
CrossTalk
April 2007 The Journal of Defense Software Engineering Vol. 20 No. 4
The Plan is the Problem!
Deep Agile
Software Development
Jeff Sutherland, Ph.D.
Co-Creator of Scrum
http://jeffsutherland.com/scrum
AGILE
DEVELOPMENT

Jeff Sutherland background

jeffsutherland.com/scrum

- Agile Systems Architect since 1986
  - CTO/VP Engineering for 9 software companies
  - Prototyped Scrum in 4 companies
  - Conceived and executed first Scrum at Easel Corp. in 1993
  - Rolled out Scrum in next 5 companies
- Signatory of Agile Manifesto and founder of Agile Alliance
- ScrumMaster Certification World Tour

Jan 10-12 Medco
Jan 25-26 CSM Boston
Jan 29-31 Ulticom
Feb 1-2 CSM Philips Medical
Feb 7-8 CSM Aarhus
Feb 15-16 Constant Contact
Feb 22-23 CSM Ft Myers
Mar 1-2 CSM St. Petersburg, Russia
Mar 5-6 CSM Copenhagen
Mar 8-9 CSM Copenhagen
March 12-13 CSM QCON London
March 22-23 CSM Boston
Apr 3-4 CSM Amsterdam
Apr 18-19 CSM Boston
Apr 25-26 CSM Aarhus
Apr 28-29 Deep Agile with Ron Jeffries Boston
May 7-11 Scrum Gathering Portland
May 17-18 CSM Boston
May 23-24 Scrum Tuning Ft. Meyers
Jun 4-5 CSM Systematic
Jun 6-7 CSM Aarhus
Jun 11-12 CSM Copenhagen
Jun 13 Oresund Agile Copenhagen
Jun 18-19 CSM Oslo
Jun 20-21 CSM Stockholm

Agile Alliance founders

Plan driven development

- High failure rate
- Produces software that sucks
  - Fails to fit customer needs
  - High defect rate
- Over 50% waste
- Delays time to market
- Poor working environment

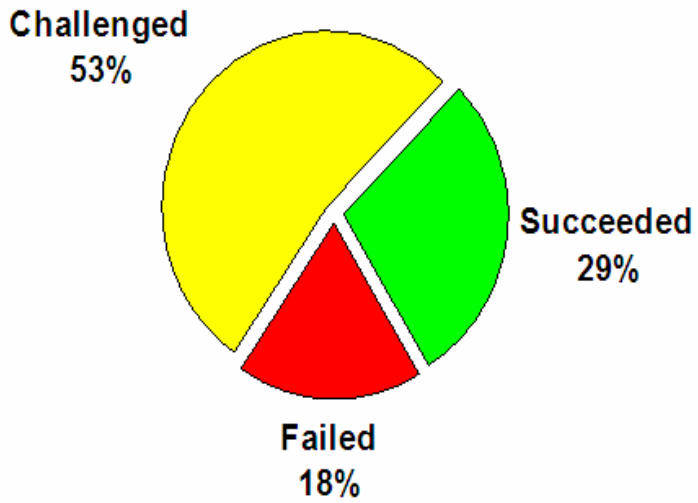
Value driven development

- High success rates
- Produces software that meets customer needs
- Minimal waste
- Accelerates early revenue
- Energized working environment

Plan based failure rates

Projects fail (CHAOS report 2004: Standish group)

Project Size	People	Time (months)	Success Rate
Less than $750K	6	6	55%
$750K to $1.5M	12	9	33%
$1.5M to $3M	25	12	25%
$3M to $6M	40	18	15%
$6M to $10M	+250	+24	8%
Over $10M	+500	+36	0%



Mainly caused by lack of:

- User Involvement,
- Executive Support and
- Clear Business Objectives.

Unplanned software case study

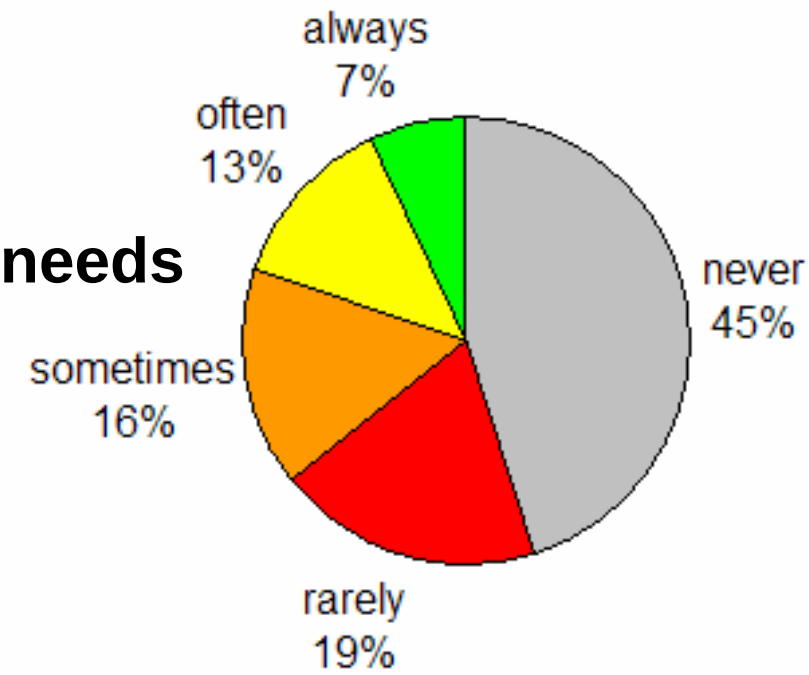
- > $1B project
- > 30 million lines of code,
- 8,000 person-years of development for one release
- No plan
- No specification
- Much higher quality than competitors
- Better fit to user needs

How to produce software that sucks …

- With the Waterfall approach, a great idea late in the development cycle is not a gift, it's a threat.
  - Pete Deemer, Chief Product Officer, Yahoo! India Research and Development and Gabrielle Benefield, Senior Director of Agile Development, Yahoo!, Inc.
- Over 50% of requirements change during software development. The Change Control Board makes sure the that development does not respond to change. As a result, about 50% of software is never used.
- We deliver on time and on budget 100% of the time. Our waterfall projects are 100% successful. The customer then says it is not what they wanted 100% of the time. Our failure rate is 100%. *BellSouth management*

80% of software sucks

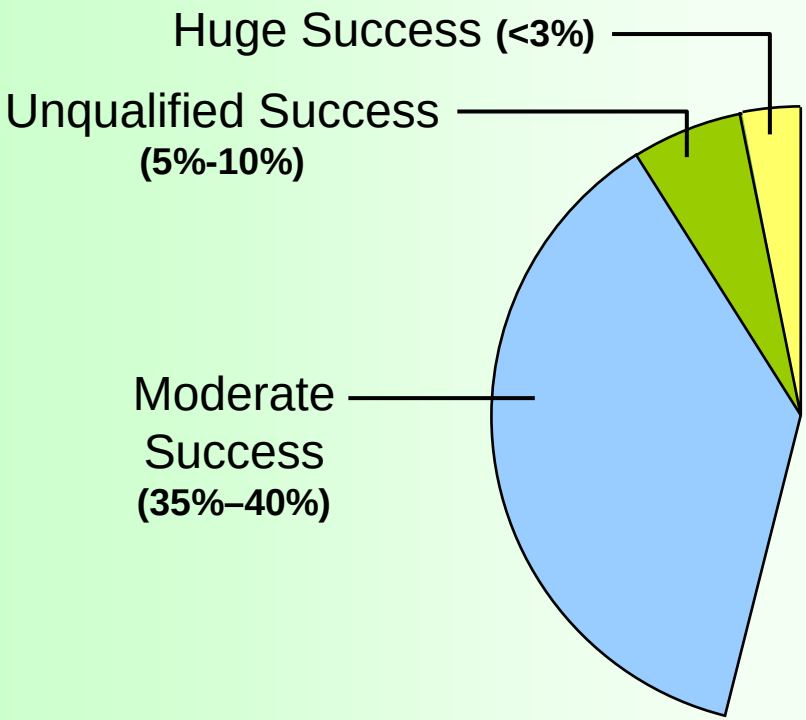
Fit to user needs



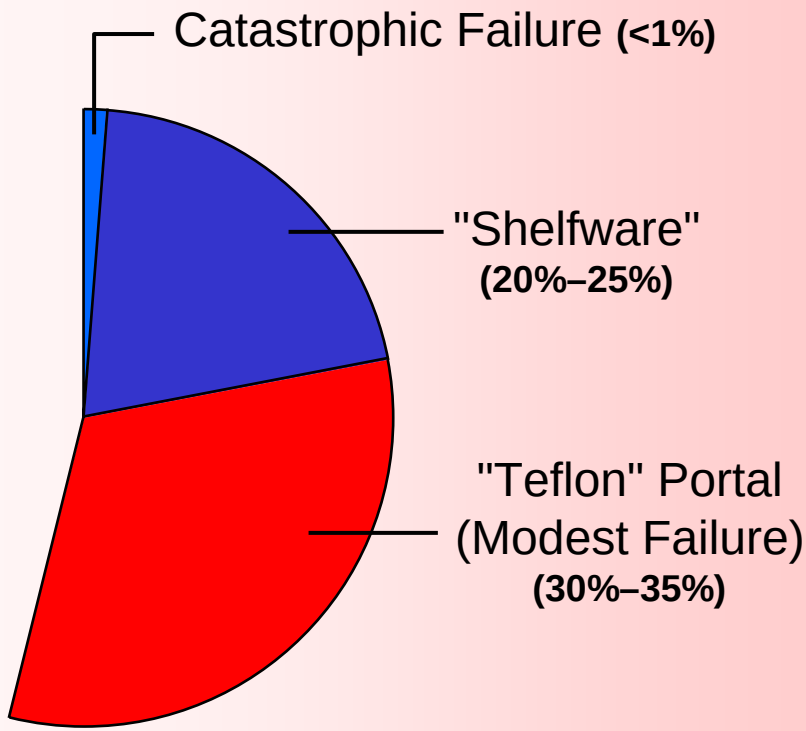
Jim Johnson. The Standish Group International Inc. 2002

Gartner Portal Project informal case study (David Norton, 2007)

Modes of Success



Modes of Failure



Wicked Problems: Righteous Solutions

- Wicked problems have no definitive formulation. Each attempt at creating a solution changes your understanding of the problem.
- Wicked problems have no stopping rule. The problem-solving process ends when resources are depleted, stakeholders lose interest or political realities change.
- Solutions to wicked problems are not true-or-false, but good-or-bad … getting all stakeholders to agree that a resolution is "good enough" can be a challenge.
- There is no immediate or ultimate test of a solution to a wicked problem.
- Every implemented solution to a wicked problem has consequences.
- Wicked problems don't have a well-described set of potential solutions. Various stakeholders have differing views of acceptable solutions.
- Each wicked problem is essentially unique. Part of the art of dealing with wicked problems is the art of not knowing too early what type of solution to apply.
- Each wicked problem can be considered a symptom of another problem. A wicked problem is a set of interlocking issues and constraints that change over time, embedded in a dynamic social context.
- The causes of a wicked problem can be explained in numerous ways.
- The planner (designer) has no right to be wrong.

Rittel, H and Webber M. Dilemmas in a General Theory of Planning. Policy Sciences, Vol. 4. Elsevier, 1973.
Degrace and Hulet's book, Wicked Problems, Righteous Solutions, Prentice Hall, 1990

The Enterprise as a complex adaptive system

- Business entities are examples of complex adaptive systems.
- Modification time of business processes is rapidly accelerating.
- Automating business processes renders parts of the business in software.
- When software modification time exceeds business modification time, the company cannot meet market demands.

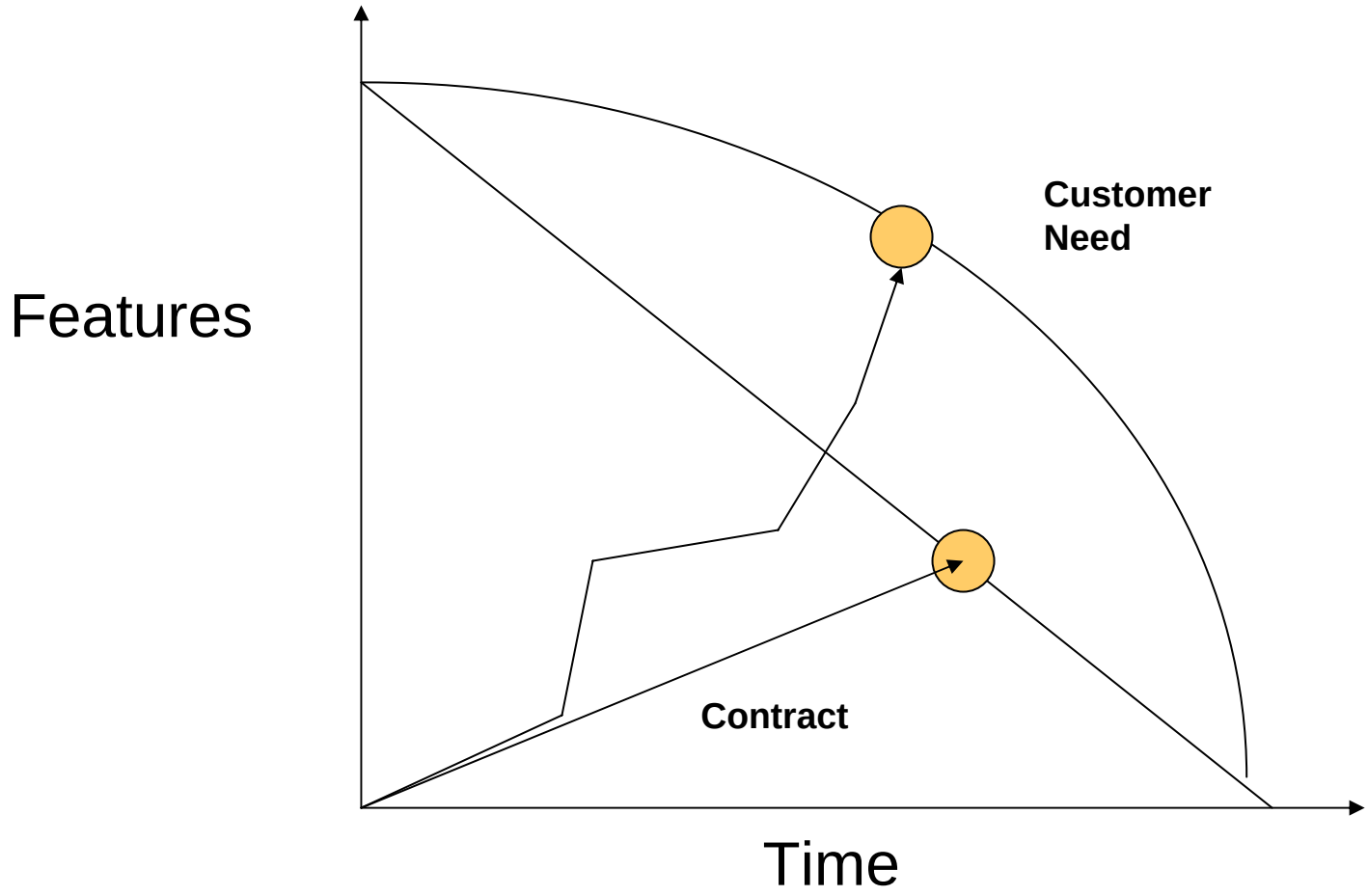
Sutherland, Jeff and van den Heuvel, Willem-Jan (2002) Developing and integrating enterprise components and services: Enterprise application integration and complex adaptive systems. ***Communications of the ACM 45:10:59-64.***

Complex Adaptive Behavior

- Self organization
- No single point of control
- Interdisciplinary teams
- Emergent behavior
- Outcomes emerge with high dependence on relationship and context
- Team performance far greater than sum of individuals

J. Sutherland, A. Viktorov, and J. Blount, ***Adaptive Engineering of Large Software Projects with Distributed/Outsourced Teams,*** in International Conference on Complex Systems, Boston, MA, USA, 2006.

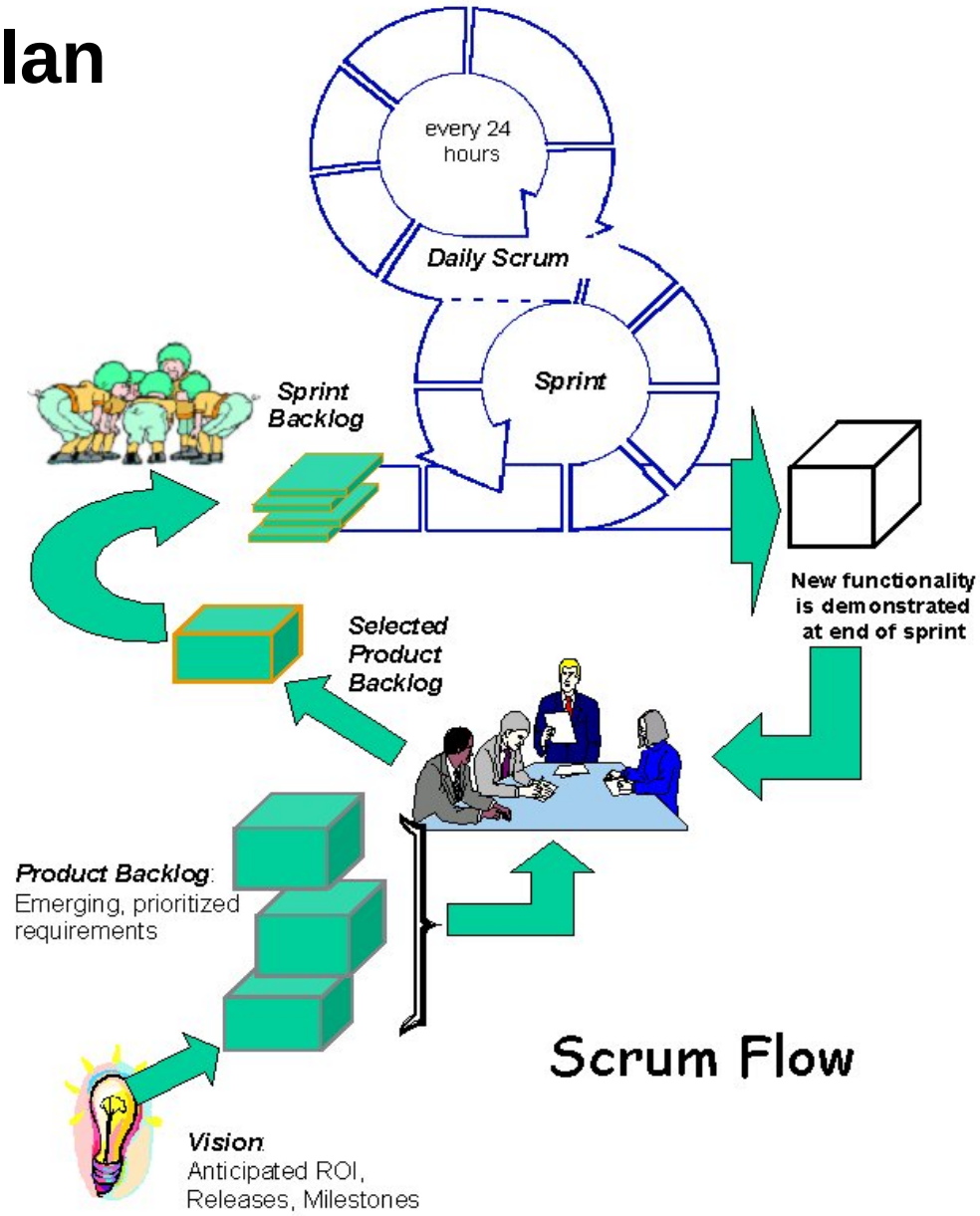
Successful delivery on plan is the root cause of failure.



Scrum – *value driven* not plan driven

- Empower lean teams to deliver more software earlier with higher quality.
- Demonstrate working features to the customer early and often so the customer can inspect progress and prioritize change.
- Deliver exactly what the client wants by directly involving the customer in the development process.
- Provide maximum business value to the customer by responding to changing priorities in real time.

Replacing the plan with adaptive planning …



Local action, inspect and adapt, forces self-organization

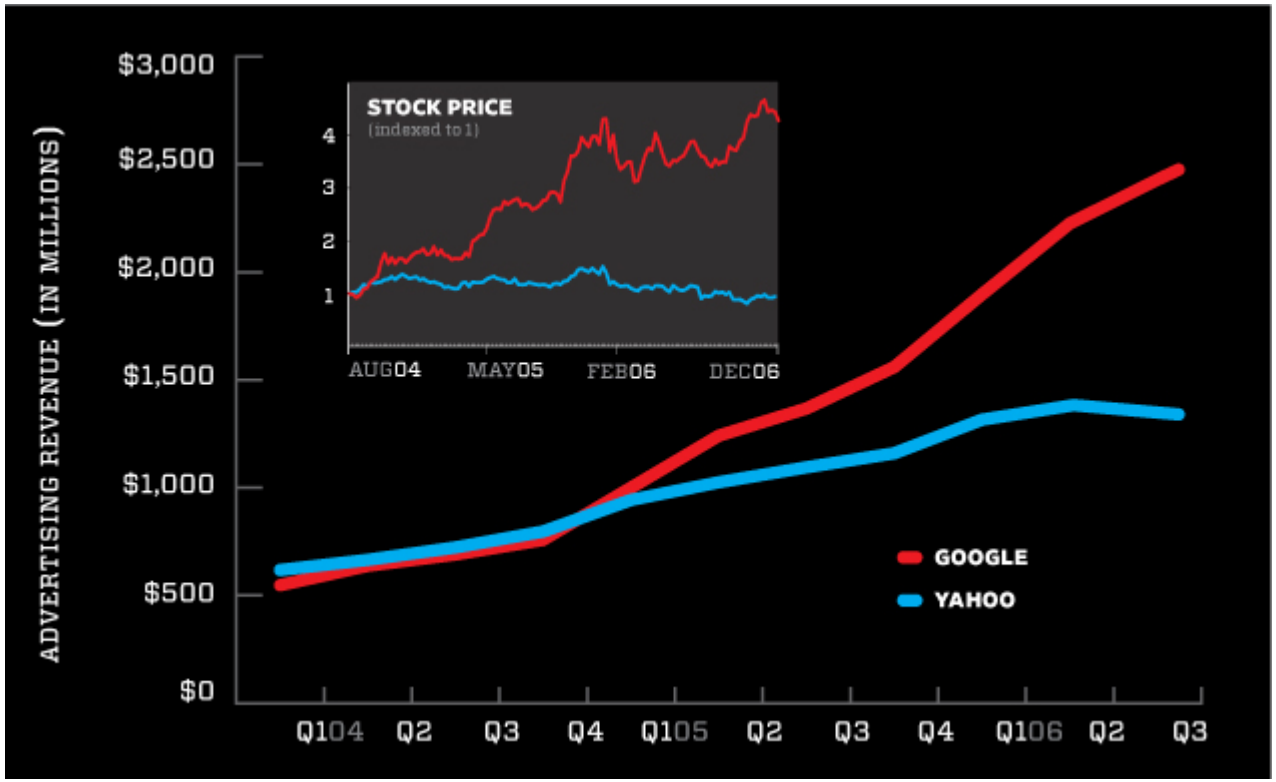
- Individual self-organizes work
- Team self-organizes around goals
- Architecture self-organizes around working code
- Product emerges through iterative adaptation
- Collaborative approach as opposed to authoritative approach
- Flat organizational structure

Google and Scrum

- “With the help of an experienced agile leader (Scrum Master, XP coach…) it was possible to carefully introduce agile practices into Google - an environment that does not have an affinity to processes in general.
- “Along with these practices came a visibility into the development status that gave the approach great management support.
- “All this could be done without destroying the great bottom-up culture that Google prides itself of.”

Mark Striebeck
Google AdWords Project Leader and ScrumMaster

Powered by Scrum



AdWords is the “KaChing” machine at Google. It makes most of the money (2.5B Q3/06) that gives Google its market cap of $147B just behind Chevron and ahead of Intel. The most profitable software application in the history of computing is powered by Scrum.

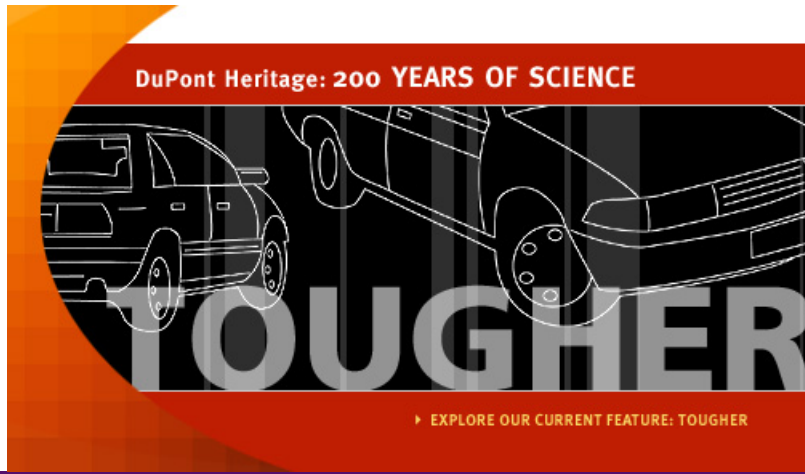
Theory: Process

Defined vs. Empirical Process

- It is typical to adopt the defined (theoretical) modeling approach when the underlying mechanisms by which a process operates are reasonably well understood. When the process is too complicated for the defined approach, the empirical approach is the appropriate choice.

Process Dynamics, Modeling, and Control. Ogunnaike and Ray, Oxford University Press, 1992

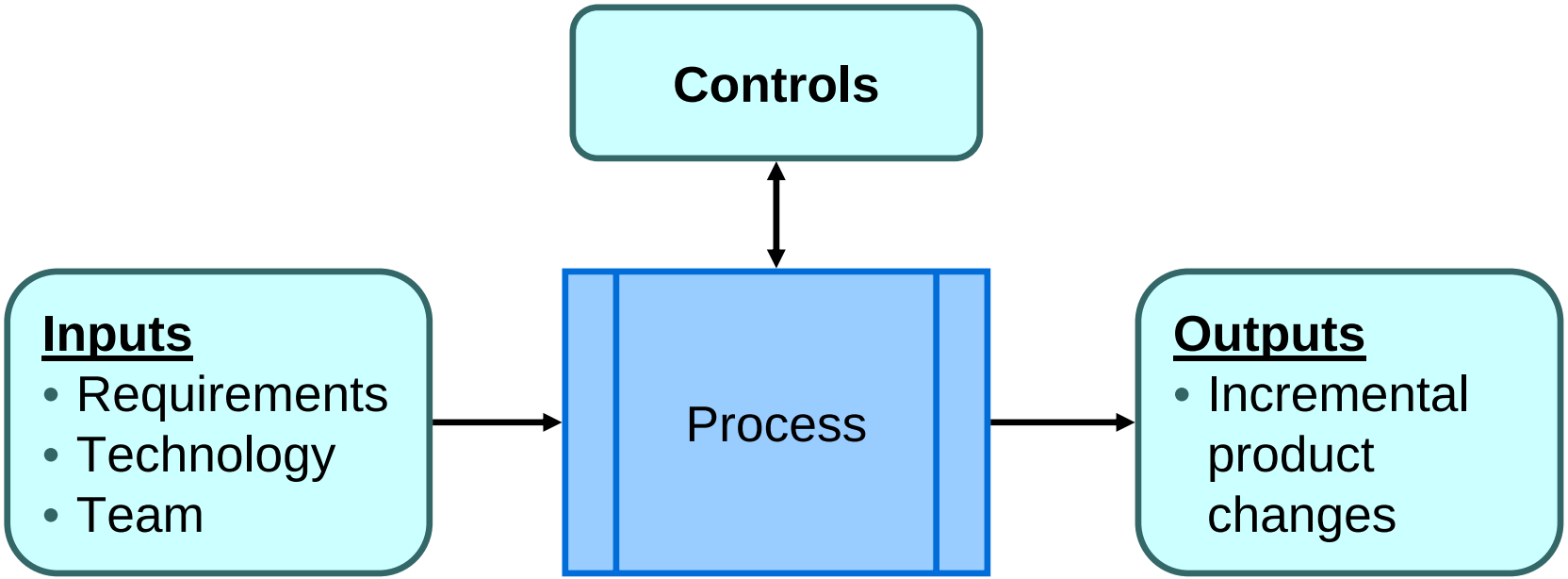
DUPONT The miracles of science™



Software Development is an Empirical Process

- Ziv's Uncertainty Principle in Software Engineering - uncertainty is inherent and inevitable in software development processes and products *[Ziv, 1996].*
- Humphrey's Requirements Uncertainty Principle - for a new software system, the requirements will not be completely known until after the users have used it.
- Wegner's Lemma - it is not possible to completely specify an interactive system *[Wegner, 1995].*

Uncertainty demands Empirical process control



Adapted from *Agile Software Development with Scrum* by Ken Schwaber and Mike Beedle. Courtesy of Mike Cohn, Mountain Goat Software

Bell Labs Report on most productive project ever: Borland Quattro for Windows

1,000,000 lines of C++ code	BWP	Industry standard
Time in months	31	>50
Staff	8	>100
Function points per staff month	77	2

Jones, Capers. Applied Software Measurement, Second Edition. McGraw Hill, 1997.

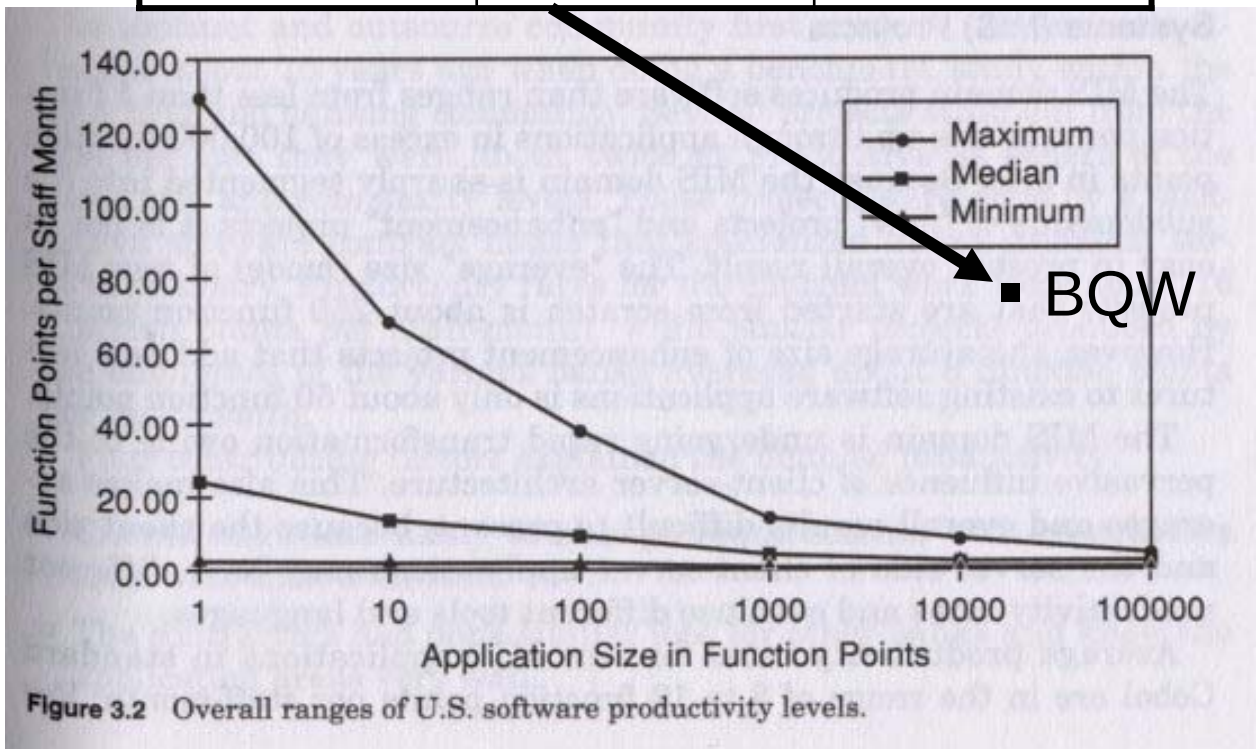
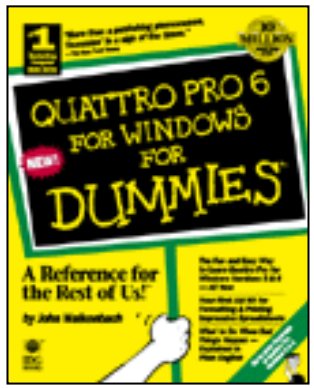


Figure 3.2 Overall ranges of U.S. software productivity levels.



James Coplien. Borland Software Craftsmanship: A New Look at Process, Quality, and Productivity. Proceedings of the 5th Annual Borland International Conference, Orlando, 1994.

- One of most remarkable organizations, processes, and development cultures seen in AT&T Bell Laboratories Pasteur process research project
- Project management, product management, QA integral to team, all making technical contributions
- Higher communication saturation than 89% of projects
- “Anti-schismogenetic” – no cliques
- Highly iterative development
- Strong architectural interaction with implementation
- More time spent in project team meetings than anything else – several hours a day
- Gerry Weinberg notes that CMM Level 1 and 2 teams need strong managerial direction. Level 3 paradigm shift is self-directing team. Borland team was clearly in this category, although not by commonly accepted criteria.

Team comments on Quattro project

- “We are satisfied by doing real work.”
- “Software is like a plant that grows. You can’t predict its exact shape, or how big it will grow.”
- “There are no rules for this kind of thing—it’s never been done before.”

“Evolutionary development is best technically, and it saves time and money.”

Report of the Defense Science Board Task Force on Military Software. Oct 1987.

History of Iterative and Incremental Development (IID)

- 1956 – Benington's stagewise model – USAF SAGE System
- 1957 – IBM Service Bureau Corp, Project Mercury, IBM Federal Systems Devision – Gerry Weinberg
- 1960 – Weinberg teaching IID at IBM Systems Research Institute
- 1969 - Earliest published reference to IID:
  - Robert Glass. Elementary Level Discussion of Compiler/Interpreter Writing. ACM Computing Surveys, Mar 1969

Larman, Craig and Basili, Vic. Iterative and Incremental Development: A Brief History. IEEE Computer, June 2003 (Vol. 36, No. 6) pp. 47-56

History of Iterative and Incremental Development (IID)

- **1971 – IBM Federal Systems Division**
  - Mills, Harlan. Top-down programming in Large Systems. In Debugging Techniques in Large Systems. Prentice Hall, 1971
- **1972 – TRW uses IID on $100M Army Site Defense software**
- **1975 – First original paper devoted to IID**
  - Gasili, Vic and Turner, Albert. Iterative Enhancement: A Practical Technique for Software Development. IEEE Transactions on Software Engineering. Dec 1975.
- **1977-1980 – IBM FSD builds NASA Space Shuttle software in 17 iterations over 31 months, averaging 8 weeks per iteration**
  - Madden and Rone. Design, Development, Integration: Space Shuttle Flight Software. Communications of the ACM, Sept 1984.

Larman, Craig and Basili, Vic. Iterative and Incremental Development: A Brief History. IEEE Computer, June 2003 (Vol. 36, No. 6) pp. 47-56

History of Iterative and Incremental Development (IID)

- **1985 – Barry Boehm's Spiral Model**
  - Boehm, Barry. A Spiral Model of Software Development and Enhancement. Proceedings of an International Workshop on Software Process and Software Environments. March, 1985
- **1986 – Brooks, Fred. No Silver Bullet. IEEE Computer, April 1987**
  - Nothing … has so radically changed my own practice, or its effectiveness [as incremental development].
- **1993 – First SCRUM at Easel Corporation**
- **1994 – DOD must manage programs using iterative development**
  - Report of the Defense Science Board Task Force on Acquiring Defense Software Commercially. June 1994.
- **1995 – Microsoft IID published**
  - McCarthy, Jim. Dynamics of Software Development. Microsoft Press, 1995.
- **1996 – Kruchten. A Rational Development Process. Crosstalk. July.**
  - Origins of RUP

Larman, Craig and Basili, Vic. Iterative and Incremental Development: A Brief History. IEEE Computer, June 2003 (Vol. 36, No. 6) pp. 47-56

History of Iterative and Incremental Development (IID)

- **1996 – Kent Beck Chrysler Project**
  - Origin of XP
- **1996 – Larman meets with principal author of DD-STD-2167**
  - David Maibor expressed regret for the creation of the waterfall-based standard. He had not learned of incremental development at the time and based his advice on textbooks and consultants advocating the waterfall method. With the hindsight of iterative experience, he would recommend IID.
- **1997 – Coad and DeLuca rescue Singapore project**
  - Origin of Feature-Driven Development
- **1998 – Standish Group CHAOS Project**
  - Top reason for massive project failures was waterfall methods. “Research also indicates that smaller time frames, with delivery of software components early and often, will increase success rate.
- **1999 – Publication of extensive DOD failures**
  - Out of a total cost of $37B for the sample set, 75% of projects failed or were never used, and only 2% were used without extensive modification. Jarzombek. The 5th Annual JAWS S3 Proceedings, 1999.

Larman, Craig and Basili, Vic. Iterative and Incremental Development: A Brief History. IEEE Computer, June 2003 (Vol. 36, No. 6) pp. 47-56

History of Iterative and Incremental Development (IID)

- **2001 – 17 process expert "anarchists" meet at Snow Bird**
  - Agile Manifesto initiated 100s of books and papers on agile development
- **2001 – MacCormack's study of key success factors**
  - MacCormack, Alan. Product-Develoment Practices that Work. MIT Sloan Management Review 42:2, 2001.

Nokia Checklist

- **You know you are not doing any Agile process if you are not doing iterative development**
- **You know you do not do iterative development when:**
  - Iterations are longer than 6 weeks
  - Iterations are not timeboxed
  - Team tries to finish all specification before programming
  - Iteration doesn't result in workable code
  - Iterations doesn't include testing
- **You know you do not use scrum when:**
  - The team doesn't know who the product owner is
  - Your product backlog doesn't contain estimates
  - You cannot generate a release burn-down chart and don't know your velocity
  - There is a project manager in the project disrupting the work of the team

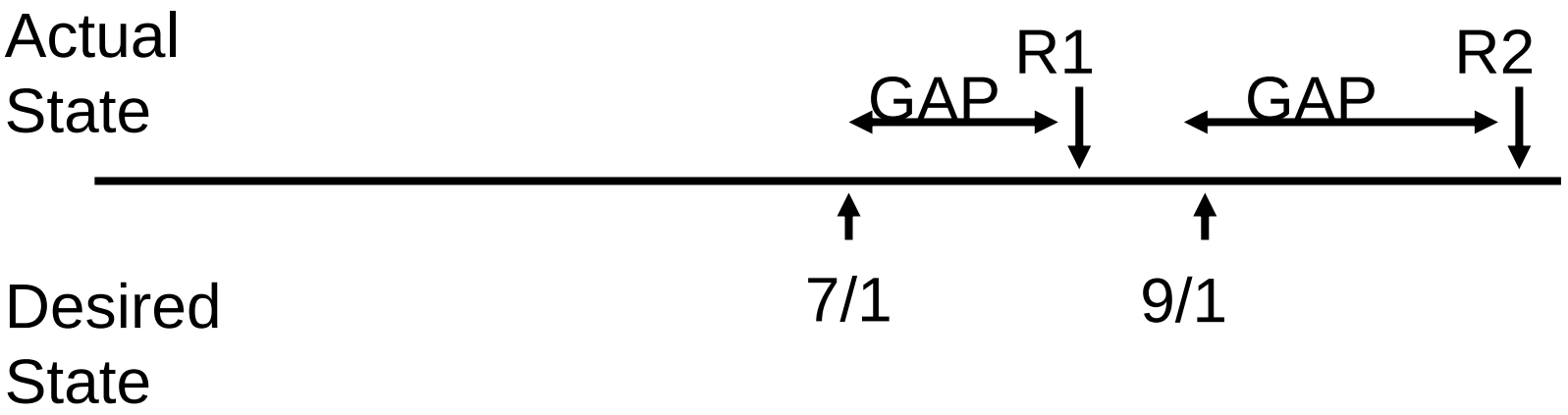
Product Owner drives the plan

- Vision
- Roadmap
- Slide deck to evangelize project
- Product Backlog
- Estimates
- Understand velocity of teams
- Incorporate technical backlog
- Release dates
- Manage the GAP

Delivering Releases on Time

- Product backlog prioritized by business value with estimates
- Product owner knows velocity of teams
- Manage the gap between desired state and current state
  - Can we reprioritized the backlog to deliver early release that captures value?
  - Can team increase velocity?
  - If not, what can be pulled off the bottom of the product backlog?
  - Can management sponsors provide more resources?

Product Owner: Manage the Gap

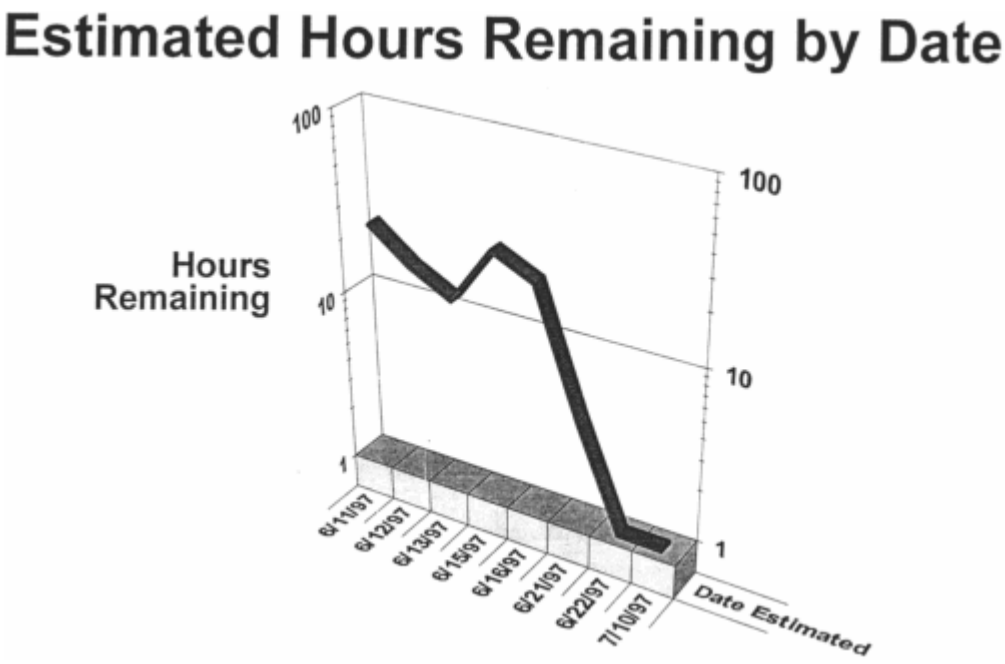


How do we meet dates with Scrum?

- Microsoft TV team from Redmond drinking beer at happy hour with me at the Avante Hotel in Mountain View …
  - "Jeff, we have a question for you. How do you meet fixed dates with Scrum?"
- *"Do you have a burndown chart and know your velocity?"*
- "No."
- *"You will never meet a date with Scrum!"*

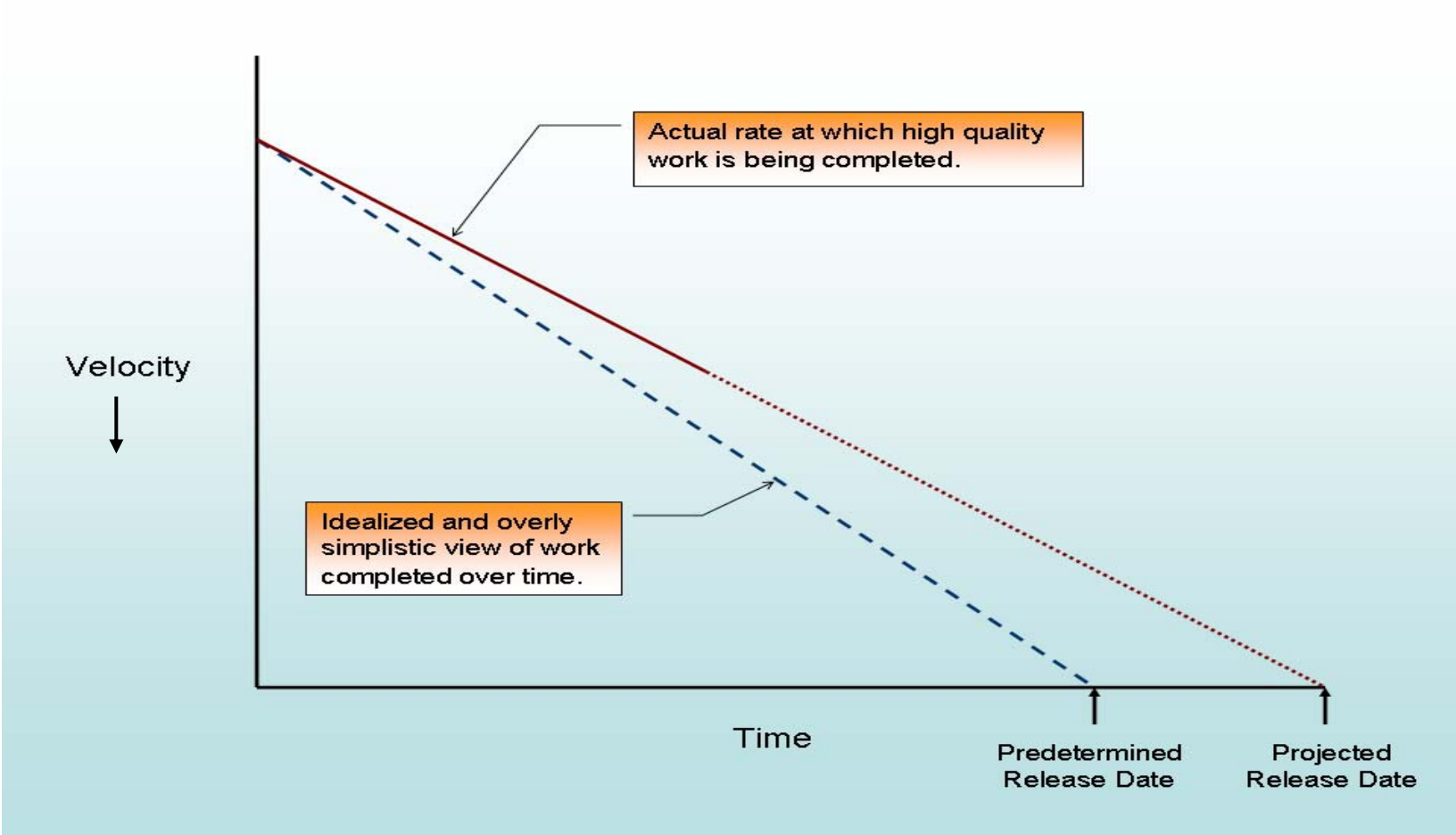
They did not meet the Nokia test for doing Scrum.

ScrumMaster – Manage the Burndown



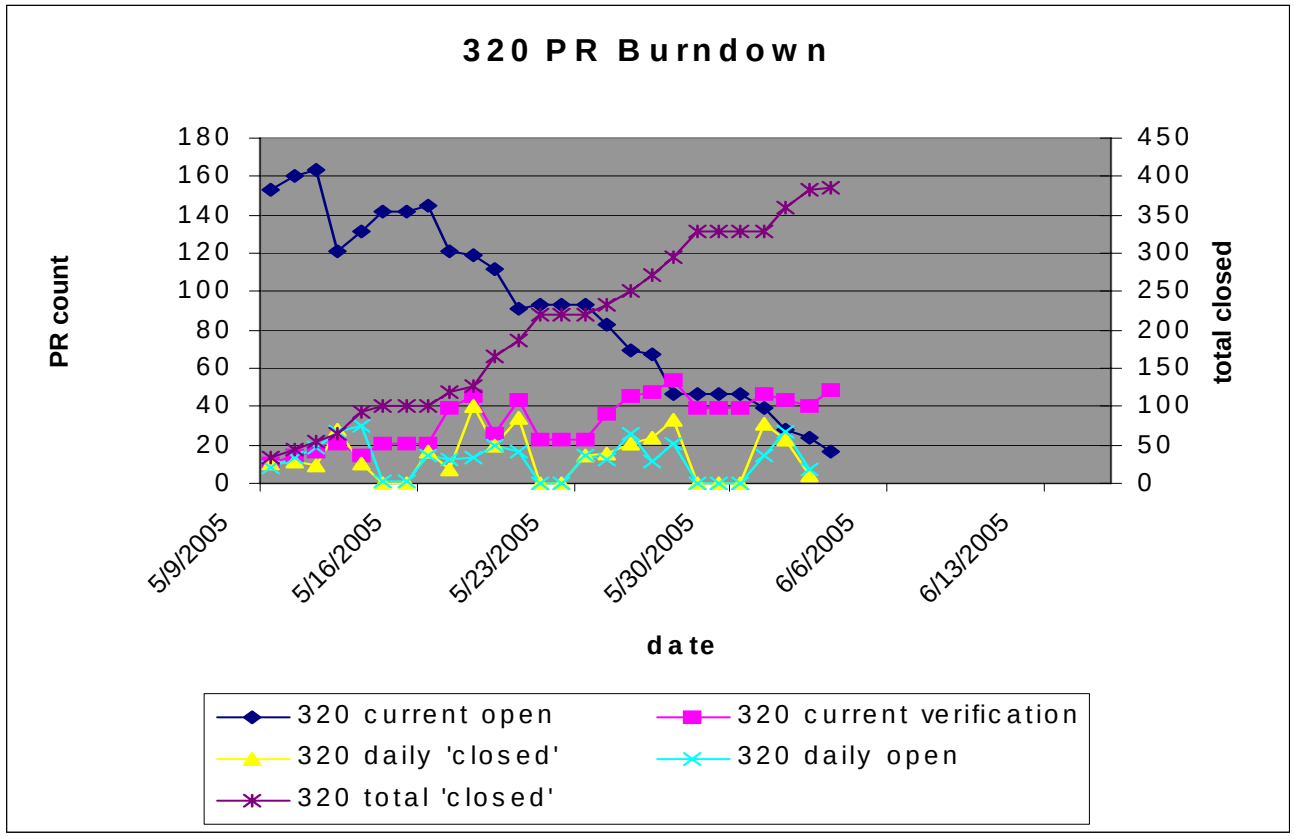
http://www.controlchaos.com/about/burndown.php

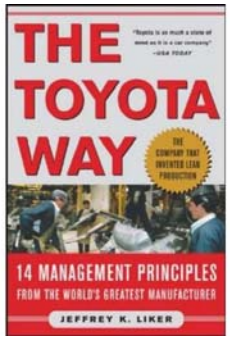
Sprint in Trouble



Graphic from Kan Mare (2006) Technical Debt and Design Debt.

ScrumMaster: managing the burndown, knowing the velocity





Toyota Way: Learn by Doing

Fujio Cho, President, 2002

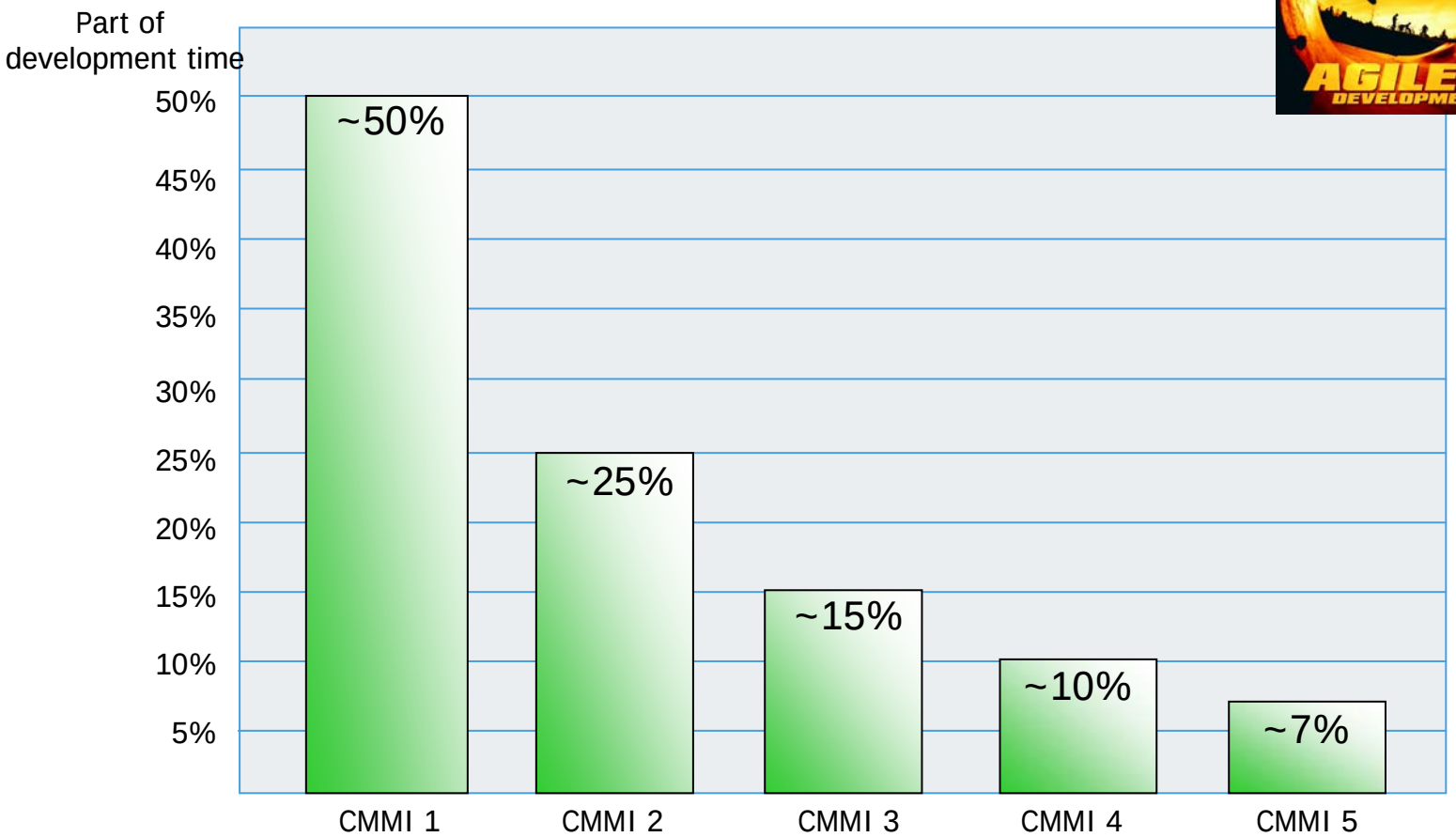
- We place the highest value on actual implementation and taking action. *Agile Principle #1*
- There are many things one doesn't understand the therefore, we ask them why don't you just go ahead and take action? *Agile Principle #3, #11*
- You realize how little you know and you face your own failures and redo it again and at the second trial you realize another mistake … so you can redo it once again. *Agile Principle #11, #12*
- So by constant improvement … one can rise to the higher level of practice and knowledge. *Agile Principle #3*

Toyota consulting applied to most agile U.S. company (industrial sensor company – 6 month results)

- 93% reduction in lead time to product product
- 83% reduction in work-in-progress inventory
- 91% reduction in finished goods inventory
- 50% reduction in overtime
- 83% improvement in productivity

- The mindset is the key to transformation. Consulting help from outside experts is critical in early stage.

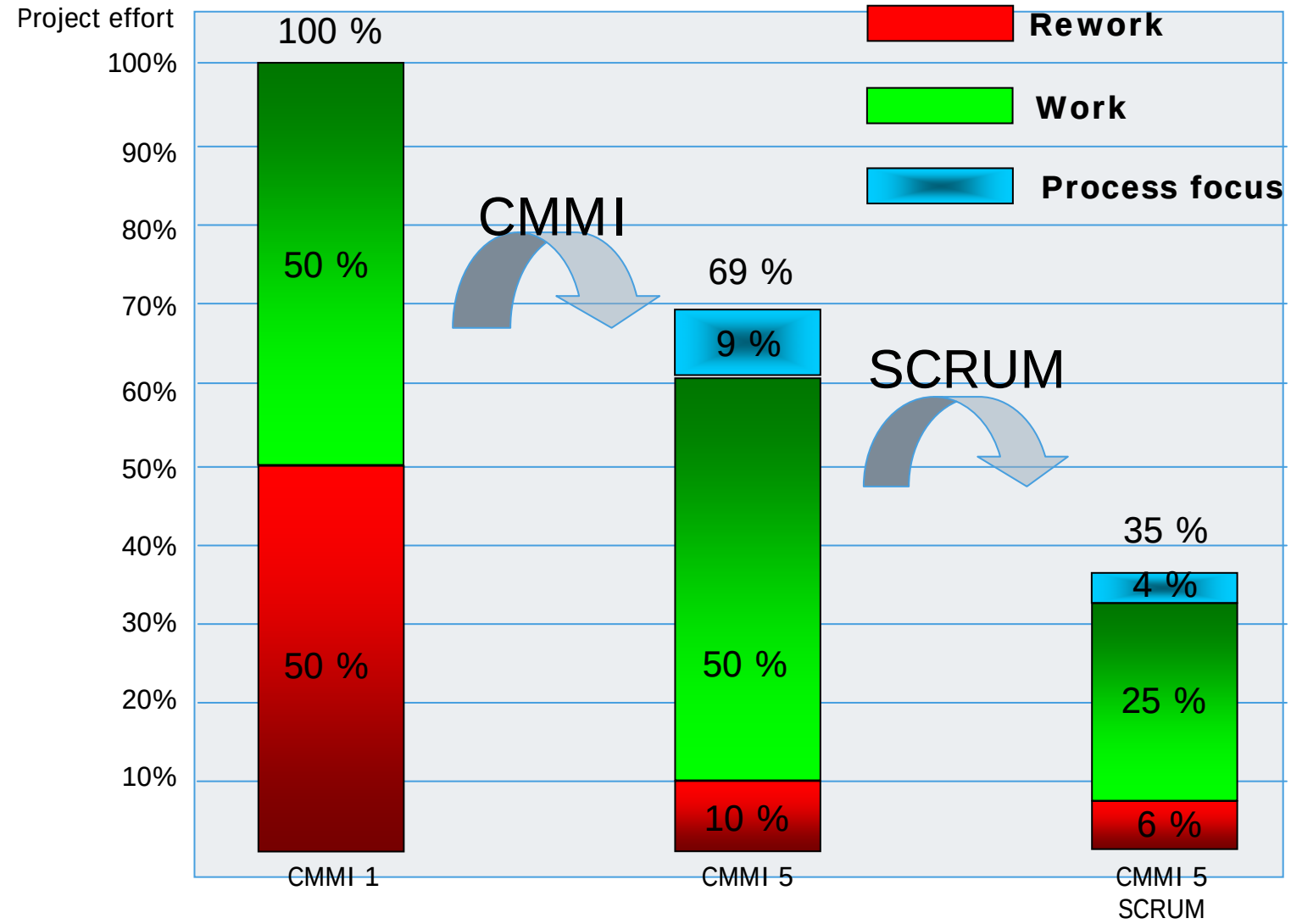
Published experiences with ”rework”



Source: Krasner & Houston, CrossTalk, Nov 1998
Diaz & King, CrossTalk, Mar 2002

Scrum applied to CMMI Level 5 company

– *6 month results*

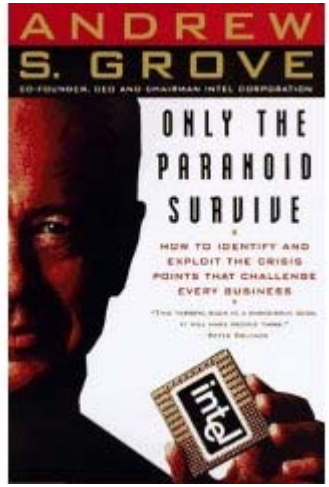


Systematic CMMI 5 Analysis
First six months of Scrum

- 80% reduction in planning cost
- 40% reduction in defects
- 50% reduction in rework
- 100% increase in overall productivity
- Systematic decided to change CMMI Level 5 process to make Scrum the default mode of project management
- When waterfall project management is demanded, they are now contracted for twice the price of Scrum projects
  - Required by some defense and healthcare agencies
  - Results are lower business value
  - Lower customer satisfaction
  - Lower quality
  - Twice the cost

Sutherland, J., C. Jacobson, et al. (2007). Scrum and CMMI Level 5: A Magic Potion for Code Warriors! Agile 2007, Washington, D.C., IEEE.

Following the plan will ...



- Increase planning costs by 500%
- Double the price of the project (at least)
- Increase bugs by 166% (at least)
- Decrease user satisfaction
- Reduce employee morale and increase turnover
- May lead to 100% failure rates in terms of meeting customer needs.

You can stop following the plan. To function like Toyota, stopping is mandatory. Do it before your competition or you may be roadkill on the information highway!

Questions?

Bibliography

- Cohn, M. (2004). User Stories Applied : For Agile Software Development, Addison-Wesley.
- Cohn, M. (2005). Agile Estimation and Planning, Addison-Wesley.
- Larman, Craig and Basili, Vic. Iterative and Incremental Development: A Brief History. IEEE Computer, June 2003 (Vol. 36, No. 6) pp. 47-56
- Poppendieck, M. and T. Poppendieck (2006). Lean Software Development: An Implementation Guide, Addison-Wesley.
- Schwaber, K. (2004). Agile project management with Scrum. Redmond, Wash., Microsoft Press.
- Sutherland, J., C. Jacobson, et al. (2007). Scrum and CMMI Level 5: A Magic Potion for Code Warriors! Agile 2007, Washington, D.C., IEEE.
- Sutherland, J. and K. Schwaber (2007). The Scrum Papers: Nuts, Bolts, and Origins of an Agile Method. Boston, Scrum, Inc.
- Takeuchi, H. and I. Nonaka (1986). "The New New Product Development Game." Harvard Business Review(January-February).
- Takeuchi, H. and I. Nonaka (2004). Hitotsubashi on Knowledge Management. Singapore, John Wiley & Sons (Asia).