Carnegie Mellon
Software Engineering Institute

Pittsburgh, PA 15213-3890

Reflections on Software Agility and Agile Methods: Challenges, Dilemmas, & the Way Ahead

Linda Levine
Software Engineering Institute

May 11, 2005

Sponsored by the U.S. Department of Defense

Carnegie Mellon
Software Engineering Institute

Agenda

Defining agile

Research study: context and questions

What we have learned so far: The current state
- Case studies of Internet Software Development
- Discovery Colloquium

Where we're going: The future state
- propagating agile approaches, scaling, self-organizing systems (i.e., nonlinear, adaptive)

Challenges, dilemmas, and conundrums
- process, discipline, and governance

Defining Agile

Merriam Webster (2004) defines agile from the Middle French and from the Latin *agilis*, from *agere* to drive, act as "**1:** marked by ready ability to move with quick easy grace [and] **2:** having a quick resourceful and adaptable character <an *agile* mind>." Agility is defined as "the quality or state of being agile: NIMBLENESS, DEXTERITY <played with increasing *agility*>."

Four key attributes

- *speed*: quick, fast
- *nimble*: able to improvise, use patterns creatively to construct new solutions on the fly, flexible
- *adaptable*: responsive (sense and respond), dynamic and interactive in response to a customer or to changing circumstances.
- *resourceful*: thoughtful or exhibiting some discipline (Note: not the same as traditional "command and control" approach with defined, formal procedures)

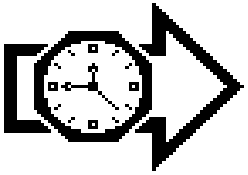
The Agile Manifesto says….

- Top level:
  "We are uncovering better ways of developing software by doing it and helping others do it."
- 2nd level:
  "Through this work, we have come to value…

Individuals and actions	over	**processes and tools.**
Working software	over	**comprehensive documentation.**
Customer collaboration	over	**contract negotiation.**
Responding to change	over	**following a plan.**

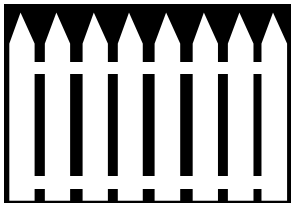
That is, while there is value in the items on the right, we value the items on the left more." *www.AgileAlliance.org*

Carnegie Mellon
Software Engineering Institute



practiced

Well-defined gates



simple

flexible

collaborative

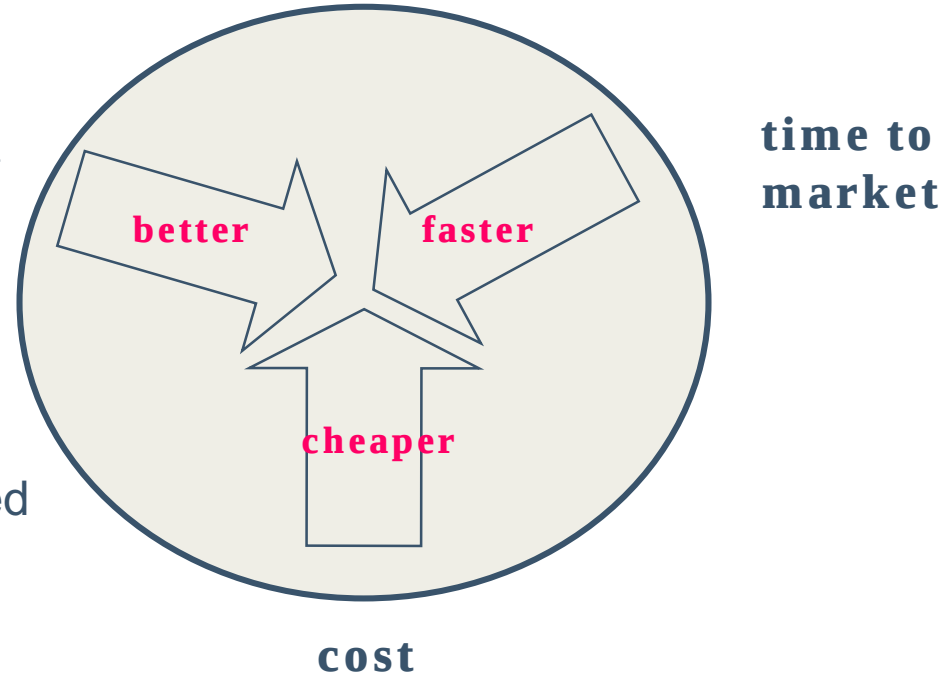
The Current Climate

Formula for gaining and controlling market share is changing

- In the past, a company positioned itself along a single dimension.
- Now, competitive edge goes to companies that deliver better products, faster *and* cheaper.

Accelerated by Internet software development

In the digital economy, separation between business & the software engineering system has closed



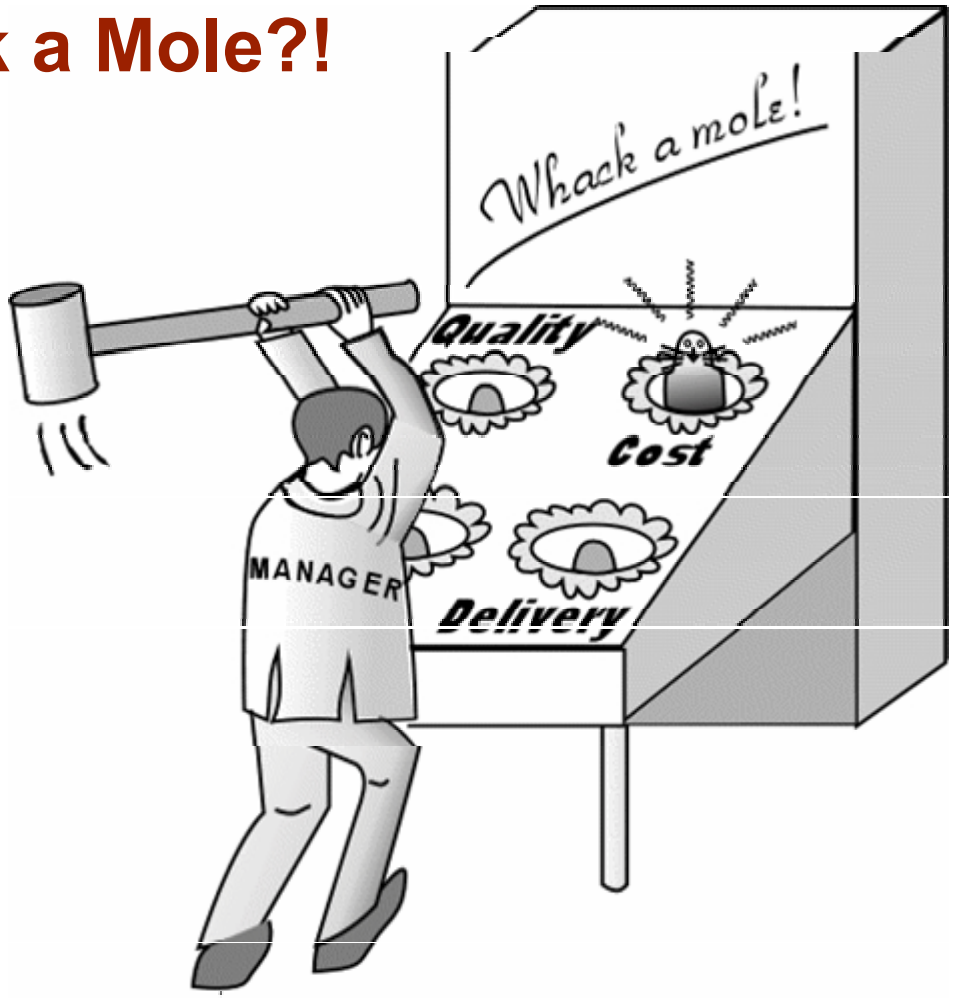
Carnegie Mellon
Software Engineering Institute

What does “Internet speed” mean anyway?

“The pressure to release new software products faster and faster has grown over the years ... ten years ago, development cycles of 24-36 months were typical, today, a year to 18 months development cycle is normal for software products ...
Within “emerging fields such as electronic commerce and Web portal sites competing on Internet time demand significant product and feature changes every three to six months”

Cusumano, 1998

Whack a Mole?!



Source: Michel Baudin, 3/16/99. Lean production: the end of management whack-a-mole. http://www.mmt-inst.com/End_of_management_whack_a_mole.html

Carnegie Mellon
Software Engineering Institute

Key Research Questions

How do firms develop fast cycle time software?

Is “Internet speed” software development really different from traditional software development?

How can both quality and agility be achieved in fast-paced software development?

What development practices are effective in this rapid pace environment?

Carnegie Mellon
Software Engineering Institute

Research Team

Richard Baskerville and Balasubramanian Ramesh
Department of Computer Information Systems,
Georgia State University

Linda Levine
Software Engineering Institute,
Carnegie Mellon University

Jan Pries-Heje
The IT University of Copenhagen

Sandra Slaughter
Graduate School of Industrial Administration,
Carnegie Mellon University

Carnegie Mellon
Software Engineering Institute

A Multi-Phase Approach

PHASE 1: Case studies conducted in nine firms

PHASE 2: Discovery colloquium to synthesize knowledge on principles and practices

PHASE 3: Longitudinal revisit to firms after two years

The firms range in size from 10 employees to more than 300,000

- new Internet software dot.coms & established brick & mortar firms
- private & public sectors:

  - software houses
  - courier services
  - travel
  - utilities
  - financial services and insurance
  - business & consulting services
  - media

Carnegie Mellon
Software Engineering Institute

Phase 1: Detailed Interviews

With:

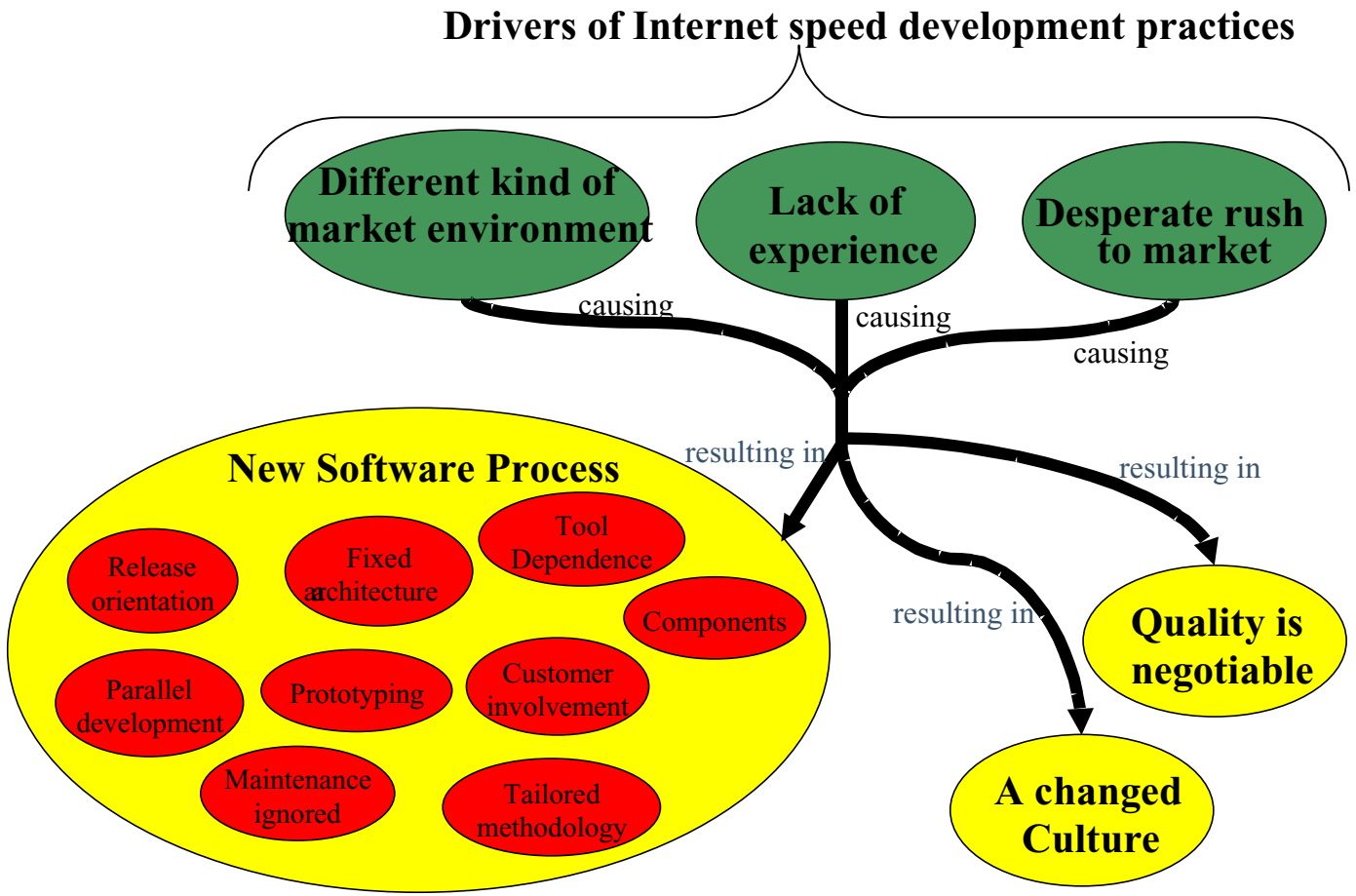
- senior managers, project managers
- software engineers, QA engineers

Questions on:

- demographics on organization & interviewees
- “Internet speed”
- products & business strategy
- quality
- teaming & people
- development methods & tools
- issues, problems & challenges

Carnegie Mellon
Software Engineering Institute

Phase 1: Results



Carnegie Mellon
Software Engineering Institute

Phase 2: Discovery Colloquium

Rich mix of participants; maximum exchange of ideas

Leverage forward-looking methodologies

- "Future search" approaches to system learning
  - form of action research
  - "bring the whole system in the room"

- "Difference questioning"
  - identify relevant differences among participants
  - "when members of a group question differences (they) generate new information" (Goldstein, 1994)

- "Creative abrasion"
  - "creates a collision of ideas within a climate of social cohesion"
  - result: original and innovative ideas (Leonard, 1995)

Carnegie Mellon
Software Engineering Institute

Phase 2: Colloquium Approach

Process

- inclusion exercise: metaphors
- “talking circles”
- breakout groups: on core issues

1. **Hypothesis Testing**
   - develop hypotheses, select focus

2. **Difference Questioning**
   - assumption surfacing
   - identify principles & promising practices (patterns of commonality or difference across the assumptions)

3. **Futuring, scenario development**
   - emerging conditions & impacts

Facilitators and scribes

Analysis and Results

- materials transcribed
- transcripts analyzed
- technical report drafted, distributed
- feedback solicited

Findings emerge

- participants’ insights
- post-colloquium comparison of principles underlying agile and traditional methods

Carnegie Mellon
Software Engineering Institute

Breakout Group: What is different about agile methods?

General observations

- popular claims that agile methods are radically new and the best approach to Internet software development
- key elements
  - collaborative work
  - incremental development
  - evolutionary life cycles within a strategy
  - strong customer communication
- many of these elements embodied in eXtreme Programming
- effectiveness lies in people, not process – heroes rewarded

Hypotheses: Agile Methods

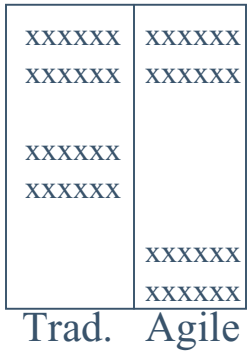
H1. Agile methods are more effective than traditional methods.
H2. Agile methods are not different than traditional methods.
H3. Partially implementing key agile practices will lead to project failure.
H4. Agile methods require good people to be successful.
H5. Agile methods are needed for Internet speed development because it is fundamentally different from traditional.
H6. Agile methods are effective when the time horizon is short, and not as effective over the long term.
H7. Agile methods aren't really new *per se*, but their implementation is extreme.

Comparing Agile & Traditional Principles

What principles overlap?
- common principles across agile and traditional methods

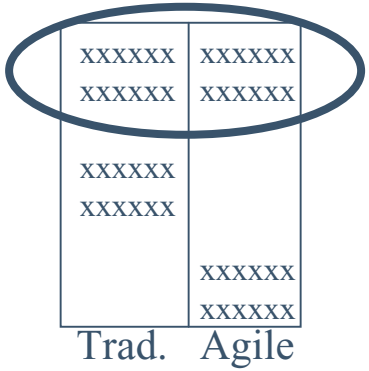
What agile principles are missing?
- those traditional principles with no apparent agile equivalent

What traditional principles are missing?
- those agile principles with no apparent equivalent traditional principle



Overlapping Principles

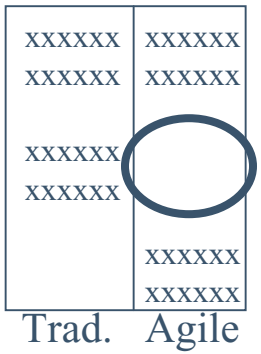
Flexibility
Understanding functional requirements
Responding to change
Learning from experience



Missing Principles

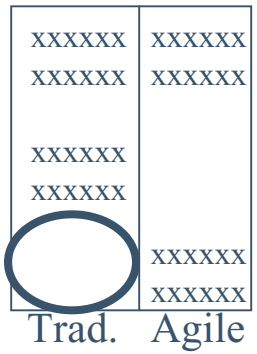
Agile Principles set is missing:

- quantitative measurement
- interchangeable components
- complexity & uncertainty control
- rigorous requirements specification
- formal quality management
- documentation
- component coupling
- stepwise assembly
- disciplined process



Traditional Principles set is missing:

- teamwork & on-the-fly software process adaptation
- informal knowledge exchange
- collaboration & experience
- tailoring project practices to environmental conditions



Carnegie Mellon
Software Engineering Institute

Phase 2: Summary

- Many principles of traditional methods consistent with an industrial production paradigm, or software “factory”
- Agile principles include many features of a “job shop” environment
- Agile software development occurs in a more informal, dynamic, learning environment. Agile methods support shorter project life cycles in order to respond to complex, fast-moving, and competitive marketplaces.
- Agile methods have emerged and are used by some organizations for fast-paced software development
  - some organizations also continue to develop software using traditional methods
  - some use BOTH agile methods and traditional methods
  - parallel paradigms for software development

Carnegie Mellon
Software Engineering Institute

Phase 3: Case Study Continues

Two years later, only five of the original companies remained in business or were available to participate in the study. Only one of the small Internet software houses had survived.

To maintain the representative nature of the companies, we added an additional company—a small innovative Internet software house. In all, six companies participated in Phase 3.

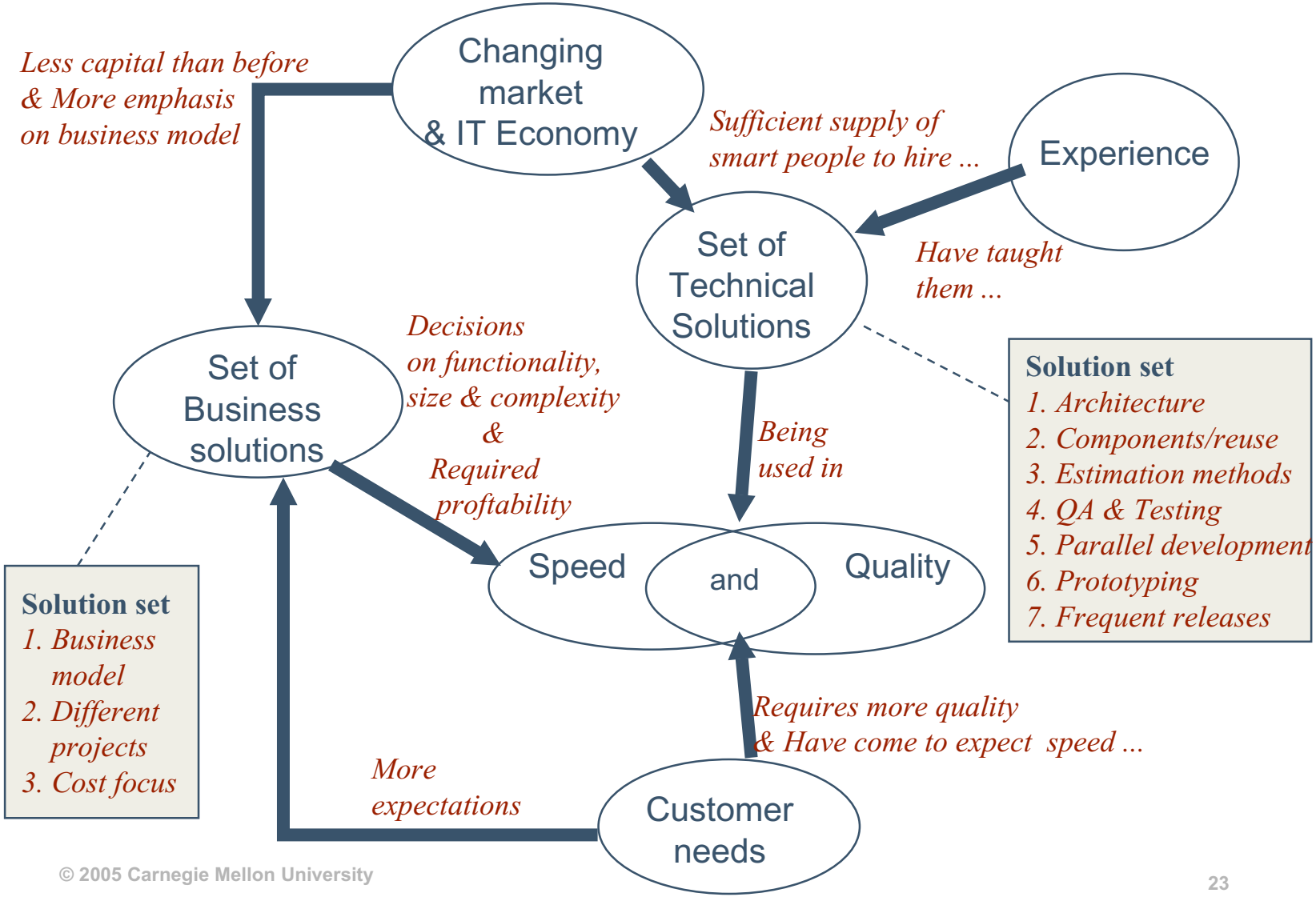
Re-interviewed managers and staff using same questions from Phase 1 on strategies, methods, tools, issues, etc.

Compared what has changed and what has not

- confirmation of which Internet speed practices prevail
- insight into how environmental contingencies impact the choice and effectiveness of Internet speed practices

Carnegie Mellon
Software Engineering Institute

Phase 3: Results



Carnegie Mellon
Software Engineering Institute

Phase 3: Summary

Trade-offs and balancing decisions—a high-speed balancing game—were taking place at three different levels

Market: IT economy slowed the interest in IT products, easing the intense competition for human resources. Consolidation of best practice.

Portfolio: *Business case* became the primary vehicle for selecting projects for inclusion or continuation. Managers "cherry pick" the most ideal projects to meet their customers' needs.

Project: Project managers consolidate product development to embrace construction of fewer products. Major values persist, such as parallel development, limited maintenance & documentation, frequent releases & other factors

- still necessary to maintain customer satisfaction and compete
- also noted for enabling quick, economical products.

Carnegie Mellon
Software Engineering Institute

Where are we going? The Future State

- Use of agile methods and agility is consistently associated with software development techniques.

- Fledgling signs of expansion
  - contracting of the market and tightening of resources has contributed to increased complexity in the balancing game
  - may spur further growth for agile approaches in atypical areas

- Current state for agile methods is still isolated and limited
  - partial understanding of what agility means for software development activities.
  - best insights still achieved through discrete activities—through projects which exist like *islands* in our organizations
  - development, adoption, knowledge transfer

The Future

Optimize the current state. Loosely integrate and propagate agile approaches.

- business & technical
- leverage what we know; reinforce discrete areas of success

More radically, tackle the issue of scaling to investigate options and opportunities that can span organizations.

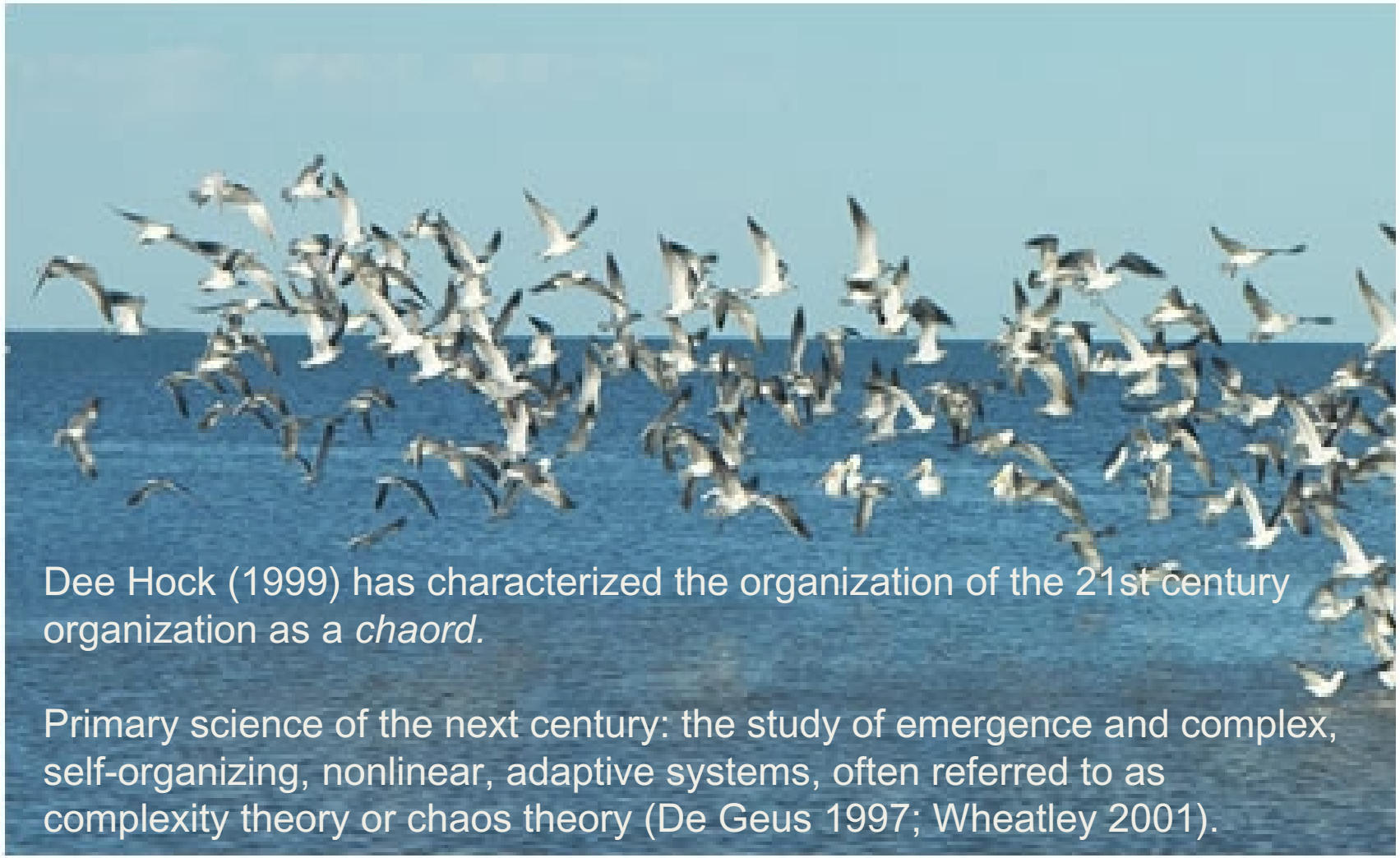
- ask how such methods adapt and scale

Austin and Devin (2003) speculate that old production models for software development are no longer useful. Rather, agile software development has the potential to be “artful making.”

They build a framework using the analogies of theatrical production, extending beyond surface collaboration to the on-cue innovation that theater companies routinely achieve.

Carnegie Mellon
Software Engineering Institute

The Future: Emergence



Dee Hock (1999) has characterized the organization of the 21st century organization as a *chaord.*

Primary science of the next century: the study of emergence and complex, self-organizing, nonlinear, adaptive systems, often referred to as complexity theory or chaos theory (De Geus 1997; Wheatley 2001).

Living systems "arise and thrive on the edge of chaos with just enough order to give them pattern, but not so much to slow their adaptation and learning."

This resembles the challenge for agility.

Does this represent the larger paradigm shift of which agile methods are a part?

Emerging Systems: The Interconnectedness of All Things
http://www.imaginify.org/complexitygallery/complexity.html

Challenges, Dilemmas, & Conundrums

Achieving the future state is a challenge in itself—enhancing, adapting, applying, and scaling agile approaches

Areas of controversy

- process
- discipline
- governance

Carnegie Mellon
Software Engineering Institute

Process

Agile methods vs. process-intensive or *monumental* models like the SW-CMM® framework (Highsmith 2000).

Paulk (2001) looks at how such approaches are not entirely at odds and illustrates how a development group following extreme programming might simultaneously embrace CMM, at least up until Level 3. At Level 3, the approaches diverge.

Boehm (2002) and Boehm and Turner (2003) argue that agile and plan-driven methods each have a "home ground." They emphasize balance and attempt to make a case for hybrid strategies.

The split between process and agility has become a lightning rod, reinforcing entrenched positions and a strict drawing of lines.

Misconceptions about Process

I don't need process

I have
- really good people
- advanced technology
- an experienced manager

Process
- interferes with creativity
- = bureaucracy + regimentation
- hinders agility in fast-moving markets
- is only useful on large projects

. . . It will wrinkle m

Carnegie Mellon
Software Engineering Institute

Agile or Plan-Driven?

Agile Methods

- rely on tacit knowledge embodied on team
- premium developers
- dedicated customers w/ knowledge of full span of application
- turbulent environment, constant change
- requirements are emergent
- tightly coordinated teamwork needed to succeed becomes increasingly difficult beyond 15 or 20 (Constantine 2001)

Plan-driven Methods

- invest in process & product plans; major milestones
- compensate for customer shortfalls via use of architecture review boards and independent expert project reviews
- requirements can be determined in advance, or via prototyping; requirements remain relatively stable
- vital for stable, safety critical embedded software

Discipline

Rakitin (2001) takes a skeptical view. He sees values on the right as essential, while those on the left serve as easy excuses for irresponsibly throwing code together.

Individuals and interactions over processes and tools
Translation: Talking to people gives us the flexibility to do whatever we want in whatever way we want to do it. Of course, it's understood that we know what you want—even if you don't.

Working software over comprehensive documentation
Translation: We want to spend all our time coding. Real programmers don't write documentation.

Customer collaboration over contract negotiation
Translation: Let's not spend time haggling over the details, it only interferes with our ability to spend all our time coding. We'll work out the kinks once we deliver something.

Responding to change over following a plan
Translation: Following a plan implies we would have to spend time thinking about the problem and how we might actually solve it. Why would we want to do that when we could be coding?

Carnegie Mellon
Software Engineering Institute

Discipline

Agile proponents see CMM framework as engendering bureaucratic, prescriptive processes, fostering a command and control environment

More subtle definitions of discipline have not yet been brought to bear.

Under the auspices of agility, there must be some structure, order and organization.

- We know that, in actuality, it takes time to speed up, unless you are simply cutting things out. (Smith & Reinertsen 1998)
- By extension, it takes discipline to be agile.

There are new approaches to experimentation and frameworks such as *artful making*—where the emphasis is on a method of control that accepts wide variation within known parameters.

What kind of discipline contributes in the agile environment?

Governance

Leadership vs. Management

- "No one has yet figured out how to manage people effectively into battle; they must be led" --John Kotter, What Leaders Really Do
- Leadership is about helping people cope with change, while management is about coping with complexity. Leaders *set direction*, managers *plan and budget*. Leaders *align people*, managers *organize and staff*. Leaders *motivate*, managers *control*.
- *Shusa*, or scrum master vs. traditional manager

We are faced with conflicting models—one for development which can be agile, and one for project management, for oversight, and monitoring.

Governance

Development vs. Acquisition

Acquisition program managers have expressed interest in their development teams using agile methods. However, they are at a loss to identify appropriate mechanisms that could be employed for monitoring and oversight of systems development.

It is naïve to assume that oversight is antithetical to agile approaches. Once again, we are challenged to reach beyond comfortable and convenient walls to explore new territory.

Conclusion

Speed, Quality, Processes, Technical Solutions, Principles, Business Solutions, and Balance

Agility in software development has implications for organizational agility. The shift to agile methods and models signals a larger transformation in the workplace and the organization of the 21st century.

This transition state is turbulent, marked by continuous change. No clear or easy solutions have resulted.

The transformation is a work in progress, and by no means complete. To be realized, it invites investigation across a range of disciplines and initiatives.

Carnegie Mellon
Software Engineering Institute

References

Agile Manifesto. http://agilemanifesto.org/

Baskerville, R., Pries-Heje, J., Levine, L., & Ramesh, B. (2005). The high speed balancing game: How software companies cope with Internet speed. *Scandinavian Journal of Information Systems 16*, 11-54.

Baskerville, R., Levine, L., Pries-Heje, J., Ramesh, B., & Slaughter, S. (2003), Is Internet-speed software development different? *IEEE Software 20* (6), Nov-Dec, 70-77.

Baskerville, R., Levine, L., Pries-Heje, J., Ramesh, B., & Slaughter, S. (2001) How Internet software companies negotiate quality. *Computer 34* (5), May, 51-57.

Carnegie Mellon
Software Engineering Institute

References

Boehm, Barry. (2002). Agile and plan-driven methods: Oil and water?, USC, Agile Universe 2002, August 5, 2002. http://www.xpuniverse.com/pdfs/agileAndPlanDrivenMethods

Boehm, Barry. (2002). Get ready for agile methods, with care. *IEEE Computer*, 64-69

Bourque, P. et al., (2002). Fundamental principles of software engineering—A journey, *J. Systems and Software*, v. 62(1), May,59–70.

Constantine, L. (2001). Methodological agility, *Software Development*, June 2001, 67-69.

Cusumano, M.A., & Yoffie, D.B. (1999). Software Development on Internet Time, *Computer*, vol. 32 (10), Oct., 60–69.

Carnegie Mellon
Software Engineering Institute

References

Cusumano, M.A., & Yoffie, D.B. (1999). What Netscape learned from cross-platform software development, *Comm. ACM*, v42 (10), Oct., 72–78.

Iansiti M. & MacCormack, A. (1997). Developing products on Internet time," *Harvard Business Rev.*, v.75 (5), Sept./Oct.,108–117.

Levine, L., Baskerville, R., Loveland Link, J.L., Pries-Heje, J., Ramesh, B., & Slaughter, S., (2002). Discovery colloquium: Quality software development @ Internet speed. (SEI Technical Report CMU/SEI-2002-TR-020) Pittsburgh, PA.; Software Engineering Institute.

Miner, A. S., Bassoff, P., & Moorman, C. (2001). Organizational improvisation and learning: A Field study. Administrative Science Quarterly, 46 (June 2001), pp. 304-337.

Carnegie Mellon
Software Engineering Institute

References

Paulk, M. (2001). Extreme Programming from a CMM perspective. *IEEE Software*, 18 (6),19-26.

Smith, Preston G. & Reinertsen, Donald. (1998). *Developing products in half the time*. New York: John Wiley & Sons.

Agile Approaches

Jim Highsmith identifies seven agile software development ecosystems

1. Scrum
2. DSDM (Dynamic Systems Development Method)
3. Crystal Methods
4. FDD (Feature Driven Development)
5. Lean Development
6. XP (eXtreme Programming)
7. Adaptive Software Development

Agile Methods (Theoretical)

The amount of specific vs. general guidance is key: Scrum, DSDM, FDD, and XP give specific guidance. Lean Development, Adaptive Software Development, and Crystal Methods present a theoretical basis for agile practices.

Theoretical methods

- Crystal concentrates on communication and varying practices based on project size and risk.
- Agile Software Development focuses on emergence
- Lean Development emphasizes traditional lean concepts of value and flow.
- All three emphasize the primary importance of the development team. None of these three methods focus on specific practices, so they do not find themselves in conflict with the four agile methods that offer more specific practices, or with each other.

Other Agile Methods

- Scrum, DSDM, FDD, and XP offer specific practices, generally expected to be followed as a package until one is expert enough in the method to modify practices.
- DSDM and Scrum focus on project management and are agnostic about the underlying technical approach. Both emphasize business value, fixed time boxes, significant customer involvement, developing high priority items first, and stopping when you run out of time.
- DSDM is the higher ceremony approach, while Scrum spends less time on project initiation activities.
- FDD is a complete approach covering both technical issues and project management. It is different from the other agile practices because it does not advocate common code ownership, but rather assigns class ownership. This rather large technical difference underlies many of its other differences from other agile practices. Because of its comprehensiveness and different technical approach, FDD would not co-exist well with other agile approaches, but would be used separately.

Carnegie Mellon
Software Engineering Institute

Other Agile Methods -2

- Scrum and XP are often used together because their practices are more or less disjoint:
- Scrum focuses on project management while XP focuses on developer practices. Both of these approaches are often billed as a complete set of practices that are supposed to be used without modification, at least until the users become skilled enough to make appropriate changes.
- However, when used together, a few differences have to be resolved, including recommended iteration length, specific planning details including release planning and stories or backlog items.
- For all of their differences, one interesting difference is the way leadership is viewed in the various practices.

Agile Ecosystems **A Position Paper for the Workshop: Are Agile Methodologies Really Different? Mary Poppendieck.**
http://www.coldewey.com/publikationen/conferences/oopsla2003/MaryPoppendieck.pdf

Boehm's Planning Spectrum

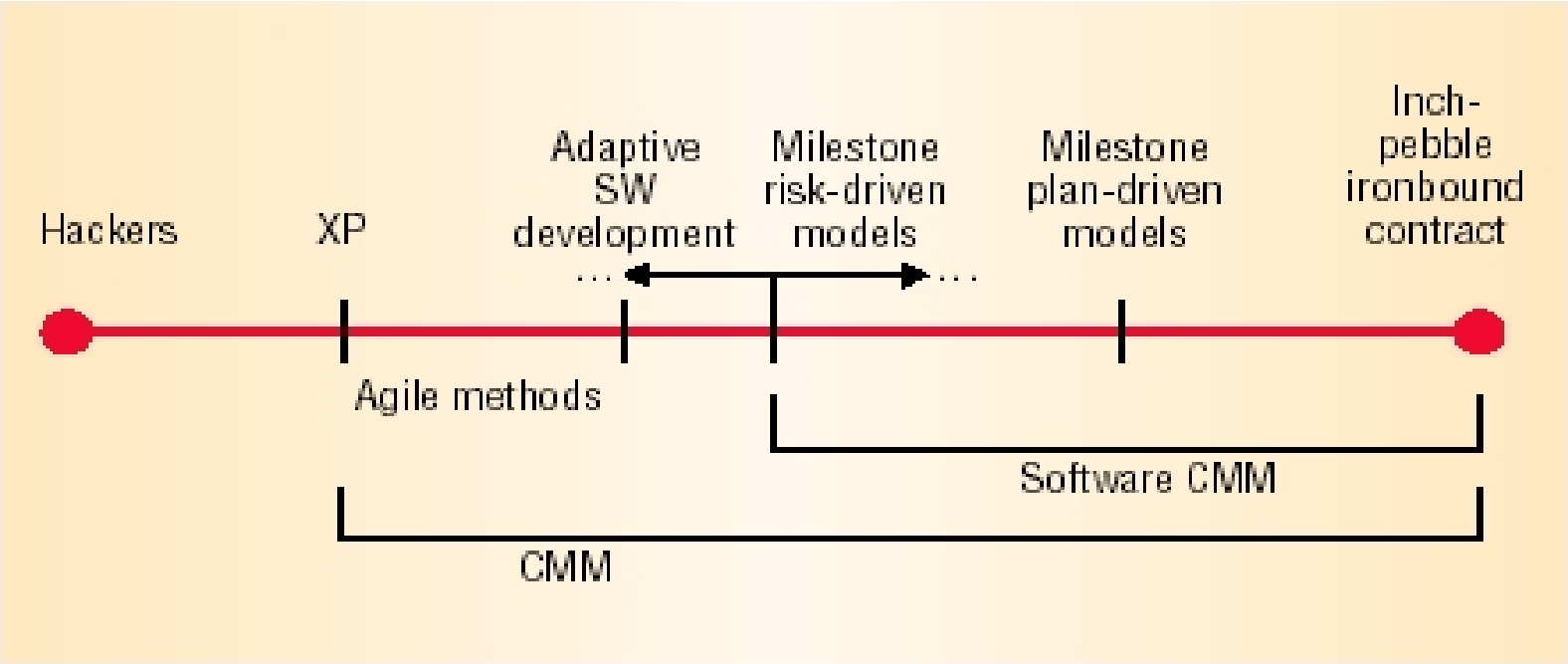


Figure 1. The planning spectrum. Unplanned and undisciplined hacking occupies the extreme left, while micromanaged milestone planning, also known as inch-pebble planning, occupies the extreme right.

Boehm, Barry. (2202) Get ready for agile methods, with care. IEEE Computer, pp. 64-69

Carnegie Mellon
Software Engineering Institute

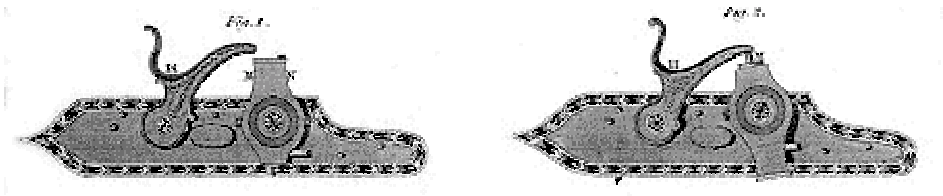
What is Process?

Process n. 1. a continuing development involving many changes 2. a particular method for doing something, usually involving a number of steps or operations. *(Webster's, 1976)*

IEEE — a sequence of steps performed for a given purpose

CMM — a set of activities, methods, practices, and transformations that people use to develop and maintain software and the associated products

Why do we need process?



Enables
- repeatability
- insight, oversight
- control, tracking
- measurement
- improvement
- training

Interchangeable parts

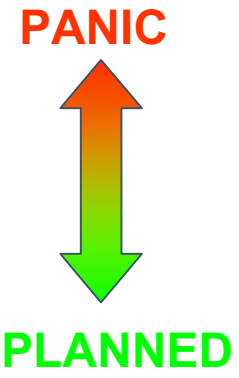
Interchangeable parts

Carnegie Mellon
Software Engineering Institute

Behavior under Stress

In times of stress, Process may be:

- abandoned
- abbreviated
- tailored



Carnegie Mellon
Software Engineering Institute

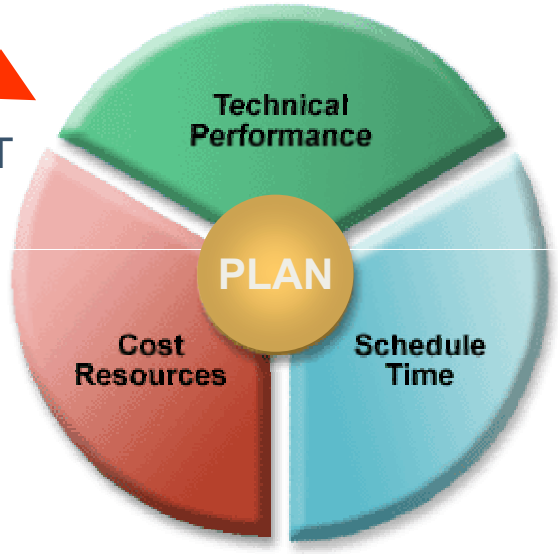
Behavior under Stress

REACT

The difference between PANIC and PLANNING is understanding risks.

First plan, then build, then document the process.
Only then can you tailor a process

ACT



Tailoring is the result of thoughtful abbreviation of processes.

Tailoring takes into account:

- Tradeoffs: identify what program goals are more critical
- Contingency planning: work out what to do under various conditions of program stress

Carnegie Mellon
Software Engineering Institute

Software Engineering Institute (SEI) Overview

DoD R&D Laboratory FFRDC

Created in 1984, based on a recommendation of a DoD Joint Task Force, chaired by the Deputy Under Secretary of Defense (R&AT)

Mission: Advance the state of the practice of software engineering and software-intensive systems acquisition

Location: A college-level unit of Carnegie Mellon University with principal locations in Pittsburgh, PA and Arlington, VA