CMMI & Agile

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Mike Konrad
Shane McGraw

SEI Webinar
November 13, 2008

Brief Bios of SEI Staff

Mike Konrad

- With SEI for 20 years
- CMMI Architect, CCB Chair, Model Team Lead
- Your presenter!

Shane McGraw

- With SEI for 6 years
- SPIN Coordinator for the SEI
- Your host!

CMMI and Agile Are Perceived to be at Odds with Each Other

Reliant on process definitions, measures, and artifacts	Skeptical of process definitions, measures, and artifacts
Predictive	**Emergent**
Generally driven by management	Often begun as grass-roots effort

CMMI and Agile proponents are apt to demonize or ignore each other.

SEI Report on CMMI & Agile

To build awareness of the synergies, the SEI has co-authored a report:

CMMI or Agile: Why Not Embrace Both!

http://www.sei.cmu.edu/publications/documents/08.reports/08tn003.html

SEI's co-authors include expert Agile practitioners in the Agile community:

- **Hillel Glazer**
  - author of one of the first articles on CMM and Agile (2001) and *Hillel Glazer's Blog* at http://www.agilecmmi.com/, and a SCAMPI Lead Appraiser
- **Jeff Dalton**
  - co-author of the forthcoming *Agile CMMI (to be published by Auerbach) and* author of *Jeff Dalton's Blog: Ask The CMMI Appraiser* at http://askthecmmiappraiser.blogspot.com/, and a SCAMPI Lead Appraiser
- **Dave Anderson**
  - co-author of *The Declaration of Interdependence*, program manager for *MSF for CMMI*, author of one of the first books on an Agile methodology (FDD), and author of *Dave Anderson's Website and Blog* at http://www.Agilemanagement.net

Reasons for Discord

Early adopters of each approach represent extremes:

- CMMI: Large-scale, mission critical, extensive oversight
- Agile: Fast-moving, empowered teams, volatile requirements, access to customer

Negative perceptions of the other side

- 20 years of CMM and then CMMI sometimes being misapplied
- Top-down vs. bottom-up approaches
- Terminology and style can be “turnoffs!”
  - Technical data package, predictability, Extreme programming

Lack of Accurate Information

- Agilistas more familiar with CMM
- Many incremental improvements in CMMI go unrecognized
- Some claiming to be Agile (or using CMMI) are not!
- Agile DOES requires discipline

CMMI Users Need to Know

Agile arose, in part, as a backlash to CMM and CMMI Misuse

Agile applies sounds SWE principles in certain project contexts

Practitioners need to be involved in implementing CMMI

- Address business needs and priorities
- Capture what is already done well
- Use understandable language and formalisms
- Balance standardization and tailoring
- Ensure and support deployment to new projects (OPF SG3, IPM SP1.1)
- Make the project process transparent
- Encourage, evaluate, and implement feedback and improvement suggestions
- Periodically revise
- Articulate benefits

CMMI Use Should Be More Like the Left

Appropriate use	*Misuse*
Focus on improved performance; ratings should be a natural by-product	**Focus on maturity level**
Use to identify, motivate, learn about, and improve processes by which the business gets done	**Use as a process standard (Note: CMMI says it "contains neither processes nor procedures")**
Use to achieve greater flexibility and leanness in dynamic, high-trust environments	**Use to create bureaucracy; unnecessarily ceremonial, wasteful processes**
Regularly migrate to newer versions of CMMI; investigating what changed and why	**Stick to the older versions; fail to investigate reasons for change; falling behind (and giving real CMMI practitioners a bad name)**
Appropriately complementing the journey with new frameworks, methods, and technologies; learn what works and use it!	**Only sticking to CMMI and ignoring or demonizing Six Sigma, Agile, etc.**
Consider the informative material in interpreting and implementing the practices	**Either ignoring the informative material or using it as a checklist**

Skill in CMMI Interpretation is Necessary

Goals are required, not practices

Practices are expected, not subpractices, typical work products

But informative material is still important as it clarifies practice intent

- a practice statement is a summary of the larger concept characterized by the informative material around it
- supports correct interpretation and implementation
- helps understand dependencies with other parts of model
- describes implementation typical of large, complex projects, high cost of failure

Failing to understand the above either leads to superficial processes or over-prescriptive standard processes (causing waste and frustration)

Instead, each level (required, expected, informative) serves as the starting point for the level above

- How important is that PA goal to achieving the business's objectives?
- What activities are necessary to achieving the goal?

What Agile Addresses that CMMI Doesn't

Lightweight proven approaches that work well for small co-located teams with embedded customer (e.g., XP)

- In some cases, larger teams as well (FDD, more-recent Scrum and TSP variants, and more robust versions of Crystal)

A set of "how to's" to standardize and build a process infrastructure around (OSSP, PAL, etc.)

Providing practitioners more voice in the processes they use

Providing customers more voice in the products and services they receive

Performance Results Summary

Improvements	Median	# of data points	Low	High
Cost	20%	21	3%	87%
Schedule	37%	19	2%	90%
Productivity	67%	16	11%	255%
Quality	50%	18	29%	132%
Customer Satisfaction	14%	6	-4%	55%
Return on Investment	4.8 : 1	14	2 : 1	27.7 : 1

- N = 25, as of 15 December 2005
- Organizations with results expressed as change over time

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 in the items on the right, we value the items on the left more."

Typical Agile Concepts

Iterative, incremental, and time-boxed development
Customer embedded with developer
- Tacit knowledge is the key
- Low risk when trust (with customer) is high

Continuous integration
Each increment delivers value
Test written first, then the code
Tools reverse engineer artifacts
Everyone is responsible for quality
“Fail early” attitude
Requirements development JIT
Change is “embraced”
Empowered co-located teams
Status meetings: ceremony replaced with frequent interactions
Periodically evaluate and adjust process
Rolling-wave planning

Agile Use/Misuse

A frequent form of misuse is treating the words of the Agile Manifesto or Agile Principles as absolutes rather than conditional or relative, e.g., many ignore the last line and assume things on the right have 0 value.

- Serves as “justification” for no processes, plans, designs, documents
- Can lead to chaos; and Agile opponents get distracted by a red herring

As with CMMI Interpretation, nuance appropriate to a situation is everything.

And yet thinking of the abuse mentioned earlier, is the Agile Manifesto really so unreasonable?

Agilistas Need to Know

The SEI acknowledges “level mania” and imposing processes from outside (or exclusively top-down) as misuse.

While CMMI retains its CMM roots with a focus on process, when introduced correctly, it provides transparency, and learning and reuse of what works well.

CMMI is method and tools agnostic

- Iteration is just as consistent with CMMI as is Waterfall
- CMMI & SCAMPI focus on What not How

CMMI embodies sound systems engineering and SWE principles applicable in some project contexts that Agile approaches might not be

What CMMI Addresses that Agile Doesn't*

Systems engineering (including risk management) practices that extend Agile to work in more complex situations

- Align and coordinate across teams on: larger strategy, objectives, architecture choices, interfaces, changes, and overall VER and VAL
- Maintain visibility into status, predicted completion, and risk

High maturity assets and practices that help project teams exercise greater control and prediction

An infrastructure for organizational learning and improvement

- Benefits projects even before they start
- Supports use of processes, measurement, training, and improvement
- Reduces waste

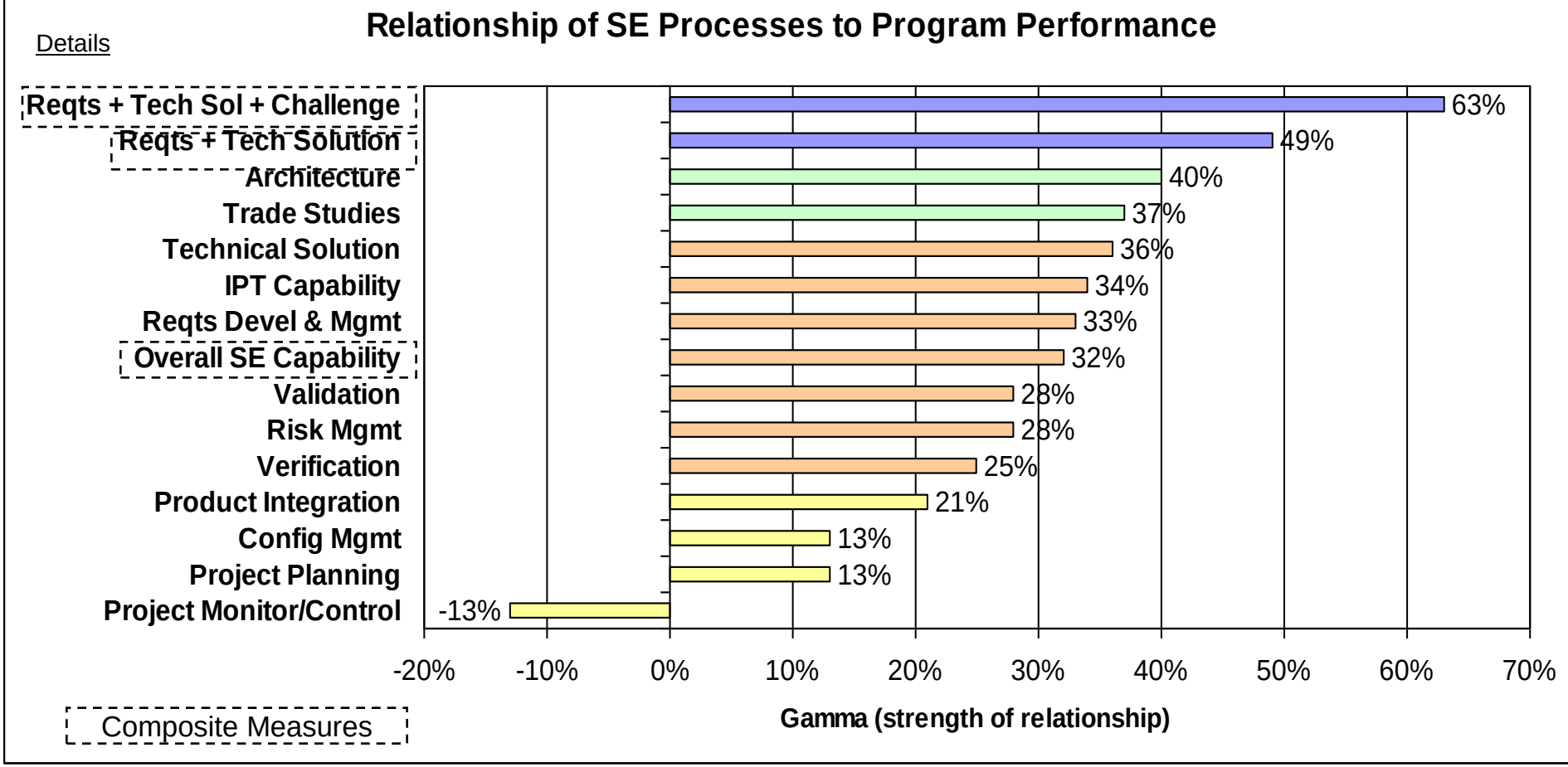
A "safety net" that helps identify gaps and lapses in attention

Helps address lack of management support and resistance to change

*Generally

A Survey of Systems Engineering Effectiveness: Initial Results

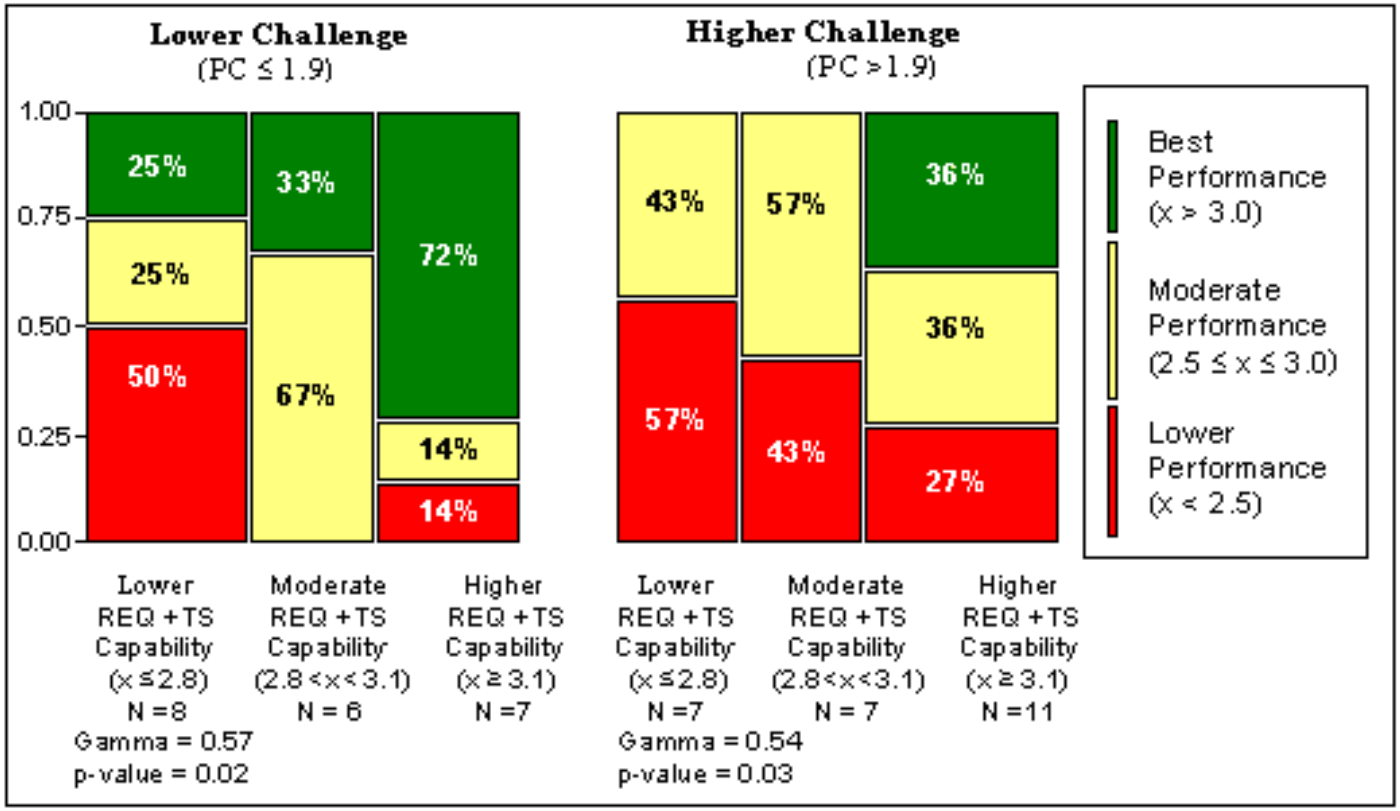
Summary of Relationships - Composite



Reqts + Tech Solution controlled by Project Challenge

Project challenge factors:

- Life cycle phases
- Project characteristics (e.g., size, effort, duration, volatility)
- Technical complexity
- Teaming relationships

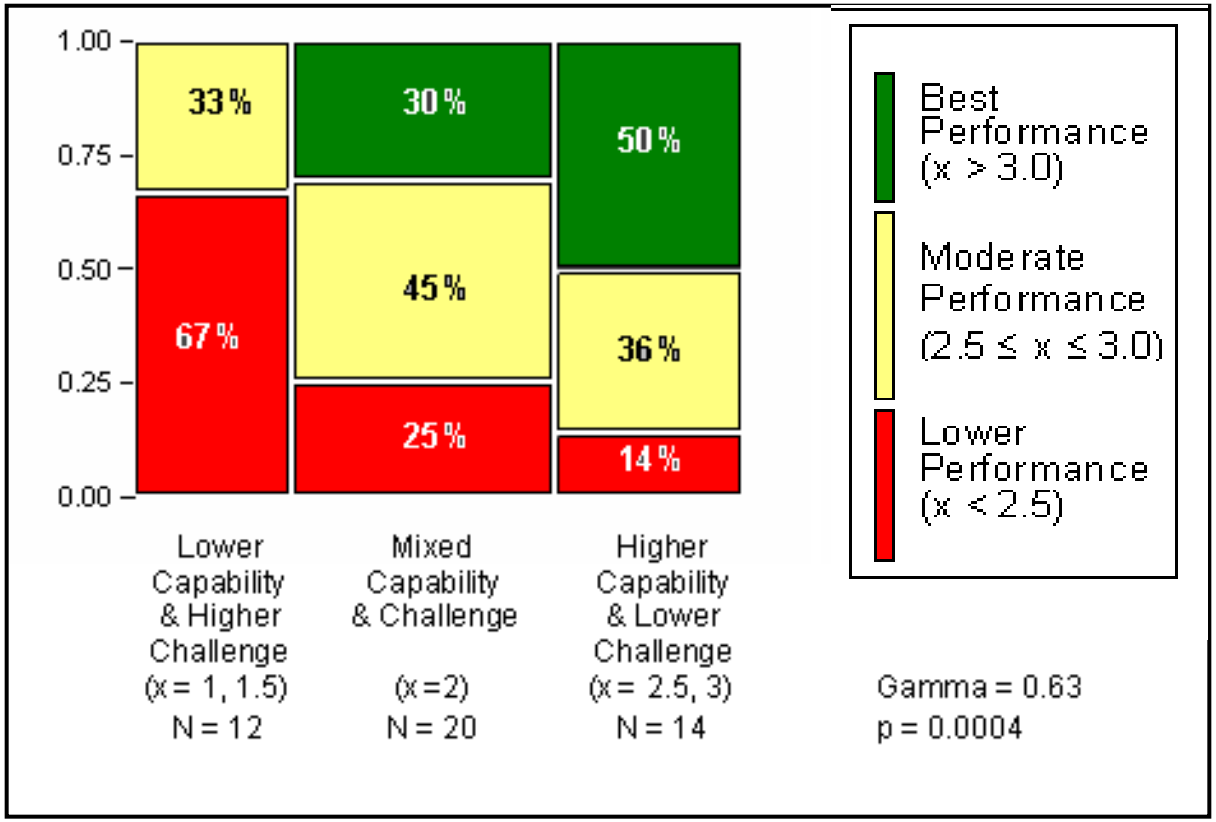


Projects with higher <u>Requirements and Technical Solution</u> capability are better able to achieve higher performance even in challenging programs

Performance vs. Reqts + Tech Solution + Project Challenge

Project challenge factors:

- Life cycle phases
- Project characteristics (e.g., size, effort, duration, volatility)
- Technical complexity
- Teaming relationships



Joseph P. Elm, Dennis R. Goldenson, et al. *A Survey of Systems Engineering Effectiveness: Initial Results.* (CMU/SEI-2007-SR-014). Software Engineering Institute, Carnegie Mellon University, 2007. http://www.sei.cmu.edu/publications/documents/07.reports/07sr014.html

Using CMMI & Agile Together

Dave Anderson: MSF® for CMMI is a superset of MSF for Agile (Agile 2005).

Jeff Sutherland presented a case of a ML 5 organization adopting Scrum and the benefits of both together providing a “magic bullet” (Agile 2007).

SEPG conference presentations have described approaches to using both:

- Characterizing in advance where Agile methods can be successfully applied (based on customer commitment, project scope, value of partial delivery, etc.)
- Employing Agile approaches for selected product components; but with traditional project management and systems engineering at the top level
- When the situation warrants, extend the usual Agile approach with systems engineering practices to improve coordination and risk management
  - Boehm & Turner (2003) suggest a risk assessment of a project’s characteristics to determine how to best balance Agility with traditional approaches

Other resources include presentations at SEPG (various locations) and CMMI Technology Conference (since 2006); and the TN described in this presentation.

Conclusions

CMMI and Agile are synergistic

- Share similar goals (happier customer, superior execution, capable organization)
- Each addresses principles of good software engineering
- Each addresses something the other lacks

For small co-located teams, committed customer, low cost-of-failure (CoF)

- Use Agile for product development (SEI recommends you consider the TSP)
- Use CMMI to provide a supporting organizational environment

For large complex projects with geographically and organizationally-dispersed teams (“high-challenge programs”)

- Use Agile approaches proven to work in such contexts
- But ensure adequate coverage of Systems engineering practices found in CMMI

CMMI-based process improvement can alienate practitioners

- Concurrent introduction of Agile practices can mitigate

Agile approaches may not fully scale up or deploy broadly

- At a minimum, CMMI provides “safety net;” a set of practices that help close the gap

What You Can Do

CMMI proponents should become more familiar with the principles of Agile development

- empowered teams
- continuous customer engagement
- time box-based iteration
- continuous integration
- test-driven development

that in the right situation bring about rapid *emergence* of the right product or service concept and highly-motivated teams.

Agile proponents should become more familiar with CMMI practices that address:

- non-functional requirements
- product architecture
- risk management
- organizational learning

that bring about *predictability* of what product capabilities can actually be achieved

Contact Information for anything on this presentation

Mike Konrad

Telephone: +1 412-268-5813

Email: mdk@sei.cmu.edu

U.S. mail:

Software Engineering Institute

Customer Relations

4500 Fifth Avenue

Pittsburgh, PA 15213-2612

USA

World Wide Web:

www.sei.cmu.edu

www.sei.cmu.edu/contact.html

Customer Relations

Email: customer-relations@sei.cmu.edu

Telephone: +1 412-268-5800

SEI Phone: +1 412-268-5800

SEI Fax: +1 412-268-6257

BACKUP SLIDES

A History of CMMI

CMMI has its roots in the work of Shewart in application of statistical process control to manufacturing and that of Deming on project and process management.

In the 1970s, Watts Humphrey and Ron Radice at IBM extended the application of these ideas to software development.

In early 1980s, Watts Humphrey, inspired by Phil Crosby's book *Quality is Free*, formulated a maturity framework for software development, as a way to organize the application of these ideas and help senior management set targets for process improvement.

In 1984, the US DoD, because of delays, cost overruns and quality problems in software developed as part of safety-critical systems, awards CMU a contract to establish the SEI

CMM for Software in 1991. Other models quickly follow.

A major effort was undertaken beginning in 1998 to find common ground between systems engineering and software engineering, culminating in CMMI.

Maturity levels become entry criteria in some markets (particularly the U.S. DoD) and are assumed to promise successful contractor performance

All this largely predates Web, open source, mobile communications, and millions of new software developers who've not had this history

Through multiple releases, CMMI evolves to incorporate improved best practices

Though language and examples are now clearer and applicable to a broader range of contexts, a bias towards traditional development approaches remains.

A History of Agile

Roots can be traced back to Deming and PDSA

NASA and USAF utilized time-boxed, iterative and incremental design development (IIDD) back in the 1960s

But era of mainframes, COBOL, fixed-price contracting became dominant

Tom Gilb in 1976 argued the merits of evolutionary development

Barry Boehm in 1985 published “The Spiral Model of Software Development…”

The above set the stage for a resurgence in interest in IIDD in the 1990s

- rapid prototyping and RAD

XP invented at Chrysler in 1996 by Ron Jeffries and Kent Beck

Lightweight methods begin to proliferate

Meeting of the minds in Utah led to “Manifesto for Agile Software Development”

- Agile Principles, Agile Alliance, annual Agile conferences, DoI, APLN established

Recently, broadening to enterprise adoption, supporting tools, scaling up to larger projects, and non-software application

Contrasting CMMI and Agile[1]

	CMMI	*Agile*
Focus	Organization and Project; What	Project; How
Time frame	Long-term	Short-term
Cost of Failure	Developed for a domain where cost of failure can be very high	Domain with low cost of failure (web design, game industry, social networking)
Management	Focuses on the broad responsibility for management; not who	Manager seen as Coach, modeling Agile values
Trust	Does not presume opaqueness nor transparency in dealings with stakeholders	Hallmark assumption is high-trust relationship with customer
Planning	Not specific how it is done, but size, cost, schedule estimates with milestones assumed	Release planning; what features to put in next iteration; Gantt charts are anathema
Market assumption	Broadly applicable; but in mature market, process innovation is critical	Works best in emerging, not well-understood market
Design	Examples generally favor early design, revisited as more is learned	Avoid premature "lock in;" but benefit from using standard architecture

Contrasting CMMI and Agile[2]

	CMMI	*Agile*
Learning	(1)Organizational based on knowledge & skill needs (2)Development activities (3)Process management (4)CAR and High Maturity	(1)Within development: dynamic, iteration to iteration, change is embraced (2)Project retrospectives
People	Focus on process as liberator	Focus on people to determine their process; assumes people already mature
Life cycle	Agnostic to life cycle	Assumes incremental delivery is viable
Testing	Review-as-you-go; validate as needed (but can be often)	Fully test each iteration; validate often ("fail early")
Predictability	Exploit understanding of behavior (SPC at event level with statistical modeling to span time) to predict quality and process performance	Develop understanding of velocity to predict iteration scope and provide better control

Limitations to Research on CMMI & Agile

Anecdotes vs. systematic empirical investigation

Concepts being studied are abstract (different ways to implement)

Some approaches work well with some teams in some situations but not others

Human nature: quick to declare success or failure, over-generalize

Problems in human perception and learning (memory and experience jaded by our filters, reporting biases)

Intangibles (e.g., people factors, measurement system error often unknown)

Failure to distinguish important differences in concepts

- CMMI versus “Traditional development”
- Agile versus “Hacking”

Sometimes the benefits are real but accrue gradually