maturity Model,” Software Engineering Institute, Carnegie Mellon University, CMU/SEI-94-TR-024, November 1995.

Watts S. Humphrey, **Managing the Software Process**, Addison-Wesley, Reading, MA, 1989.

Donna L. Johnson and Judith G. Brodman, “Tailoring the CMM for Small Businesses, Small Organizations, and Small Projects,” Software Process Newsletter, IEEE Computer Society Technical Council on Software Engineering, No. 8, Winter 1997, p. 1-6.

Mark C. Paulk, Charles V. Weber, Bill Curtis, and Mary Beth Chrissis, **The Capability Maturity Model: Guidelines for Improving the Software Process**, Addison-Wesley Publishing Company, Reading, MA, 1995.

Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, and Charles V. Weber, “Capability Maturity Model, Version 1.1,” IEEE Software, Vol. 10, No. 4, July 1993, pp. 18-27.

Mark C. Paulk, “The Evolution of the SEI’s Capability Maturity Model for Software,” Software Process: Improvement and Practice, Vol. 1, No. 1, Spring 1995.

Mark C. Paulk, “Using the Software CMM With Good Judgment,” ASQ Software Quality Professional, Vol. 1, No. 3, June 1999, pp. 19-29.

Software Process Assessment

"Appraisal Requirements for CMMI Version 1.1 (ARC)," Carnegie Mellon University, Software Engineering Institute, CMU/SEI-2001-TR-024, December 2001.

Donna K. Dunaway and Steve M. Masters, “CMM-Based Appraisal for Internal Process Improvement (CBA IPI): Method Description,” Software Engineering Institute, Carnegie Mellon University, CMU/SEI-96-TR-007, 1996.

Khaled El Emam and Dennis R. Goldenson, " An Empirical Review of Software Process Assessments," National Research Council of Canada, Institute for Information Technology, November 1999.

Mark C. Paulk, Watts S. Humphrey, and George J. Pandelios, “Software Process Assessments: Issues and Lessons Learned,” **Proceedings of ISQE92**, Juran Institute, 10-11 March 1992, pp. 4B/41-4B/58.

Software Capability Evaluation

Joseph J. Besselman, Paul Byrnes, Cathy J. Lin, Mark C. Paulk, and Rajesh Puranik, “Software Capability Evaluations: Experiences from the Field,” **SEI Technical Review ‘93**, 1993.

Paul Byrnes and Mike Phillips, “Software Capability Evaluation Version 3.0 Method Description,” Software Engineering Institue, Carnegie Mellon University, CMU/SEI-96-TR-002, 1996.

Maj. George A. Newberry, “The Relationship Between the SEI’s CMM Levels and Source Selection,” Crosstalk: The Journal of Defense Software Engineering, Vol. 9, No. 5, May 1996, p. 6.

David Rugg, “Using a Capability Evaluation to Select a Contractor,” IEEE Software, Vol. 10, No. 4, July 1993, pp. 36-45.

SEI Criticism

James Bach, “Enough About Process: What We Need Are Heroes,” IEEE Software, Vol. 12, No. 2, February 1995, pp. 96-98.

James Bach, “The Immaturity of the CMM,” American Programmer, Vol. 7, No. 9, September 1994, pp. 13-18.

Bill Curtis, “A Mature View of the CMM,” American Programmer, Vol. 7, No. 9, September 1994, pp. 19-28.

T. Bollinger and C. McGowan, “A Critical Look at Software Capability Evaluations,” IEEE Software, Vol. 8, No. 4, July 1991, pp. 25-41.

Watts S. Humphrey and Bill Curtis, “Comments on ‘A Critical Look’,” IEEE Software, Vol. 8, No. 4, July 1991, pp. 42-46.

Capers Jones, Chapter 5, “Artificial Maturity Levels,” in **Assessment and Control of Software Risks**, PTR Prentice-Hall, Inc., Englewood Cliffs, NJ, 1994, pp. 63-70. See also pp. 19-20 and 23-26.

Capers Jones, “The SEI’s CMM—Flawed?,” Software Development, Vol. 3, No. 3, March 1995, pp. 41-48.

Hossein Saiedian and Richard Kuzara, “SEI Capability Maturity Model’s Impact on Contractors,” IEEE Computer, Vol. 28, No. 1, January 1995, pp. 16-26.

ISO 15504 (SPICE)

Alec Dorling, “Software Process Improvement and Capability dEtermination,” Software Quality Journal, Vol. 2, No. 4, December 1993, pp. 209-224.

David H. Kitson, “An Emerging International Standard for Software Process Assessment,” **Proceedings of the Third IEEE International Software Engineering Standards Symposium and Forum**, Walnut Creek, CA, 1-6 June 1997, pp. 83-90.

Various Appraisal Methods and Models

Sarah A. Sheard, “The Frameworks Quagmire,” Crosstalk: The Journal of Defense Software Engineering, Vol. 10, No. 9, September 1997.

H.E. Thomson and P. Mayhew, “Approaches to Software Process Improvement,” Software Process: Improvement and Practice, Vol. 3, Issue 1, March 1997, pp. 3-17.

Case Studies of Software Process Improvement

C. Billings, J. Clifton, B. Kolkhorst, E. Lee, and W.B. Wingert, “Journey to a Mature Software Process,” IBM Systems Journal, Vol. 33, No. 1, 1994, pp. 46-61.

Kelley Butler and Walter Lipke, “Software Process Achievement at Tinker Air Force Base,” Carnegie Mellon University, Software Engineering Institute, CMU/SEI-2000-TR-014, September 2000.

Bradford K. Clark, “The Effects of Software Process Maturity on Software Development Effort,” PhD Dissertation, Computer Science Department, University of Southern California, August 1997.

Michael Diaz and Joseph Sligo, “How Software Process Improvement Helped Motorola,” IEEE Software, Vol. 14, No. 5, September/October 1997, pp. 75-81.

Pat Ferguson, Gloria Leman, Prasad Perini, Susan Renner, and Girish Seshagiri, “Software Process Improvement Works!” Carnegie Mellon University, Software Engineering Institute, CMU/SEI-99-TR-027, November 1999.

Thomas J. Haley, “Raytheon’s Experience in Software Process Improvement,” IEEE Software, Vol. 13, No. 6, November 1996, pp. 33-41.

Donald E. Harter, Mayuram S. Krishnan, and Sandra A. Slaughter, “Effects of Process Maturity on Quality, Cycle Time, and Effort in Software Product Development,” Management Science, Vol. 46, No. 4, April 2000, pp. 451-466.

Will Hayes and James W. Over, “The Personal Software Process (PSP): An Empirical Study of the Impact of PSP on Individual Engineers,” Software Engineering Institute, Carnegie Mellon University, CMU/SEI-97-TR-001, December 1997.

James Herbsleb, David Zubrow, Dennis Goldenson, Will Hayes, and Mark Paulk, “Software Quality and the Capability Maturity Model,” Communications of the ACM, Vol. 40, No. 6, June 1997, pp. 30-40.

Declan P. Kelly and Bill Culleton, “Process Improvement for Small Organizations,” IEEE Computer, Vol. 32, No. 10, October 1999, pp. 41-47.

Herb Krasner, “Accumulating the Evidence for the Payoff of Software Process Improvement – 1997,” http://www.utexas.edu/coe/sqi/archive/krasner/spi.pdf.

Patricia K. Lawlis, Robert M. Flowe, and James B. Thordahl, “A Correlational Study of the CMM and Software Development Performance,” Crosstalk: The Journal of Defense Software Engineering, Vol. 8, No. 9, September 1995, pp. 21-25.

Frank McGarry, Rose Pajerski, et al, “Software Process Improvement in the NASA Software Engineering Laboratory,” Software Engineering Institute, Carnegie Mellon University, CMU/SEI-94-TR-22, December 1994.

Thomas McGibbon, "A Business Case for Software Process Improvement Revised," Data and Analysis Center for Software, 30 Sept 1999.

Jeff Nowell, “An Analysis of Software Process Improvement,” *[performed at the Oklahoma City Air Logistics Center, Directorate of Aircraft, Software Division (LAS), Tinker AFB, Oklahoma; prepared for the Deputy Assistant Secretary of the Air Force, Communications, Computers, and Support Systems (SAF/AQK), Washington, DC]*, Software Productivity Research, Burlington, MA, 26 September 1994.

Sandra A. Slaughter, Donald E. Harter, and Mayuram S. Krishnan, “Evaluating the Cost of Software Quality,” Communications of the ACM, 41(8) 67-73, August 1998. Reprinted in Engineering Management Review, 26(4) 32-37, Winter 1998.

R.R. Willis, R.M. Rova, et al, “Hughes Aircraft’s Widespread Deployment of a Continuously Improving Software Process,” Software Engineering Institute, Carnegie Mellon University, CMU/SEI-98-TR-006, May 1998.

Harvey Wohlwend and Susan Rosenbaum, “Schlumberger’s Software Improvement Program,” IEEE Transactions on Software Engineering, Vol. 20, No. 11, November 1994, pp. 833-839.

George Yamamura and Gary B. Wigle, “SEI CMM Level 5: For the Right Reasons,” Crosstalk: The Journal of Defense Software Engineering, Vol. 10, No. 8, August 1997, pp. 3-6.

ORGANIZATIONAL CULTURE AND PSYCHOLOGY

People Issues

Larry L. Constantine, **Constantine on Peopleware**, Yourdon Press Computing Series, Englewood Cliffs, NJ, 1995.

Bill Curtis, William E. Hefley, and Sally A. Miller, **People Capability Maturity Model**, Addison-Wesley Publishing Company, Reading, MA, 2001.

Robyn M. Dawes, **Rational Choice in an Uncertain World**, Harcourt Brace Jovanovich College Publishers, Orlando, FL, 1988.

Tom DeMarco and Timothy Lister, **Peopleware, 2nd Edition**, Dorset House, New York, NY, 1999.

Paul J. DiMaggio and Walter W. Powell, “The Iron Cage Revisited: Institutional Isomorphism and Collective Rationality in Organizational Fields,” American Sociological Review, Vol. 48, April 1983, pp. 147-160.

Roger Fisher, William Ury, and Bruce Patton, **Getting to Yes: Negotiating Agreement Without Giving In, Second Edition**, Penguin, 1991.

Robert L. Glass, **Software Creativity**, Prentice Hall, Englewood Cliffs, NJ, 1995.

Watts S. Humphrey, **Managing Technical People**, Addison-Wesley Publishing Company, Reading, MA, 1997.

Thomas Teal, "The Human Side of Management," Harvard Business Review, November/December 1996, pp. 35-44.

Robert Townsend, **Up the Organization: How to Stop the Corporation from Stifling People and Strangling Profit**s, Fawcett Crest, New York, NY, 1970.

William Ury, **Getting Past No : Negotiating Your Way from Confrontation to Cooperation**, Bantam Doubleday, 1993.

Gerald M. Weinberg, **The Psychology of Computer Programming**, Van Nostrand Reinhold, New York, NY, 1971.

Gerald M. Weinberg, **Quality Software Management Volume 3: Congruent Action**, Dorset House Publishing, New York, New York, 1994.

Organizational Culture and Teams (IC)

Karen Bemowski, "What Makes American Teams Tick?" ASQC Quality Progress, Vol. 28, No. 1, January 1995, pp. 39-43.

Charles Handy, **Gods of Management: The Changing Work of Organizations – Third Edition**, Oxford University Press, New York, New York, 1991.

Jon R. Katzenbach and Douglas K. Smith, **The Wisdom of Teams**, HarperCollins, New York, NY, 1993.

Robert Pool, "When Failure Is Not An Option," MIT Technology Review, July 1997. Reprinted in IEEE Engineering Management Review, Vol. 27, No. 1, Spring 1999, pp. 27-31.

Peter R. Scholtes, Brian L. Joiner, and Barbara J. Streibel, **The TEAM Handbook, Second Edition**, Oriel Incorporated, Madison, WI, 1996.

Michael Schrage, **No More Teams! Mastering the Dynamics of Creative Collaboration**, Currency Doubleday, New York, NY, 1989.

MATURITY LEVEL 2 - REPEATABLE

Requirements Management (RM)

Barry Boehm and Hoh In, “Cost vs. Quality Requirements: Conflict Analysis and Negotiation Aids,” ASQ Software Quality Professional, Vol. 1, No. 2, March 1999, pp. 38-50.

Herb Krasner, “Requirements Dynamics in Large Software Projects,” **Proceedings of the 11th World Computer Congress (IFIP89)**, Elsevier Science Publishers B.V., Amsterdam, The Netherlands, August 1989.

Gerald M. Weinberg, “Requirements as the Foundation of Measurement,” Chapter 19 in **Quality Software Management Vol. 2: First-Order Measurement**, Dorset House Publishing, New York, New York, 1993, pp. 295-306.

(Software) Project Planning (SPP)

Tarek K. Abdel-Hamid and Stuart E. Madnick, “Impact of Schedule Estimation on Software Project Behavior,” IEEE Software, Vol. 3, No. 4, July 1986, pp. 70-75.

Barry W.Boehm, Ellis Horowitz, et al, **Software Cost Estimation with COCOMO II**, Prentice Hall, Upper Saddle River, NJ, 2000.

Alan M. Davis, “Software Life Cycle Models,” **Software Engineering Project Management, Second Edition**, R.H. Thayer (ed), IEEE Computer Society Press, Los Alamitos, CA, 1997, pp. 105-114.

J. Hihn and H. Habib-Agahi, “Cost Estimation of Software Intensive Projects: A Survey of Current Practices,” **Proceedings of the 13th International Conference on Software Engineering**, Austin, TX, 13-17 May 1991, pp. 276-287.

Albert L. Lederer and Jayesh Prasad, “Nine Management Guidelines for Better Cost Estimating,” Communications of the ACM, Vol. 35, No. 2, February 1992, pp. 51-59.

Steve McConnell, "The Nine Deadly Sins of Project Planning," IEEE Software, September/October 2001, pp. 5-7.

Robert E. Park, “A Manager’s Checklist for Validating Software Cost and Schedule Estimates”, American Programmer, Vol. 9, No. 6, June 1996, pp. 30-35.

(Software) Project Management (PTO, ISM)

Tarek Abdel-Hamid and Stuart E. Madnick, **Software Project Dynamics**, Prentice-Hall, Englewood Cliffs, NJ, 1991.

David I. Cleland, **Project Management: Strategic Design and Implementation, Second Edition**, McGraw-Hill, New York, NY, 1994.

Larry L. Constantine, **Beyond Chaos: The Expert Edge in Managing Software Development**, Addison-Wesley, Boston, MA, 2001.

Kenneth G. Cooper, "The Rework Cycle: Vital Insights into Managing Projects," IEEE Engineering Management Review, Fall 1993, pp. 4-12.

Kenneth G. Cooper, "The $2,000 Hour," IEEE Engineering Management Review, Vol. 22, No. 4, Winter 1994, pp. 12-23.

Jane T. Lochner, "16 Critical Software Practices for Performance-Based Management," Crosstalk: The Journal of Defense Software Engineering, Vol. 12, No. 10, October 1999, pp. 6-9

Tom DeMarco, **Controlling Software Projects**, Yourdon Press, New York, NY, 1982.

Tom Gilb, **Principles of Software Engineering Management**, Addison-Wesley, Reading, MA, 1988.

Neil C. Olsen, "The Software Rush Hour," IEEE Software, Vol. 10, No. 5, September 1993, pp. 29-37.

George Stark, Robert C. Durst, and C.W. Vowell, "Using Metrics in Management Decision Making," IEEE Computer, Vol. 27, No. 9, September 1994, pp. 42-49.

H.J. Thamhain and D.L. Wilemon, "Criteria for Controlling Projects According to Plan," Project Management Journal, June 1986, pp. 75-81. Reprinted in **Software Engineering Project Management**, R.H. Thayer (ed), IEEE Computer Society Press, 1988, pp. 392-398.

N. Whitten, **Managing Software Development Projects, 2nd Edition**, John Wiley and Sons, New York, NY, 1995.

Stuart Woodward, "Evolutionary Project Management," IEEE Computer, Vol. 32, No. 10, October 1999, pp. 49-57.

Customer-Supplier Relationship (SSM, Acquisition, Customer Satisfaction)

Thomas O. Jones and W. Earl Sasser, Jr., “Why Satisfied Customers Defect,” Harvard Business Review, November/December 1995. Reprinted in IEEE Engineering Management Review, Fall 1998, Vol. 26, No. 3, pp. 16-26.

B. Craig Meyers and Patricia Oberndorf, **Managing Software Acquisition: Open Systems and COTS Products**, Addison-Wesley, Reading, MA, 2001.

Jim Nielsen and Ann Miller, “Selecting Software Subcontractors,” IEEE Software, Vol. 13, No. 4, July 1996, pp. 104-109.

Software Quality Assurance (SQA)

M.A. Aquino, “Improvement vs. Compliance: A New Look at Auditing,” ASQC Quality Progress, October 1990, pp. 47-49.

F.J. Buckley and R. Poston, “Software Quality Assurance,” IEEE Transactions on Software Engineering, Vol. SE-10, No. 1, January 1984, pp. 36-41.

F. J. Buckley, “The Roles of a SQA Person,” ACM Software Engineering Notes, Vol. 12, No. 3, July 1987, pp. 42-44.

Rushby Craig, "Software Quality Assurance in a CMM Level 5 Organization," Crosstalk: The Journal of Defense Software Engineering, May 1999, pp. 11-15.

Software Configuration Management (SCM)

E.H. Bersoff, “Elements of Software Configuration Management,” IEEE Transactions on Software Engineering, January 1984, pp. 27-35. Reprinted in **Software Engineering Project Management**, R.H. Thayer (ed), IEEE Computer Society Press, 1988, pp. 430-438.

E.H. Bersoff and A.M. Davis, “Impacts of Life Cycle Models on Software Configuration Management,” Communications of the ACM, Vol. 34, No. 8, August 1991, pp. 105-118.

Susan A. Dart, “The Past, Present, and Future of Configuration Management,” Software Engineering Institute, Carnegie Mellon University, CMU/SEI-92-TR-8, 1992.

Peter H. Feiler, “Configuration Management Models in Commercial Environment,” Software Engineering Institute, Carnegie Mellon University, CMU/SEI-91-TR-7, 1991.

MATURITY LEVEL 3 - DEFINED

Organizational Process (OPF, OPD)

James W. Armitage, Marc I. Kellner, and Richard W. Phillips, “Software Process Definition Guide: Content of Enactable Software Process Definitions,” Software Engineering Institute, Carnegie Mellon University, CMU/SEI-93-SR-18, August 1993.

Bill Curtis, Herb Krasner, Vincent Shen, and Neil Iscoe, “On Building Software Process Models Under the Lamppost,” **Proceedings of the Ninth International Conference on Software Engineering**, Monterey, CA, IEEE Computer Society, 30 March - 2 April 1987, pp. 96-103.

Bill Curtis, Marc I. Kellner, and Jim Over, “Process Modeling,” Communications of the ACM, Vol. 35, No. 9, September 1992, pp. 75-90.

A. Dandekar and D.E. Perry, “Barriers to Effective Process Architecture – An Experience Report,” Software Process: Improvement and Practice, Vol. 2, Issue 1, March 1996, pp. 13-20.

Priscilla Fowler and Stan Rifkin, “Software Engineering Process Group Guide,” Software Engineering Institute, Carnegie Mellon University, CMU/SEI-90-TR-24, September 1990.

Marc I. Kellner, Raymond J. Madachy, and David M. Raffo, “Software Process Simulation Modeling: Why? What? How?” The Journal of Systems and Software, Vol. 46, No. 2-3, 15 April 1999, pp. 91-105.

Ray Madachy and Denton Tarbet, “Initial Experiences in Software Process Modeling,” ASQ Software Quality Professional, Vol. 2, No. 3, June 2000, pp. 15-27.

Stan Rifkin, "What I Would Do Differently If I Wrote the SEPG GuideToday," **SEPG Conference 2002**, Phoenix, AZ, 18-21 February 2002.

Training (TP)

Chris Argyris, “Teaching Smart People How to Learn,” Harvard Business Review, May/June 1991, pp. 99-109.

David A. Garvin, “Building a Learning Organization,” Harvard Business Review, July/August 1993, pp. 78-91.

W. Wiggenhorn, “Motorola U: When Training Becomes an Education,” Harvard Business Review, July/August 1990, pp. 71-83.

Risk Management

Peter L. Bernstein, **Against the Gods: The Remarkable Story of Risk**, ISBN 0-471-29563-9, John Wiley & Sons, New York, NY, 1996.

B.W. Boehm (ed), **Software Risk Management**, IEEE Computer Society Press, July 1989.

Robert N. Charette, "Large-Scale Project Management is Risk Management," IEEE Software, Vol. 13, No. 4, July 1996, pp. 110-117.

Barbara Kitchenham and Stephen Linkman, "Estimates, Uncertainty, and Risk," IEEE Software, Vol. 14, No. 3, May/June 1997, pp. 69-74.

Ray C. Williams, Julie A. Walker, and Audrey J. Dorofee, "Putting Risk Management Into Practice," IEEE Software, Vol. 14, No. 3, May/June 1997, pp. 75-82.

Integrated Product and Process Development (Concurrent Engineering)

J.D. Blackburn, G. Hoedemaker, and L.N. Van Wassenhove, "Concurrent Software Engineering: Prospects and Pitfalls," IEEE Transactions on Engineering Management, Vol. 43, No. 2, May 1996, pp. 179-188.

R.P. Smith, "The Historical Roots of Concurrent Engineering Fundamentals," IEEE Transactions on Engineering Management, Vol. 44, No. 1, February 1997, pp. 67-78.

Durward K. Sobek II, Jeffrey K. Liker, and Allen C. Ward, "Another Look at How Toyota Integrates Product Development," Harvard Business Review, July/August 1998. Reprinted in IEEE Engineering Management Review, Vol. 26, No. 4, Winter 1998, pp. 69-78.

Software Engineering (SPE, SQM)

Requirements

D.C. Gause and Gerald M. Weinberg, **Exploring Requirements: Quality Before Design**, Dorset House, New York, NY, 1989.

Ralph R. Young, **Effective Requirements Practices**, Addison-Wesley, Reading, MA, 2001.

Design

Len Bass, Paul Clements, and Rick Kazman, **Software Architecture in Practice**, Addison-Wesley, Reading, MA, 1998.

Paul Clements, Rick Kazman, and Mark Klein, **Evaluating Software Architectures: Methods and Case Studies**, Addison-Wesley, Reading, MA, 2001.

Programming

Watts S. Humphrey, "CASE Planning and the Software Process," Software Engineering Institute, Carnegie Mellon University, CMU/SEI-89-TR-26, May 1989.

Testing

Ilene Burnstein, Ariya Homyen, et al, "A Testing Maturity Model for Software Test Process Assessment and Improvement," ASQ Software Quality Professional, Vol. 1, Issue 4, September 1999, pp. 8-21.

Gregory T. Daich, "Emphasizing Software Test Process Improvement," Crosstalk: The Journal of Defense Software Engineering, Vol. 9, No. 6, June 1996, pp. 20-26.

Peer Reviews (PR)

A.F. Ackerman, L.S. Buchwald, and F.H. Lewski, "Software Inspections: An Effective Verification Process," IEEE Software, Vol. 6, No. 3, May 1989, pp. 31-36.

M.E. Fagan, "Advances in Software Inspections," IEEE Transactions on Software Engineering, Vol. 12, No. 7, July 1986, pp. 744-751. Reprinted in **Software Engineering Project Management**,, R.H. Thayer (ed), IEEE Computer Society Press, 1988, pp. 416-423.

Daniel P. Freedman and Gerald M. Weinberg, **Handbook of Walkthroughs, Inspections, and Technical Reviews, Third Edition**, Dorset House, New York, NY, 1990.

Robert L. Glass, "Inspections - Some Surprising Findings," Communications of the ACM, April 1999, pp. 17-19.

Robert Grady and Tom Van Slack, "Key Lessons in Achieving Widespread Inspection Use," IEEE Software, Vol. 11, No. 4, July 1994, pp. 46-57.

Philip M. Johnson and Danu Tjahjono, "Does Every Inspection Really Need a Meeting?" Empirical Software Engineering, Kluwer Academic Publishers, Vol. 3, No. 1, Boston, MA, 1998, pp. 9-35.

John C. Knight and E. Ann Myers, "An Improved Inspection Technique," Communications of the ACM, Vol. 36, No. 11, November 1993, pp. 51-61.

Vahid Mashayekhi, Janet M. Drake, Wei-Tek Tsai, and John Riedl, "Distributed, Collaborative Software Inspection," IEEE Software, Vol. 10, No. 5, September 1993, pp. 66-75.

David L. Parnas and David M. Weiss, "Active Design Reviews: Principles and Practices," Journal of Systems and Software, Vol. 7, No. 4, December 1987, pp. 259-265.

Ronald A. Radice, **High Quality Low Cost Software Inspections**, Paradoxicon Publishing, Andover, MA, 2002.

G. Russell, "Inspection in Ultralarge-Scale Development," IEEE Software, Vol. 8, No. 1, January 1991, pp. 25-31.

Edward F. Weller, "Lessons from Three Years of Inspection Data," IEEE Software, Vol. 10, No. 5, September 1993, pp. 38-45.

Karl E. Wiegers, **Peer Reviews in Software**, Addison-Wesley, Reading, MA, 2002.

MATURITY LEVEL 4 - MANAGED

Mark C. Paulk, Dennis Goldenson, and David M. White, "The 1999 Survey of High Maturity Organizations," Software Engineering Institute, Carnegie Mellon University, CMU/SEI-2000-SR-002, February 2000.

Mark C. Paulk and Mary Beth Chrissis, "The 2001 High Maturity Workshop," Software Engineering Institute, Carnegie Mellon University, CMU/SEI-2001-SR-014, January 2002.

Statistical Process / Quality Control (QPM, SQM)

George E.P. Box and Soren Bisgaard, "The Scientific Context of Quality Improvement," ASQC Quality Progress, June 1987, pp. 54-61. Reprinted in IEEE Engineering Management Review, Vol. 24, No. 1, Spring 1996, pp. 33-42.

Michael Brassard and Diane Ritter, **The Memory Jogger II**, GOAL/QPC, Methuen, MA, 1994.

Norman Fenton and Martin Neil, "A Critique of Software Defect Prediction Models," IEEE Transactions on Software Engineering, Vol. 25, No. 5, September/October 1999, pp. 675-689.

William A. Florac and Anita D. Carleton, **Measuring the Software Process: Statistical Process Control for Software Process Improvement**, Addison-Wesley, Reading, MA, 1999.

William A. Florac, Anita D. Carleton, and Julie Barnard, "Statistical Process Control: Analyzing a Space Shuttle Onboard Software Process," IEEE Software, Vol. 17, No. 4, July/August 2000, pp. 97-106.

Lynne B. Hare, Roger W. Hoerl, John D. Hromi, and Ronald D. Snee, "The Role of Statistical Thinking in Management," ASQC Quality Progress, Vol. 28, No. 2, February 1995, pp. 53-60.

Dan Houston, "Cost of Software Quality: Justifying Software Process Improvement to Managers," ASQ Software Quality Professional, Vol. 1, No. 2, March 1999, pp. 8-16.

K. Ishikawa, **Guide to Quality Control**, Asian Productivity Organization, Tokyo, Japan, (available from Unipub - Kraus International Publications, White Plains, NY) 1986.

Stephen H. Kan, **Metrics and Models in Software Quality Engineering**, Addison-Wesley, Reading, MA, February 1995.

J.D. Musa and A.F. Ackerman, “Quantifying Software Validation: When to Stop Testing?” IEEE Software, Vol. 6, No. 3, May 1989, pp. 19-27.

J.D. Musa, A. Iannino, and K. Okumoto, **Software Reliability: Measurement, Prediction, Application**, McGraw-Hill, New York, NY, 1987.

M.A. Ould, “CMM and ISO 9001,” Software Process: Improvement and Practice, Vol. 2, Issue 4, December 1996, pp.281-289.

Thomas Pyzdek, “Process Control for Short and Small Runs,” ASQC Quality Progress, Vol. 26, No. 4, April 1993, pp. 51-60.

Edward F. Weller, “Practical Applications of Statistical Process Control,” IEEE Software, Vol. 17, No. 3, May/June 2000, pp. 48-55.

Donald J. Wheeler, **Understanding Variation: The Key to Managing Chaos**, SPC Press, Knoxville, TN, 1993.

Donald J. Wheeler, “Charts Done Right,” Quality Progress, Vol. 27, No. 6, June 1994, pp. 65-68.

Donald J. Wheeler and Sheila R. Poling, **Building Continual Improvement: A Guide for Business**, SPC Press, Knoxville, TN, 1998.

Product Knowledge Management: Domain Engineering, Product Lines, and Reuse

Joe Besselman and Stan Rifkin, “Exploiting the Synergism Between Product Line Focus and Software Maturity,” **Proceedings of the 1995 Acquisition Research Symposium**, Washington, D.C., pp. 95-107.

Paul Clements and Linda Northrop, **Software Product Lines: Practices and Patterns**, Addison-Wesley, Reading, MA, 2001.

Shari L. Pfleeger, “Measuring Reuse: A Cautionary Tale,” IEEE Software, Vol. 13, No. 4, July 1996, pp. 118-127.

Kurt C. Wallnau, Scott A. Hissam, and Robert C. Seacord, **Building Systems from Commercial Components**, Addison-Wesley, Reading, MA, 2001.

MATURITY LEVEL 5 - OPTIMIZING

Defect Prevention (DP)

Inderpal Bhandari, Michael Halliday, et al., “A Case Study of Software Process Improvement During Development,” IEEE Transactions on Software Engineering, Vol. 19, No. 12, December 1993, pp. 1157-1170.

David N. Card, “Learning from Our Mistakes with Defect Causal Analysis,” IEEE Software, Vol. 15, No. 1, January/February 1998, pp. 56-63.

R. Chillarege and I. Bhandari, “Orthogonal Defect Classification – A Concept for In-Process Measurements,” IEEE Software, Vol. 18, No. 11, November 1992, pp. 943-955.

Bonnie Collier, Tom DeMarco, and Peter Fearey, “A Defined Process for Project Postmortem Review,” IEEE Software, Vol. 13, No. 4, July 1996, pp. 65-72.

Julia L. Gale, Jesus R. Tirso, and C. Art Burchfield, “Implementing the Defect Prevention Process in the MVS Interactive Programming Organization,” IBM Systems Journal, Vol. 29, No. 1, 1990, pp. 33-43.

C.L. Jones, “A Process-Integrated Approach to Defect Prevention,” IBM Systems Journal, Vol. 24, No. 2, 1985, pp. 150-167.

Juichirou Kajihara, Goro Amamiya, and Tetsuo Saya, “Learning from Bugs,” IEEE Software, Vol. 10, No. 5, September 1993, pp. 46-54.

R.G. Mays, C.L. Jones, G.J. Holloway, and D.P. Studinski, “Experiences with Defect Prevention,” IBM Systems Journal, Vol. 29, No. 1, 1990, pp. 4-32.

Norman Bridge and Corinne Miller, “Orthogonal Defect Classification Using Defect Data to Improve Software Development,” **Proceedings of the 7th International Conference on Software Quality**, Montgomery, Alabama, 6-8 October 1997, pp. 197-213.

Change Management (TCM, PCM)

Eric Abrahamson, “Change Without Pain,” Harvard Business Review, July-August 2000, pp. 75-79.

Victor R. Basili, Michael K. Daskalantonakis, and Robert H. Yacobellis, “Technology Transfer at Motorola,” IEEE Software, Vol. 11, No. 2, March 1994, pp. 70-76.

M. Beer, R.A. Eisenstat, and B. Spector, “Why Change Programs Don’t Produce Change,” Harvard Business Review, November/December 1990, pp. 158-166.

Roger E. Bohn, “Measuring and Managing Technological Knowledge,” Sloan Management Review, Fall 1994. Reprinted in IEEE Engineering Management Review, Vol. 25, No. 4, Winter 1997, pp. 77-88.

Alan M. Davis, “Why Industry Often Says ‘No Thanks’ to Research,” IEEE Software, November 1992, pp. 97-99.

R. Fichman and C.F. Kemerer, “The Illusory Diffusion of Innovations: An Examination of Assimilation Gaps,” Information Systems Research, Vol. 10, No. 3, September 1999, pp. 255-275.

Barbara Kitchenham, Lesley Pickard, and Shari Lawrence Pfleeger, “Case Studies for Method and Tool Evaluation,” IEEE Software, July 1995, pp. 52-62.

John P. Kotter, “Leading Change: Why Transformation Efforts Fail,” Harvard Business Review, March-April 1995, pp. 59-67.

Geoffrey A. Moore, **Crossing the Chasm**, HarperCollins Publishers, New York, NY, 1991.

E.M. Rogers, **Diffusion of Innovations, Third Edition**, The Free Press, New York, NY, 1983.

R. H. Schaffer and H. A. Thomson, “Successful Change Programs Begin with Results,” Harvard Business Review, January/February 1992, pp. 80-89.

Software CMM® Presentations

This page contains links to a number of slide sets related to the Software CMM and software process improvement (SPI). As is true for all presentations, the discussion by the presenter is important for a complete understanding of the material, but these slides may be useful and are provided with the caveat that the papers and books on the CMM and SPI provide a much fuller picture. The papers on the CMM-related articles Web page are recommended for further exploration of these issues.

Mark C. Paulk, "Investing in Software Process Improvement: An Executive Perspective."

This presentation discusses some of the empirical evidence on the value obtained from investing in software process improvement using the Software Capability Maturity Model (CMM) developed by the Software Engineering Institute (SEI) at Carnegie Mellon University. Business drivers for process improvement and some challenges in organization change are described, along with data on the impact on cost, schedule, and quality of achieving the five maturity levels in the CMM. From an executive perspective, the crucial point is that continual improvement depends on systematically addressing the problems facing the organization -- regardless of the improvement framework selected. This "constancy of purpose" depends on management sponsorship, support, and investment.

Mark C. Paulk, "People Issues: The 'Soft Side' of Software Process Improvement."

"Our greatest asset is our people." This platitude is frequently followed by announcements of layoffs, downsizing, and similar Dilbertesque decisions. If people are the most important single factor in success, how do we incorporate that fact into our process improvement programs? This presentation provides an overview of "people issues" for software engineering, management, and process improvement from an individual, team, and organizational perspective.

Mark C. Paulk, "Considering Statistical Process Control for Software."

The Capability Maturity Model for Software, developed by the Software Engineering Institute at Carnegie Mellon University, is a model for building organizational capability that has been widely adopted in the software community and beyond. The Software CMM is a five-level model that prescribes process improvement priorities for software organizations. Level 4 in the CMM focuses on applying quantitative techniques, particularly statistical techniques, for controlling the software process.

In statistical process control, this means eliminating special causes of variation. Because the software process is not a repetitive manufacturing or service process, the application of statistical process control, specifically control charts, has been challenged by many in the software community. What the CMM has to say about statistical process management is discussed, along with the issues in applying statistical thinking to the software process, prerequisites for applying statistical control, and the specific techniques that should be considered. Examples from real-world software projects illustrate the challenges in stabilizing the software process and applying different statistical techniques.

"Panel: Can Statistical Process Control Be Usefully Applied to Software?," Moderators: Mark Paulk and Anita Carleton, SEI, **The 11th Software Engineering Process Group (SEPG) Conference**, Atlanta, Georgia, 8-11 March 1999.

The SEI has advocated the use of statistical process control techniques, specifically control charts, in recent technical reports, tutorials, and classes, and proposed changes for Software CMM version 2 explicitly supported the use of rigorous statistical techniques at maturity level 4. Little of substance has been published, however, on the use of SPC and control charts in the software industry. Most of what has been published has been conceptual, and many of the reported attempts to use SPC have provided little business value. The argument in favor of SPC lies in 1) its admitted value in other industries, especially manufacturing, and 2) undocumented reports of successful use for various software processes. It seems clear that SPC for software is an emerging technique that, while conceptually valuable, is still "shaking down" as a potentially useful tool for software engineering.

The purpose of this panel is to capture some of the pros and cons from industry use of this fascinating tool, focusing in particular on the prerequisites for successful use of SPC and the business value obtained.

Note: The following slides combine presentations from the USA and European SEPG Conferences. Presenters at the SEPG Conference in Atlanta were Carleton, Keller, Meade, Hirsh, Wigle, Card, and Paulk. Presenters at the European SEPG Conference in Amsterdam were Meade, Burr, Hirsh, Heijstek, Curtis, Barnard, and Paulk.

- Anita Carleton, SEI, "Intro - Can Statistical Process Control Be Usefully Applied to Software?"
- Ted Keller, IBM, "Applying SPC Techniques to Software Development - A Management Perspective."
- Susan Meade, Lockheed Martin, "Lockheed Martin Mission Systems."
- Adrian Burr, consultant, "TBD."
- Barbara Hirsh, Motorola, "A Case Study of Applying SPC."
- Andre Heijstek, Ericsson, "SPC in Ericsson."
- Bill Curtis, TeraQuest Metrics, "TBD."
- Gary Wigle, Boeing, "Quantitative Management in Software Engineering."
- David Card, Software Productivity Consortium, "Making Statistics Work for Software Engineering."
- Julie Barnard, United Space Alliance, "Analyzing a Mature Software Inspection

Process UsingStatistical Process Control."

- Mark Paulk, SEI, "Summary - Can Statistical Process Control Be Usefully Applied to Software?" *- includes European SEPG intro slides*

"Panel: What Justifies a Rating of CMM Level 4?" Moderator: Bill Curtis, **The SEPG 2000 Conference**, Seattle, WA, 23 March 2000.

Fundamental Issue Addressed in the Panel: The fundamental issue being addressed in this panel are concerns about inconsistency in the criteria being used to assess the satisfaction of CMM Level 4 Key Process Area goals.

Increasing numbers of assessments are resulting in Level 4 or 5 ratings. However, many in the CMM community sense considerable inconsistency in the criteria being used to justify these ratings, especially regarding Level 4 Key Process Areas. Openly debated issues include 1) whether an organization is required implement control charts and other classic statistical process control (SPC) techniques, 2) whether there are statistical techniques other than SPC that are more appropriate to software since it is primarily a design rather than a manufacturing activity, 3) whether statistical methods are required at all compared to reasonable use of quantitative data in controlling the process, and 4) what level of data collection and analysis must be implemented to achieve quantitative process and quality control. These questions need to be resolved for the CMM community to achieve consensus on what constitutes a Level 4 capability and what types of practices will justify a level 4 rating from a CBA IPI. The panel, which includes CMM authors and Lead Assessors experienced in high maturity assessments, will discuss and debate the criteria they use in determining whether an organization has achieved a Level 4 process capability.

Panelists were chosen for their expertise in the CMM or their experience in high maturity assessments.

- Moderator: Bill Curtis, CMM author & Lead Assessor, TeraQuest Metrics
- Mark Paulk, CMM author & Lead Assessor, SEI "What Justifies A Rating of CMM Level 4?"
- John Vu, Lead Assessor, Boeing "The Rating of SW-CMM Level 4."
- Ron Radice, Lead Assessor & Process Pioneer, Software Technology Transition "What Do I Expect to See If I Am Going to Rate an Organization at Level 4?"
- Charlie Weber, CMM author, TeraQuest Metrics "What Justifies a Rating of CMM Level 4? (What We Should Have Said!)."

Mark C. Paulk, "Lessons Learned in Process Modeling." **2000 Southeastern Quality Conference**, Atlanta, GA, 30 Oct 2000.

One of the powerful tools used by high maturity organizations is process modeling. Process models can provide insight into planning, strategic management issues, process control, operational management, process improvement, technology adoption, and training. Building useful and usable models is not, however, a trivial exercise. This paper summarizes some of the lessons learned in selecting a modeling technique, identifying the important measures to parameterize the model, collecting valid data to calibrate it, and applying the insights from the model effectively.

Mark C. Paulk, "Thinking About Change Management in High Maturity Organizations." **Working Conference on Diffusing Software Product and Process Innovations**, International Federation for Information Processing (IFIP) Working Group 8.6, Banff, Canada, 9 April 2001.

Change management is crucial in today's fast moving world. One of the challenges for organizations aspiring to Level 5 against the SEI's Capability Maturity Model for Software is dealing with process and technology change in a "systematic" way. In this context, systematic implies that the change management builds on the understanding of variation established at Level 4 and of technology transition and diffusion of innovation concepts. Systematic also implies a cultural shift that encompasses worker participation and empowerment concepts. The focus of this paper is on the culture shift for high maturity organizations and new models of technology transition and diffusion of innovation that an lend insight into change management needs for high maturity organization. Several models already widely discussed in the software world will be reviewed, plus others that add some useful insights.

Mark C. Paulk, "A History of the Software CMM." September 2001.

This presentation provides an overview of the Software CMM's evolution from its inspirations in Total Quality Management as applied to software to the software process maturity framework published in 1987 to Software CMM v1.1 as released in 1993. Issues faced in building the CMM are discussed, including the question of staged versus continuous architectures. The presentation closes with a brief comparison of the Software CMM v1.1 with CMMI v1.0, which will replace the Software CMM in 2003. A table summarizing the key process area changes from the 1987 framework to Software CMM v1.1 is also provided.

Mark C. Paulk, "Practical SPI." October 2001.

This presentation describes some of the practical considerations associated with software process improvement. Although focusing on the Software CMM, these considerations are generally applicable. The recommendations contained in this presentation are prescriptive. While there may be alternatives that are also effective, the issues and related recommendations discussed here should be thoughtfully considered by any organization undertaking a serious SPI effort.

Mark C. Paulk, "Using the Software CMM With Good Judgment." November 2001.

The Software CMM has been criticized as being applicable only to large organizations and/or large projects. It has also been successfully used across a wide range of organizational sizes and types, in many different application domains, and with many different methodologies. At the same time, CMM-based improvement programs have failed in many large military/aerospace organizations. The crucial difference between successful and failed improvement programs, regardless of the environment, is the use of common sense and good judgment in defining and improving processes. In this presentation I discuss some of the critical factors in practical software process improvement for small organizations, small projects, Internet-speed environments, and agile methodologies. The conclusion is that the Software CMM is a powerful tool for process improvement when used with good judgment, and that those who lack common sense are doomed to failure regardless of their approach to process improvement.

Mark C. Paulk, "Understanding High Maturity Practices: A Software CMM Tutorial." September 2001.

"High maturity," in terms of the Capability Maturity Model (CMM) for Software, implies a superior process capability for a software organization, yet comparatively few organizations have achieved the higher levels of maturity -- levels 4 and 5. The lack of wide-spread experience with high maturity practices is a challenge for both assessors and process engineers. The purpose of this tutorial is to discuss the fundamental principles of the Software CMM at the higher maturity levels and some of the effective engineering and management practices typically found in high maturity organizations. At the end of this tutorial, attendees should have a better understanding of how to interpret the Software CMM at levels 4 and 5 and insight into effective implementations of high maturity practices.

Mark C. Paulk, "High Maturity Practices." April 2002.

This presentation contains the results of the high maturity survey performed in 2001. The survey results will be described in much more detail in the special report, which has not yet been published.

Mark C. Paulk, "Agile Methodologies from a CMM Perspective," September 2002.

This presentation discusses several of the "agile methodologies," such as Extreme Programming and Scrum, from a CMM perspective. The conclusion is that disciplined processes are compatible with a CMM-based improvement perspective, even if some forms do emphasize "lighter weight" processes.

Mark C. Paulk, "Comparing ISO 9001:2000 and Software CMM v1.1," September 2002.

This presentation provides a brief overview of the Capability Maturity Model for Software and of the latest release of the ISO 9000 family of standards for quality management systems. It then compares the Software CMM and ISO 9001. The conclusion is that there is major overlap between these two documents, but there are some areas that each addresses that are not addressed by the other. Due to differences in focus and scope, a mapping between the two is not a simple matter, even though there is major overlap.

Mark C. Paulk, "Trends in Software Process and Quality," October 2002.

This presentation provides an overview of the current world-wide trends in the arena of software process and quality improvement. It begins with a discussion of the Capability Maturity Model for Software developed by the Software Engineering Institute and the state of the practice in software engineering. Other approaches to software process improvement are briefly discussed, including the ISO 9000 family of quality management system standards and the ISO 15504 standard being developed for software process assessment. Pros and cons of these approaches are discussed in comparison to the SEI's work.

QUESTIONS

top ▲ | CMM main page

Carnegie Mellon
Software Engineering Institute

Home Search Contact Us Site Map What's New

Organizations Improving management practices Individuals

About the SEI | Management | Engineering | Acquisition | Work with Us | Products and Services | Publications

MANAGEMENT

- Welcome
- Capability Maturity Modeling
- Team & Personal Software Process
- IDEAL Model
- Risk Management
- Software Engineering Measurement & Analysis (SEMA)
- Software Engineering Information Repository (SEIR)
- Software Process Improvement Networks (SPINs)
- Appraiser Program
- Acronyms
- SEI Initiatives
- Conferences
- Education & Training

Capability Maturity Model® (SW-CMM®) for Software V2 Archive

This Web page archives historical information about the planned Software CMM version 2.0, originally planned for late 1997, which was halted by the Office of the Under Secretary of Defense for Acquisition and Technology. See the the CMM integration Web page for further information.

1998 Archives

- Mappings between ISO 12207, ISO 15504 (SPICE), Software CMM v1.1, and Software CMM v2 Draft C, posted 5 January 1998

1997 Archives

- News release - SEI to support DOD requirement for CMM integration, posted 16 December 1997
- Deadline for Draft C comments extended, posted 31 October 1997
- **SW-CMM v2 release delayed**, posted 30 October 1997
- SW-CMM v2 Draft C, posted 22 October 1997
- Risk Management in SW-CMM v2: The Resolution, posted 19 May 1997

1996 Archives

- Change Request Report, October 1996.
- Delaying the release of SW-CMM V2.0, posted 7 October 1996
- Major decisions for SW-CMM v2, posted 6 August 1996
- Prototype key process areas, 1996

1995 (and Earlier) Archives

- Michael D. Konrad, Mark C. Paulk, and Allan W. Graydon,"An Overview of SPICE's Model for Process Management," **Proceedings of the Fifth International Conference on Software Quality**, Austin, TX, 23-26 October

1995, pp. 291-301.
Note that the SPICE model for process management has been significantly revised since this paper was written. This baseline, however, was the departure point for much of the CMM v2 architecture work and other papers referenced here.

- Working papers from the Feb 95 Brainstorming CMM Workshop
- Other working papers and articles

MANAGEMENT

- Welcome
- ● Capability Maturity Modeling
- Team & Personal Software Process
- IDEAL Model
- Risk Management
- Software Engineering Measurement & Analysis (SEMA)
- Software Engineering Information Repository (SEIR)
- Software Process Improvement Networks (SPINs)
- Appraiser Program
- Acronyms
- SEI Initiatives
- Conferences
- Education & Training

Software Capability Maturity Model® (SW-CMM®) Contact Information

For general information or questions about the SW-CMM, contact

Customer Relations
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
Phone: +1 412-268-5800
FAX: +1 412-268-5758
E-mail: customer-relations@sei.cmu.edu

top | CMM main page

CQA

CERTIFIED QUALITY ANALYST

Administered by Quality Assurance Institute

7575 Dr. Phillips Blvd. • Suite 350
Orlando, FL 32819-7273
Telephone (407) 363-1111
Fax (407) 363-1112

CLIFFORD R. KETTEMBOROUGH

CERTIFIED SINCE 1990
CERTIFICATE NUMBER 1246

Inquiries Regarding Your Certification
Should be Directed To

Director of Certification
Quality Assurance Institute
7575 Dr. Phillips Blvd., Suite 350
Orlando, FL 32819-7273
☎ 407-363-1111
FAX 407-363-1112

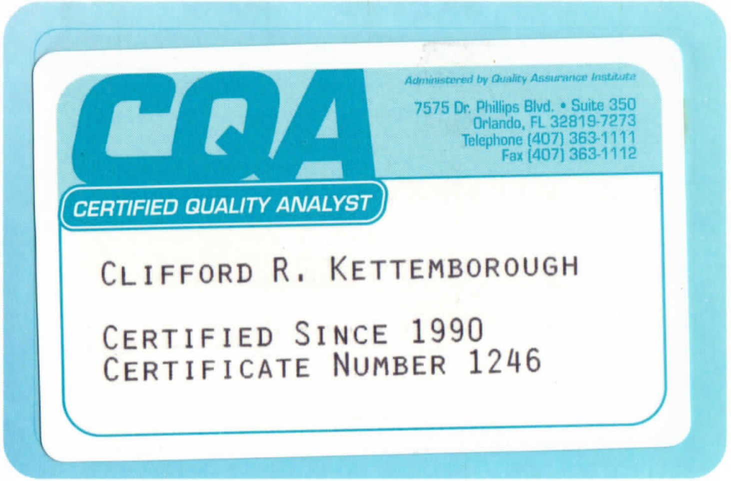
The CQA program and its designation are administered by the Quality Assurance Institute.

CERTIFIED QUALITY ANALYST

Here's your personalized ID card verifying your certification as a CQA!

Carry this card with you in evidence of meeting a professional standard of proficiency in IT quality assurance.

Please remember that meeting all Continuing Professional Education (CPE) requirements and fees is mandatory for recertification. Be sure to include your certificate number on all correspondence.



Administered by Quality Assurance Institute

7575 Dr. Phillips Blvd. • Suite 350
Orlando, FL 32819-7273
Telephone (407) 363-1111
Fax (407) 363-1112

CQA

CERTIFIED QUALITY ANALYST

CLIFFORD R. KETTEMBOROUGH

CERTIFIED SINCE 1990
CERTIFICATE NUMBER 1246

(https://www.scrumalliance.org/)

(/accounts/99653-clifford-r)

Clifford R Kettemborough, PhD, DBA, EdD

- it-smc.com (www.it-smc.com)
- wwetc.com (www.wwetc.com)
- CliffordKettemborough.Academia.edu (CliffordKettemborough.Academia.edu)
- CliffordKettemborough.com (www.CliffordKettemborough.com)
- linkedin.com/in/kettemborough (www.linkedin.com/in/kettemborough)

Director - Technology at **IT-SMC**

Location: Burbank, CA

- Certified Scrum Professional® - ScrumMaster
- Certified ScrumMaster®

- Contact Clifford (mailto:cliffrk@earthlink.net?body=I%20saw%20your%20Member%20page%20at%20https%3A%2F%2Fwww.scrumalliance.org%2Faccounts%2F99653-clifford-r&subject=ScrumAlliance%20Member%20Page)
- www.it-smc.com (www.it-smc.com)
- View Certification Profile (https://www.scrumalliance.org/community/profile/ckettembor)
- Edit Account Details (/accounts/99653-clifford-r/edit)

Clifford R. Kettemnborough, Ph.D., D.B.A., Ed.D Dr. Kettemborough (www.CliffordKettemborough.com) functioned in professional positions, includin... Read More

Trainer at **Msys Technologies**

2016-Present

Show Details

V Help l data (/takeout/99653-clifford-r)

/

The George Washington University

School of Business

certifies that

Clifford R. Kettemborough

has successfully completed

Managing IT Projects

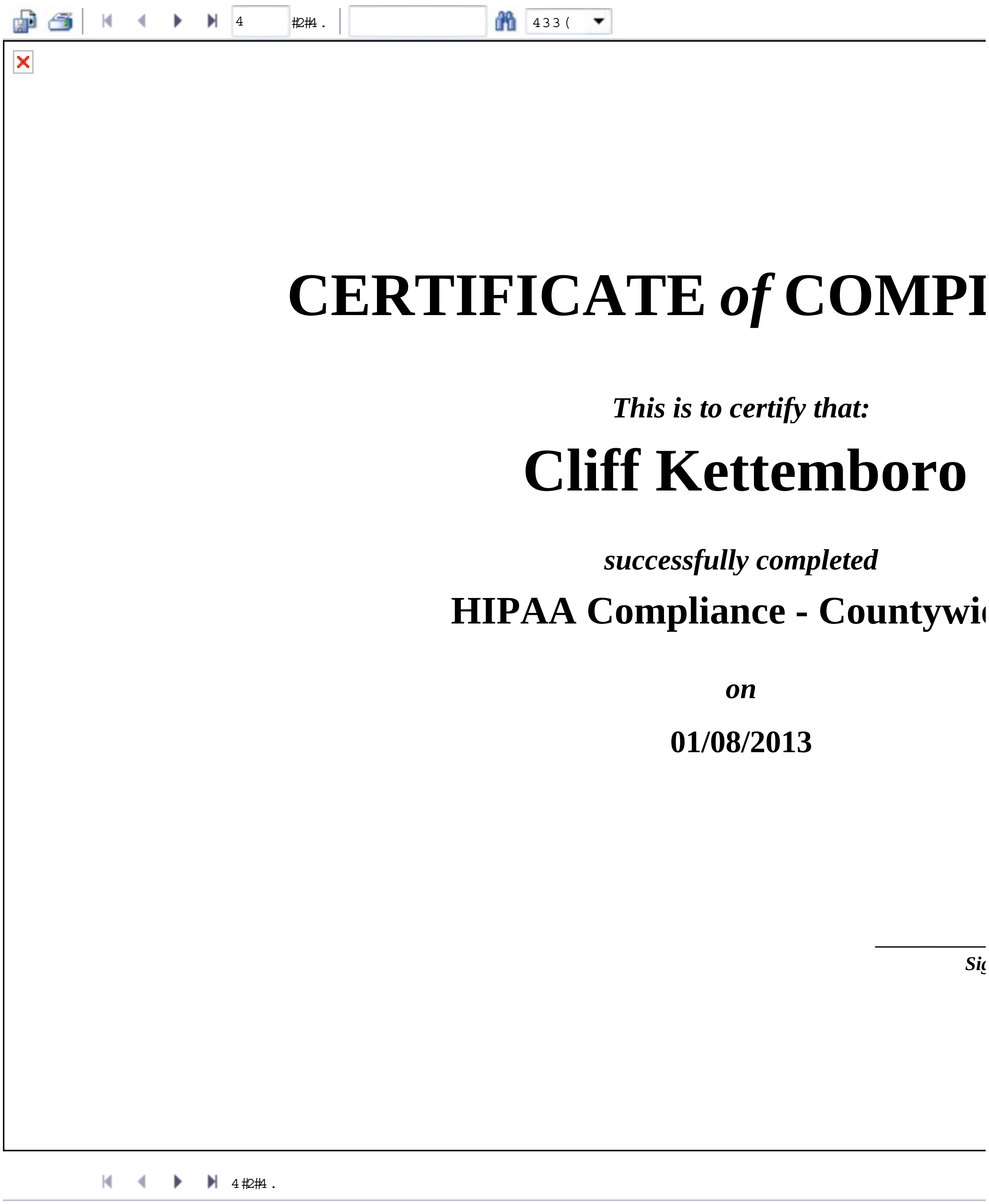
August 10, 2009 through August 12, 2009
Gardena, CA

and is awarded 2.25 *continuing education units*

Washington, D.C

[illegible]

Dean, School of Business
The George Washington University



4 #2#4 . 433 (

CERTIFICATE *of* COMPI

This is to certify that:

Cliff Kettemboro

successfully completed

HIPAA Compliance - Countywi

on

01/08/2013

Si

4 #2#4 .

International Purchasing and Supply Chain Management Institute

International Purchasing and Supply Chain Management Institute

IPSCMI

Clifford R. Kettemborough

is hereby Certified as a

Certified International Procurement Professional (CIPP)

As a purchasing and supply chain management professional, this includes the responsibility to maintain the highest ethical practice to favorably reflect upon the profession.

Given at Lewes, Delaware, the United States.

Certification ID: *CIPP2220822429*
Issue Date: *August 22, 2022*
Expiration Date: *NEVER*

IPSCMI

[illegible]

President, Certification Committee

International Purchasing and Supply Chain Management Institute

International Purchasing and Supply Chain Management Institute

IPSCMI

Clifford R. Kettemborough

is hereby Certified as a

Certified International Supply Chain Professional (CISCP)

As a purchasing and supply chain management professional, this includes the responsibility to maintain the highest ethical practice to favorably reflect upon the profession.

Given at Lewes, Delaware, the United States.

Certification ID: *CISCP2220837729*
Issue Date: *August 22, 2022*
Expiration Date: *NEVER*

[illegible]

President, Certification Committee

Quality Assurance Services of North America (QASNA)

Certificate of Successful Completion

Clifford R. Kettemborough

Has Successfully Completed The

ISO 9000 Internal Auditor Training Course

(Course conducted under the guidelines of the National Registration Scheme for Assessors of Quality Systems administered by the Institute of Quality Assurance.)

[illegible]	December 4, 1997	QIACSC/118/97
Director	Date	Number

Learn Moodle 3.4 Basics completer

Dashboard / Preferences / Badges / Manage badges / Learn Moodle 3.4 Basics completer

Recipient details

Name	Clifford Kettemborough

Issuer details

Issuer name	Learn Moodle
Contact	learn@moodle.net

Badge details

Name	Learn Moodle 3.4 Basics completer
Description	This badge is awarded to people who have completed all activities in the Learn Moodle 3.4 Basics MOOC.
Course	Learn Moodle 3.4 Basics
Criteria	Users are awarded this badge when they complete the following requirement: • Users must complete the course **"Learn Moodle 3.4 Basics"**

Download

Badge expiry

Date issued	Wednesday, January 31, 2018, 5:32 PM
Evidence	This badge was issued for completing: • Users must complete the course **"Learn Moodle 3.4 Basics"**

You are logged in as Clifford Kettemborough (Log out)
Home

About
Learn Moodle
MOOC overview
Important dates
MOOC FAQ
Learn Moodle 3.4 Basics
Learn Moodle mobile app
English - United States (en_us)
አማርኛ (am)
Afrikaans (af)
Aragonés (an)
Aranés (oc_es)
Asturianu (ast)
Azərbaycanca (az)
Bahasa Melayu (ms)
Bislama (bi)
Bosanski (bs)
Breizh (br)
Català (ca)
Català (Valencià) (ca_valencia)
Čeština (cs)
Crnogorski (mis)
Cymraeg (cy)
Dansk (da)
Dansk Rum (da_rum)
Davvisámegiella (se)
Deutsch (de)
Dolnoserbski (dsb)
Ebon (mh)
eesti (et)
English - United States (en_us)
English (en)
Español - Colombia (es_co)
Español - Internacional (es)
Español - México (es_mx)
Español Venezuela (es_ve)
Esperanto (eo)
Euskara (eu)
Èʋegbe (ee)
Filipino (fil)
Finlandssvenska (sv_fi)
Føroyskt (fo)
Français - Canada (fr_ca)
Français (fr)
Gaeilge (ga)
Gàidhlig (gd)

Microsoft CERTIFIED Professional

Microsoft Certified Professional Transcript

Latest Activity Recorded Mar 13, 2006

CLIFFORD R. KETTEMBOROUGH
6458 N. LEMON AVENUE
SAN GABRIEL, CA 91775 US
cliffrk@earthlink.net

Microsoft Certified Professional ID: **3450816**

Microsoft Certification Status

Credential	Certification \ Version	Date Achieved
Microsoft Certified Systems Engineer		Mar 13, 2006
	Microsoft Windows Server 2003	Mar 13, 2006
Microsoft Certified Systems Administrator		Feb 21, 2006
	Microsoft Windows Server 2003	Feb 21, 2006
Microsoft Certified Database Administrator		Sep 12, 2005
	Microsoft SQL Server 2000	Sep 12, 2005
Microsoft Certified Professional		Jul 25,2005

Microsoft Certification Exams Completed Successfully

Exam ID	Description	Date Completed
294	Planning, Implementing, and Maintaining a Microsoft Windows Server 2003 Active Directory Infrastructure	Mar 13, 2006
297	Designing a Microsoft Windows Server 2003 Active Directory and Network Infrastructure	Mar 02, 2006
290	Managing and Maintaining a Microsoft Windows Server 2003 Environment	Feb 21, 2006
293	Planning and Maintaining a Microsoft Windows Server 2003 Network Infrastructure	Sep 12, 2005
291	Implementing, Managing, and Maintaining a Microsoft Windows Server 2003 Network Infrastructure	Aug 29, 2005
229	Designing and Implementing Databases with Microsoft® SOL Server(TM) 2000 Enterprise Edition	Aug 12, 2005
228	Installing, Configuring, and Administering Microsoft® SQL Server(TM) 2000 Enterprise Edition	Aug 08, 2005
270	Installing, Configuring, and Administering Microsoft® Windows® XP Professional	Jul25,2005

JET PROPULSION LABORATORY

INTEROFFICE MEMORANDUM
501JWS 00-021

September 21, 2000

TO: Distribution

FROM: J. W. Suitor Jerry

SUBJECT: NASA Group Achievement Award Certificates

The NASA Group Achievement Award Certificates have arrived. I would dearly like to gather everyone together to pass out the awards but the shear number of people makes the logistics a nightmare. Instead, please accept my appreciation for the work you did in getting JPL ISO 9001 certified.

You all spent many hours struggling with understanding a different way of conducting our business and getting it in place to demonstrate that our management approach to our business is a true quality approach. Dr. Stone described it as the biggest "launch" the Laboratory has ever undertaken. You all were a part of that "launch" team and you should be proud. You have laid the foundation for a new century of awesome space discoveries. In light of the accomplishment, "Thank you" seems a bit light weight. Nevertheless, **Thank You**.

JWS:sa

attachment

American Certification Institute

AMERICAN CERTIFICATION INSTITUTE

Clifford R. Kettemborough, Ph.D., DBA

is hereby Certified as a

Certified International Professional in Project Management
(CIPPM)

As a business management professional, this includes the responsibility to maintain the highest ethical practice to favorably reflect upon the profession.

Given at Lewes, Delaware, the United States.

Certification ID:	*CIPPM2170912452*
Issue Date:	*September 20, 2017*
Expiration Date:	*September 20, 2022*

AMERICAN CERTIFICATION INSTITUTE

[illegible]

President, Certification Committee

This is to certify that

Clifford Kettemborough

is a member of Project Management Institute, the world's leading association for those who consider project, program, or portfolio management their profession; and upholds the Institute's Code of Ethics and Professional Conduct.

PMI® CERTIFIED MEMBER

PROJECT MANAGEMENT INSTITUTE
CORPORATE
SEAL
1969
PENNSYLVANIA

Project Management Institute®

Powering The Project Economy™

Project Management Institute®

Clifford Kettemborou

Member ID:
1738129

Member Since:
7/2011

Member Expiration:
7/2020

PENNSYLVANIA · 1969 · PROJECT MANAGEMENT INSTITUTE · CORPORATE SEAL ·

PMI Project Management Institute

This is to certify that

Clifford Kettemborough

is a member of
PROJECT MANAGEMENT INSTITUTE,
a global membership association
dedicated to advancing the practice,
science and profession of project
management, and upholds the
Institute's Code of Ethics
and Professional Conduct.

PMI Project Management Institute

Making project management indispensable
for business results.®

Member

PENNSYLVANIA · 1969 · PROJECT MANAGEMENT INSTITUTE · CORPORATE SEAL

PMI Project Management Institute

This is to certify that

Clifford Kettemborough

is a member of
PROJECT MANAGEMENT INSTITUTE,
a global membership association
dedicated to advancing the practice,
science and profession of project
management, and upholds the
Institute's Code of Ethics
and Professional Conduct.

PMI Project Management Institute

Making project management indispensable
for business results.®

Member

Clifford Kettemborough, PhD, DBA, EdD

This Badge is: https://bcert.me/smdzmzxoj

Scrum Foundations Educator - **CERTIFIED BADGE**

Issued To	Clifford Kettemborough, PhD, DBA, EdD
Title	IT Director - Technology
Company	The Walt Disney Co - Studio
Badge ID	1633402
Issued By	Scrum Alliance, Inc.
Location	Denver, CO, US
Issue Date	08/11/2022
Expiration Date	11/06/2022
Description	A Scrum Foundations trainer has an understanding of the practices and principles of Scrum and real-world experience in actual Scrum organizations. Scrum Foundations trainers help students understand the foundational concepts of Scrum. Certification Requirements The Scrum Foundations trainer: ● Holds an active Certified Scrum Trainer, Certified Enterprise Coach, Certified Team Coach, and/or Certified Scrum Professional designation ● Has reviewed and signed the Scrum Foundations license agreement Certificants must maintain their active status by adhering to renewal requirements, shown here.
URL evidence	1633402-scrumfoundationseduc

ScrumAlliance®
Transforming the World of Work

Clifford R. Kettemborough, Ph.D., D.B.A., Ed.D

is awarded the designation Certified Scrum Professional® on this day, November 04, 2010, for completing the prescribed requirements for this certification and is hereby entitled to all privileges and benefits offered by SCRUM ALLIANCE®.

Scrum Alliance
CSP
CERTIFIED

Member: 000099653 Certification Expires: 07 November 2014

Mike Cohn
Chairman of the Board

ScrumAlliance®
Transforming the World of Work

Clifford R. Kettemborough, Ph.D., D.B.A., Ed.D

is awarded the designation Certified ScrumMaster® on this day, June 30, 2010, for completing the prescribed requirements for this certification and is hereby entitled to all privileges and benefits offered by SCRUM ALLIANCE®.

Scrum Alliance
CSM
CERTIFIED

Member: 000099653 Certification Expires: 07 November 2014

[illegible]
Certified Scrum Trainer®

Mike Cohn
Chairman of the Board

American Certification Institute

Clifford R. Kettemborough, Ph.D., DBA

is hereby Certified as a

Certified 6-Sigma Black Belt Professional

(CSSBBP)

As a business management professional, this includes the responsibility to maintain the highest ethical practice to favorably reflect upon the profession.

Given at Lewes, Delaware, the United States.

[illegible]

President, Certification Committee

Certification ID: *CSSBBP2160746857*
Issue Date: *July 1, 2016*
Expiration Date: *June 30, 2021*

redstone learning | Creating Value That Lasts!

EZ Certifications STAY RELEVANT. STAY AHEAD

Certificate No: LSSGB.9891.001

Clifford R. Kettemborough

Has satisfactorily fulfilled the requirements established
By Redstone Learning for professional attainment in
Lean Six Sigma Green Belt
The certification acknowledges the technical expertise and the ability to apply quality methodologies to drive business improvement and increase customer satisfaction.

July 14th 2016

[illegible]

Nabin Roy, COO
Redstone Learning

Y2K LEADERSHIP TEAM

TEAM AWARD

CHARLES ACTON
PETER BRECKHEIMER
JOHN EKELUND
GREGORY GARNER
PIETER KALLEMEYN
CLIFFORD KETTEMBOROUGH

JPL

THIS AWARD IS PRESENTED TO

CLIFFORD KETTEMBOROUGH

JET PROPULSION LABORATORY

FOR

FOR OUTSTANDING LEADERSHIP OF THEY2K SOFTWARE TESTING EFFORT.

ON

JULY 1999

[illegible]

LINCOLN WOOD
SECTION MANAGER