



“Internet-Speed Processes” From a CMM⁰ Perspective

Mark C. Paulk

mcp@sei.cmu.edu -or- Mark.Paulk@ieee.org

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

⁰ 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.
©2001 by Carnegie Mellon University.



A Rose By Any Other Name...

Internet-speed processes...

aka lightweight processes...

aka light processes...

aka lean processes

aka agile processes...

Rigorous processes...

aka disciplined processes...

aka bureaucratic processes...

aka heavyweight processes...

aka heavy processes...

aka industrial-strength processes...

Currently preferred terms: *agile* vs *rigorous*



The Agile Alliance

Agile Manifesto

“We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

- **individuals and interactions** over processes and tools
- **working software** over comprehensive documentation
- **customer collaboration** over contract negotiation
- **responding to change** over following a plan

That is, while there is value on the items on the right, we value the items on the left more.”

<http://www.agilealliance.org>



Internet-Speed Processes

Three driving features:

- first-to-market
- release driven products
- fluid specifications

Five quality features:

- built-in tolerance for original “toad code”
- parallel quality assurance
- negotiated quality marketplace
- feature slip
- necessity of good people

Time drives \Downarrow quality depends \Downarrow process adjusts

Richard Baskerville, Will Hayes, Linda Levine, et al., “Negotiating Software Quality @ Internet Speed,” IEEE Computer.



Topics

→ **The Software CMM**

Extreme Programming (XP)

Crystal Light Methodologies

Scrum

Adaptive Software Processes

Concluding Thoughts



Sources of Info on the CMM

Mark Paulk, Charles Weber, Bill Curtis, and Mary Beth Chrissis, The Capability Maturity Model: Guidelines for Improving the Software Process.

Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, and Charles V. Weber, "Capability Maturity Model, Version 1.1," IEEE Software, July 1993.

Mark C. Paulk, "Using the Software CMM With Good Judgment," ASQ Software Quality Professional, June 1999.

[http:// www.sei.cmu.edu/cmm/](http://www.sei.cmu.edu/cmm/)

[http:// www.sei.cmu.edu/cmm/cmm.articles.html](http://www.sei.cmu.edu/cmm/cmm.articles.html)



What Is the Capability Maturity Model⁰ for Software?

A **common-sense** application of process management and quality improvement concepts to software development and maintenance

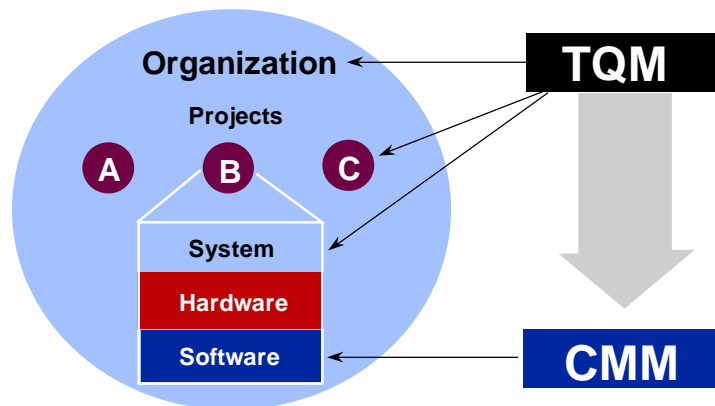
A **community-developed** guide

A model for **organizational** improvement

The underlying structure for **reliable and consistent** CMM-based appraisal methods



Applying TQM to Software



Process improvement fits in an overall business context—CMM applies to software.



Software CMM v1.1

Level	Focus	Key Process Areas	
5 Optimizing	<i>Continual process improvement</i>	Defect Prevention Technology Change Management Process Change Management	Quality Productivity
4 Managed	<i>Product and process quality</i>	Quantitative Process Management Software Quality Management	
3 Defined	<i>Engineering processes and organizational support</i>	Organization Process Focus Organization Process Definition Training Program Integrated Software Management Software Product Engineering Intergroup Coordination Peer Reviews	
2 Repeatable	<i>Project management processes</i>	Requirements Management Software Project Planning Software Project Tracking & Oversight Software Subcontract Management Software Quality Assurance Software Configuration Management	
1 Initial	<i>Competent people and heroics</i>		Risk Waste

June 2001

9

E-SEPG 2001



Where Does CMM Apply?

Software CMM was written to provide good software engineering and management practices for any project in any environment.

- practices described in hierarchy
- detailed practices primarily support large, contracting software organizations
- “normative” components of the CMM are maturity levels, key process areas, and goals
- ***all practices are informative!***

Organizational learning prevents reinventing the wheel

® repeatable processes

June 2001

10

E-SEPG 2001



CMM Criticisms

Large (approximately 500 pages)

Designed for large organizations and projects

Rigid and bureaucratic -- for “process nazis”

Process for process’ sake; focus on levels

Note that the CMM requirements to be Level 5 are the 52 goals of the 18 key process areas.

- practices, subpractices, and examples are informative material***
- common sense and professional judgment are intrinsic to effective process management***



Topics

The Software CMM

→ Extreme Programming (XP)

Crystal Light Methodologies

Scrum

Adaptive Software Processes

Concluding Thoughts



Sources of Info on XP

Kent Beck, Extreme Programming Explained: Embrace Change.

Kent Beck, "Embracing Change with Extreme Programming," IEEE Computer, October 1999.

Kent Beck, "Emergent Control in Extreme Programming," Cutter IT Journal, November 2000.

<http://computer.org/seweb/dynabook/Index.htm>

<http://www.xprogramming.com/>

<http://www.extremeprogramming.org/>

<http://ootips.org/xp.html>



XP Essentials

XP assumes the high cost of change has been addressed by technologies such as objects/patterns, relational databases, information hiding, etc.

XP improves a software project in four essential ways: communication, simplicity, feedback, and courage

Four basic activities of XP: coding, testing, listening, and designing



Basic Principles of XP

Rapid feedback

Assume simplicity

Incremental change

Embracing change

Quality work

Less central principles: teach learning, small initial investment, play to win, concrete experiments, open & honest communication, work with people's instincts not against them, accepted responsibility, local adaptation, travel light, honest measurement



XP Practices (1 of 3)

Planning game -- quickly determine the scope of the next release, combining business priorities and technical estimates

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 they whole system works

Simple design -- designed as simply as possible at any given moment



XP Practices (2 of 3)

Testing -- continually write unit tests which must run flawlessly; customers write tests to demonstrate functions are finished

Refactoring -- restructure the system without changing behavior to remove duplication, improve communication, simplify, or add flexibility

Pair programming -- all production code written by two programmers at one machine

Collective ownership -- anyone can change any code anywhere in the system at any time



XP Practices (3 of 3)

Continuous integration -- integrate and build the system many times a day, every time a task is finished

40-hour week -- work no more than 40 hours per week as a rule; never work overtime two weeks in a row

On-site customer -- real, live user on the team full-time to answer questions

Coding standards -- rules emphasizing communication throughout the code



Mapping XP to CMM

RM	ö ö	OPF	ö	QPM	
SPP	ö ö	OPD	ö	SQM	
SPTO	ö ö	TP			
SSM		ISM		DP	ö
SQA	ö	SPE	ö	TCM	
SCM	ö ö	IC	ö ö	PCM	
		PR	ö		

ö partially addressed in XP
 ö ö mostly addressed in XP
 (perhaps by inference)
 (in the appropriate environment)



Topics

The Software CMM

Extreme Programming (XP)

→ Crystal Light Methodologies

Scrum

Adaptive Software Processes

Concluding Thoughts



Basis of Crystal Light Methodologies

“The strength of a light methodology comes from the use of informal, face-to-face communication channels rather than written documents to bind the large amount of information flowing within the project.”

Crystal is segmented into color-coded bands with common characteristics, from small to large: Clear, Yellow, Orange, Red, Maroon, Blue, Violet and so on.

**<http://members.aol.com/acockburn/>
<http://crystalmethodologies.org/>**



Crystal Light Methodologies

Crystal is the nickname for *The Humans and Technology Manifesto of Software Development*.

“Software development is a cooperative game, in which people use markers and props to inform, remind and inspire themselves and each other in getting to the next move in the game. The endpoint of the game is an operating software system; the residue of the game is a set of markers to inform and assist the players of the next game. The next game is the alteration or replacement of the system, or creation of a neighboring system.”



Crystal Clear Principles

Every project deserves its own, tailored process and methodology

“Lightweight” and “face-to-face” are desirable attributes

People communicate best face-to-face

Daily variability, inconsistency and laziness are the weaknesses to allow for (“high-discipline processes are fragile” is the catch phrase)

A process should build upon good citizenship, the ability of people to look around, talk to each other, and take initiative, as the native strengths of people



Values for Crystal Clear

Strong on communications, light on deliverables.

Keep the communications paths short, rich, and informal.

Deliver frequently.

Reduce overhead.

More deliveries and better communication reduce the need for intermediate work products.



People and Methodology Design

People sensitive to communication timing and modalities

- **physical proximity and ease of communication has dominant effect**

People tend to inconsistency

- **methodologies requiring disciplined consistency are fragile in practice**

People vary, not just daily, but from group to group

- **need to deal with this cultural variation**

People like to be good citizens, are good at looking around and taking initiative

- **these combine to form that common success factor, “a few good people stepped in at key moments”**



Target for Crystal Clear

Project size: up to 6 developers, typically 3-4

System's potential for damage: essential moneys, but not life

- **might be adjustable up to 8 people or legal liability systems but not both**

Types of systems: mainframe or client-server, any database, central or distributed



Not Targeted

Not intended for

- **larger teams**
 - missing group coordination
- **life-critical systems**
 - missing verification of correctness

Adjustable to, but not directly intended for, hard real-time systems

- missing planning and verification of timing



Crystal Clear and People

Distinct people needed

- **sponsor**
- **senior designer**
- **user**
- **designer-programmers (aka “designers”)**

Roles needed

- **business expert (may come from sponsor, user, or senior designer)**
- **coordinator (may come from senior designer)**
- **tester (may come from designers)**
- **writer (may come from designers)**



Crystal Clear Deliverables (1 of 3)

Sponsor

- **mission statement**

Coordinator

- **release sequence**
- **viewing & release schedule**
- **risk list**
- **project status**



Crystal Clear Deliverables (2 of 3)

Senior designer

- **team structure (part of methodology)**
- **methodology**
- **system design**

Business expert

- **actor-goal pairs (part of requirements doc)**
- **annotated use cases (part of requirements doc)**
- **requirements document**

User

- **helps with use cases and screen drafts**



Crystal Clear Deliverables (3 of 3)

Designers

- screen drafts
- design sketches and notes
- common object model
- source code
- packaged system
- migration code
- test cases

Testers

- test results / defect reports

Writer

- user manual



Standards in Crystal Clear

Policy standards

- use of increments for project staging, tracking by milestones and predicted risks
 - increments are two months or less
- annotated usage scenarios for requirements
- use of a regression testing framework
- use of peer code reviews
- direct user involvement
- use of an ownership model

Locally set standards

- locally set programming style standards
- locally selected user interface design standards
- locally selected regression testing framework



Mapping Crystal Clear to CMM

RM	ö ö	OPF	ö	QPM
SPP	ö ö	OPD	ö	SQM
SPTO	ö ö	TP		
SSM		ISM		DP
SQA		SPE	ö ö	TCM
SCM	ö	IC	ö	PCM
		PR	ö ö	

ö partially addressed in Crystal Clear
ö ö mostly addressed in Crystal Clear
(perhaps by inference)
(in the proper environment)



Topics

The Software CMM

Extreme Programming (XP)

Crystal Light Methodologies

→ Scrum

Adaptive Software Processes

Concluding Thoughts



Scrum

A process for incrementally building software in complex environments

Basic premise: if you are committed to the team and the project, and if your boss really trusts you, then you can spend your time being productive instead of justifying your work

Ken Schwaber, "Against a Sea of Troubles: Scrum Software Development," Cutter IT Journal, November 2000.

<http://www.controlchaos.com>



Scrum Concepts

Premise: no organization can definitively predict and plan what software it will deliver, when, or what the quality and cost will be

***Backlog* -- all outstanding work for a product area**

***Sprints* -- 30-day increments of work that produce a deliverable**

- **highest priority group of backlog that can be completed in less than 30 days is worked on; no interruptions are allowed**

***Scrums* -- daily status check meetings**



Scrum Meetings

Three questions

- **What did you do since the last Scrum?**
- **What got in your way?**
- **What are you going to do before the next Scrum?**

Scrum meeting protocol

- **daily, same place and time**
- **only three questions**
- **all pigs (committed) must respond**
- **chickens (involved) can attend, but must be silent**
- **no new backlog can be introduced externally**
- **backlog can be added internally**



Scrum Sprint Rules

Use small interdisciplinary teams

Build clean interface software

Intelligent management required

Solid systems architecture and framework upfront

Prototype all new tools and technology

Develop infrastructure first

Each Sprint results in an executable

Develop, document, and test in parallel



Why Scrum is Powerful

Focus is on team's work and team's work only

Daily communication of status occurs

Enables low-overhead empirical management

Makes impediments visible

Someone is willing to make decisions and remove impediments real-time



Mapping Scrum to CMM

RM	ö ö	OPF	ö	QPM
SPP	ö ö	OPD	ö	SQM
SPTO	ö ö	TP		
SSM		ISM		DP
SQA		SPE	ö ö	TCM
SCM	ö	IC	ö	PCM
		PR		

ö partially addressed in Scrum
 ö ö mostly addressed in Scrum
 (perhaps by inference)
 (in the proper environment)



Topics

The Software CMM

Extreme Programming (XP)

Crystal Light Methodologies

Scrum

→ Adaptive Software Processes

Concluding Thoughts



Adaptive Cultures

“Adaptive cultures understand that success evolves from a succession of trying different alternatives, and learning from both success and failure.”

James A. Highsmith, Adaptive Software Development: A Collaborative Approach to Managing Complex Systems.

Jim Highsmith, "Retiring Lifecycle Dinosaurs," Software Testing and Quality Engineering, July/August 2000.



Principles of Adaptive Software Development

None of us can anticipate function very well.

Iteration -- building, trying, succeeding, failing, rebuilding -- governs successful product development.

Change, improvisation, and innovation rule.

The Adaptive Lifecycle

Speculate - Collaborate - Learn



Speculation

Keep delivery cycles short and encourage iteration

Recognize the uncertain nature of complex problems

Encourage exploration and experimentation

Admit that we don't know everything



Steps in Speculating

Conduct the project initiation phase.

Determine the project timebox.

Determine the optimal number of cycles and the timebox for each.

Write an objective statement for each cycle.

Assign primary components to cycles.

Assign technology and support components to cycles.

Develop a project task list.



Collaboration

Complex applications aren't built; they evolve.

None of us is as smart as all of us.

Shared creation crosses traditional team boundaries.



Learning

We have to test our knowledge constantly.

Post-mortem reviews should be done after each iterative cycle rather than waiting until the end of the project.



Categories of Things to Learn

Result quality from the customer's perspective

Result quality from a technical perspective

Functioning of the delivery team and the practices they are using

Project's status

“Quality evolves -- not from micromanaging the process, but from establishing appropriate exit criteria and review practices.”



Characteristics of an Adaptive Lifecycle (1 of 3)

Mission-focused

- **mission statements act as guides that encourage exploration in the beginning but narrow over the life of the project**
- **provide boundaries rather than a fixed destination**
- **focuses on results, not tasks**

Component-based

- **group of features to be developed during an iterative cycle**



Characteristics of an Adaptive Lifecycle (2 of 3)

Iterative

- **emphasizes “re-doing” as much as “doing”**

Timeboxed

- **set fixed delivery times for iterative cycles and projects**
- **minimally about time -- really about forcing hard trade-off decisions**
- **forces team and customer to continually re-evaluate the validity of the project’s mission profile**



Characteristics of an Adaptive Lifecycle (3 of 3)

Risk driven

Change tolerant

- **ability to incorporate change is a competitive advantage**



Mapping Adaptive Software Development to CMM

RM	ö ö	OPF	ö	QPM
SPP	ö ö	OPD	ö	SQM
SPTO	ö ö	TP		
SSM		ISM		DP
SQA		SPE	ö ö	TCM
SCM		IC	ö	PCM
		PR		

ö **partially addressed in Scrum**
 ö ö **mostly addressed in Scrum**
 (perhaps by inference)
 (in the proper environment)



Topics

The Software CMM

Extreme Programming (XP)

Crystal Light Methodologies

Scrum

Adaptive Software Processes

→ Concluding Thoughts



CMM and Internet Speed

CMM tells *what* to do in general terms, but does not say *how* to do it.

Internet-speed processes (aka agile processes, lightweight processes) are sets of best practices that contain fairly specific how-to information -- an implementation model.

Practices of Internet-speed processes may

- **fit within a CMM context**
- **not be mandated by the CMM practice**
- **may not completely address the intent of the practice (or goal or key process area)**
- **violate accepted “good software engineering”**



Using CMM and Internet-Speed Processes Appropriately

CMM focuses on the management issues associated with putting effective and efficient processes in place, along with systematic process improvement.

Internet-speed processes prescribe a specific set of practices -- a “methodology” -- that is effective within a specific context.

Both have good ideas that can be synergistic... particularly in conjunction with other good engineering and management practices.



“When speed is critical, you do not have time to figure out what to do or to fix mistakes, you only have time to execute.”

**Bill Curtis
European SEPG Conference, 8 June 2000**



Caveats

Internet-speed processes, as published, should not be used

- for life critical or high reliability systems
- by large and/or virtual teams

Internet-speed processes can be changed / improved for different environments...

... but will the emergent properties that provide value in its proper context still emerge?



Let Common Sense Prevail!

		Process Discipline	
		Yes	No
Common Sense	Yes	Quality	<i>Creative Chaos</i>
	No	<i>Mindless Bureaucracy</i>	<i>Mindless Chaos</i>

With thanks to Sanjiv Ahuja, former President and COO of Telcordia Technologies.