







2



**Jeff Sutherland**  
[jeffsutherland.com/scrum](http://jeffsutherland.com/scrum)

- **Agile Systems Architect since 1983**
  - 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
  - Scrum consultant to leading companies in Europe, Americas, and Russia.
  - Senior Advisor, OpenView Venture Partners
- **Signatory of Agile Manifesto and founder of Agile Alliance**
- **Scrum Certification World Tour**

<ul style="list-style-type: none"> <li>Jun 13 Oresund Agile Copenhagen</li> <li>Jun 14-15 Saxo Bank Copenhagen</li> <li>Jun 18-19 Program Utvikling Oslo</li> <li>Jun 20-21 IT University Stockholm</li> <li>Jun 23-24 Exigen St. Petersburg</li> <li>Jun 26-27 Xebia Amsterdam</li> <li>Jul 11-12 Kitty Hawk Charlotte</li> <li>Jul 25-26 Healthwise Boise</li> <li>Jul 30 SolutionsIQ Seattle</li> <li>Aug 1-2 Sunnyvale, CA</li> <li>Aug 6-7 MySpace Beverly Hills</li> <li>Aug 13-17 Agile 2007 Washington, D.C.</li> <li>Aug 23-24 Systematic Aarhus</li> <li>Aug 26-27 Crisp Stockholm</li> <li>Aug 28-29 Crisp Stockholm</li> <li>Aug 30-31 Saxo Bank Copenhagen</li> <li>Sept 5-6 New York</li> </ul>	<ul style="list-style-type: none"> <li>Sep 10-11 CSM Oslo</li> <li>Sep 13-14 CSM Google Trondheim</li> <li>Sep 17 Johns Hopkins APL</li> <li>Sep 18 Motley Fool</li> <li>Sep 19-20 CSM Insight</li> <li>Sep 27-28 JAOC CSM Aarhus</li> <li>Oct 1-2 CSM Rome</li> <li>Oct 4-5 CSM Xebia</li> <li>Oct 9 Agile Stockholm</li> <li>Oct 10-11 CSM Stockholm</li> <li>Oct 15-16 CSM Copenhagen</li> <li>Oct 18-19 CSM Copenhagen</li> <li>Oct 22-26 Camp Scrum</li> <li>Oct 29-30 CSM Nortel Ottawa</li> <li>Nov 1-2 OpenView Venture Partners</li> <li>Nov 3-4 Deep Lean MIT</li> <li>Nov 8-9 CSM Boston</li> </ul>
---	--

© Jeff Sutherland 1993-2007

<h3>Plan driven development</h3> <ul style="list-style-type: none"> <li>■ High failure rate</li> <li>■ Produces software that sucks             <ul style="list-style-type: none"> <li>– Fails to fit customer needs</li> <li>– High defect rate</li> </ul> </li> <li>■ Over 50% waste</li> <li>■ Delays time to market</li> <li>■ Poor working environment</li> </ul>	<h3>Value driven development</h3> <ul style="list-style-type: none"> <li>■ High success rates</li> <li>■ Produces software that meets customer needs</li> <li>■ Minimal waste</li> <li>■ Accelerates early revenue</li> <li>■ Energized working environment</li> </ul>
--	--

© Jeff Sutherland 1993-2007

### 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%

Challenged  
53%

Succeeded  
29%

Failed  
18%

Mainly caused by lack of:

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

© Jeff Sutherland 1993-2007

## 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

© Jeff Sutherland 1993-2007

## 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.
  - Often Change Control Board makes sure the that development does not respond to change.
  - As a result, about 50% of software is never used because customers don't want it.
- We deliver on features, 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*

© Jeff Sutherland 1993-2007

## Only 20% of software meets user needs

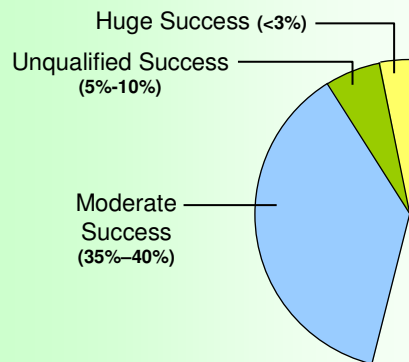


Jim Johnson. The Standish Group International Inc. 2002

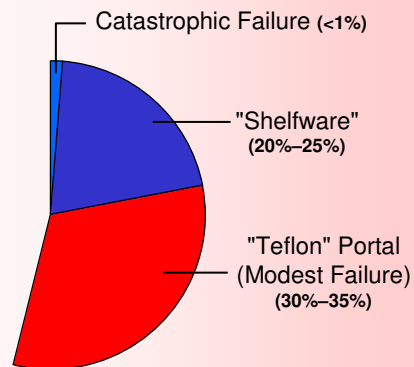
© Jeff Sutherland 1993-2007

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

### Modes of Success



### Modes of Failure



© Jeff Sutherland 1993-2007



## 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.  
 Degraace and Hulet's book, Wicked Problems, Righteous Solutions, Prentice Hall, 1990 © Jeff Sutherland 1993-2007

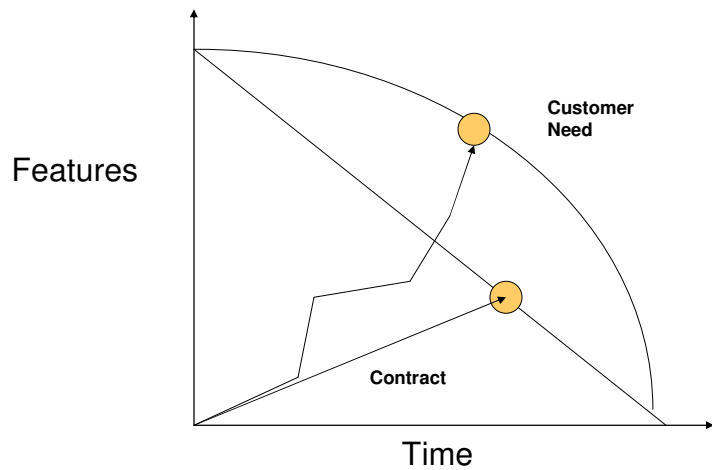
## The Enterprise as a complex adaptive system

- 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.

© Jeff Sutherland 1993-2007

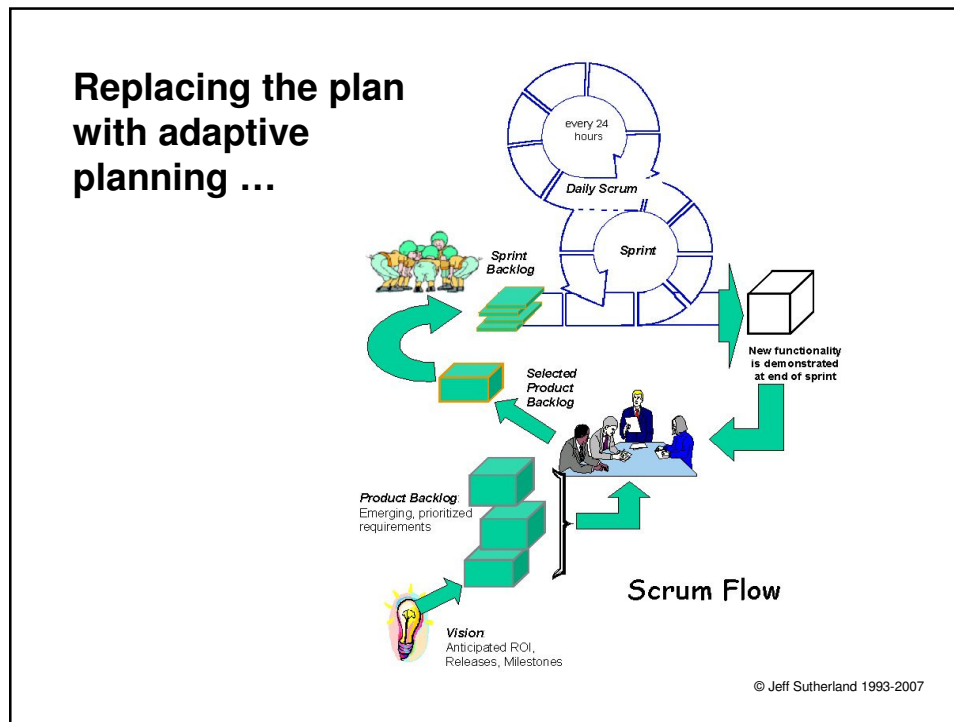
## Successful delivery on plan is the root cause of failure.



## 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.

© Jeff Sutherland 1993-2007



## 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

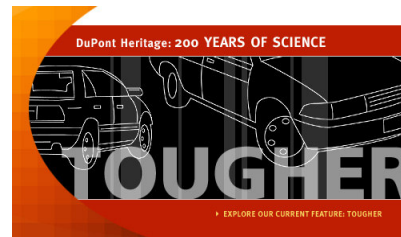
© Jeff Sutherland 1993-2007

### **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

 *The miracles of science*

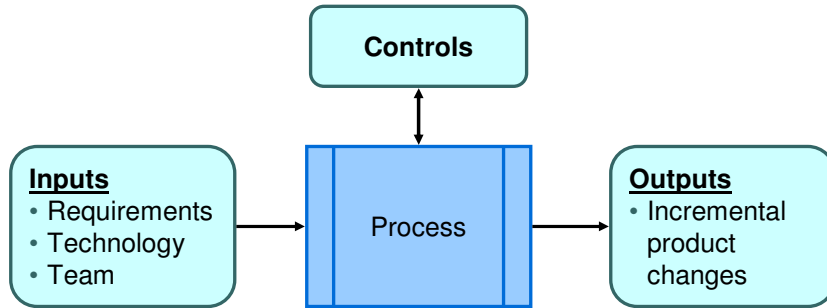


### **Software Development is an Empirical Process**

- 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].

© Jeff Sutherland 1993-2007

## Uncertainty demands Empirical process control

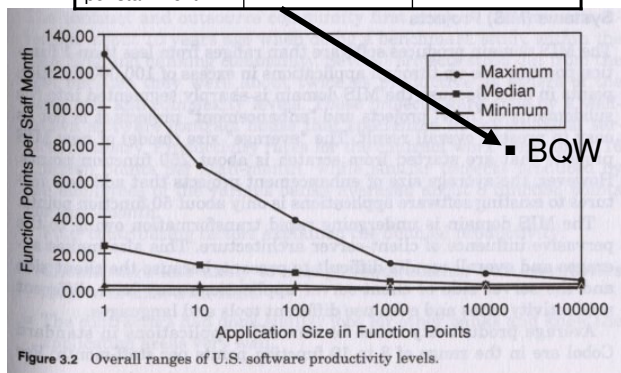


Adapted from *Agile Software Development with Scrum* by Ken Schwaber and Mike Beedle. Courtesy of Mike Cohn, Mountain Goat Software  
© Jeff Sutherland 1993-2007

## Bell Labs identifies 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.



© Jeff Sutherland 1993-2007



**James Coplien. Borland Software Craftsmanship: A New Look at Process, Quality, and Productivity. Proceedings of the 5<sup>th</sup> 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.

© Jeff Sutherland 1993-2007

## 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.*

© Jeff Sutherland 1993-2007

## 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 Division – 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

© Jeff Sutherland 1993-2007

## 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

© Jeff Sutherland 1993-2007

## 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**
- **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

© Jeff Sutherland 1993-2007

## History of Iterative and Incremental Development (IID)

- **1996 – Kent Beck Chrysler Project**
  - Origin of XP
- **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.

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

© Jeff Sutherland 1993-2007

## **Waterfall a colossal mistake!**

- **1994 – DOD must manage programs using iterative development**
  - Report of the Defense Science Board Task Force on Acquiring Defense Software Commercially. June 1994.
- **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.
- **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 5<sup>th</sup> Annual JAWS S3 Proceedings, 1999.

© Jeff Sutherland 1993-2007

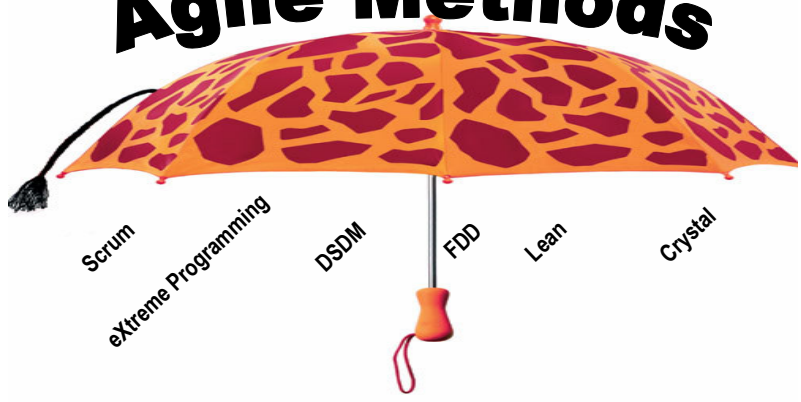
## **History of Iterative and Incremental Development (IID)**

- **2001 – MacCormack’s study of key success factors**
  - MacCormack, Alan. Product-Development Practices that Work. MIT Sloan Management Review 42:2, 2001.
- **2001 – 17 process expert “anarchists” meet at Snow Bird**
  - Agile Manifesto unleashes dozens of books and hundreds of papers on agile development

© Jeff Sutherland 1993-2007

27

# Agile Methods



Scrum  
extreme Programming  
DSDM  
FDD  
Lean  
Crystal

© Jeff Sutherland, 1993-2007  
Copyright 1993-2007, Jeff Sutherland, v7.3

28

**NOKIA**  
Connecting People

## Are you doing Scrum? (50% of Scrum teams meet the first half of the Nokia Test)

- First, you must be doing iterative development
  - Iterations must be timeboxed to four weeks or less
  - Software features must be tested and working at the end of an iteration
  - Iteration must start with an Agile specification

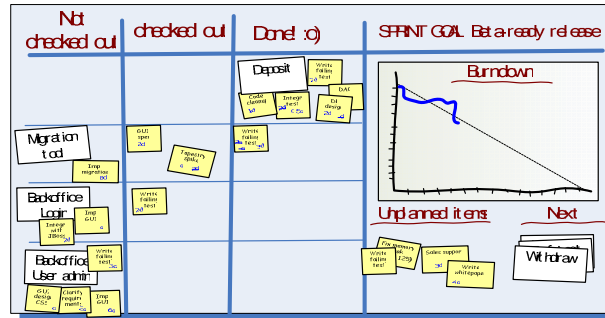
**Iterative Incremental development**

- Robert Glass. Elementary Level Discussion of Compiler/Interpreter Writing. ACM Computing Surveys, Mar 1969
- G. Booch, *Object Solutions: Managing the Object-Oriented Project*. Addison-Wesley, 1995.
- Larman, Craig and Basili, Vic. Iterative and Incremental Development: A Brief History. IEEE Computer, [June 2003 \(Vol. 36, No. 6\)](#) pp. 47-56

© Jeff Sutherland, 1993-2007  
Copyright 1993-2007, Jeff Sutherland, v7.3

### Nokia Scrum Test (10% of Scrum teams)

- You know who the product owner is
- There is a product backlog prioritized by business value
- The product backlog has estimates created by the team
- The team generates burndown charts and knows their velocity
- There are no project managers (or anyone else) disrupting the work of the team

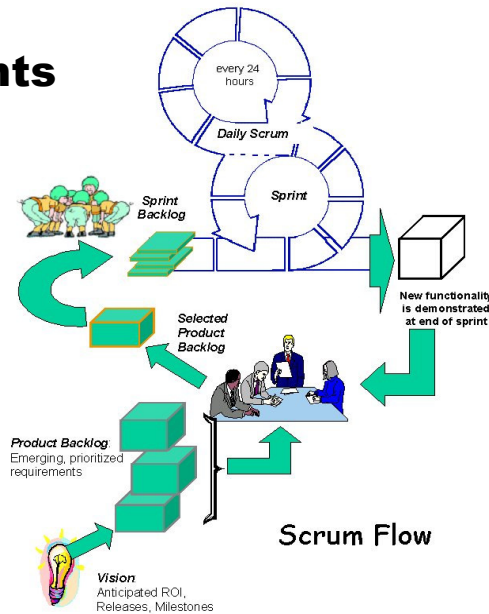


Kniberg, Henrik. Scrum and XP from the Trenches: How We Do Scrum. Version 2.1, Crisp, 5 Apr 2007.

Copyright 1993-2007, Jeff Sutherland, 1993-2007

### Scrum Failure Points

- Product Owner
- Sprint Planning
- ScrumMaster
- Team
- Daily Meeting
- Sprint Review
- Management



© Jeff Sutherland 1993-2007

## 1. Product Owner failure point

- The Product Owner does not have:
  - A vision
  - A business plan
  - A release roadmap
  - A product backlog that will deliver the right thing
- The Product Backlog:
  - Is not ordered properly
  - Does not contain all work (including technical issues)
  - Is not ready for the Sprint planning meeting
    - Is not sized properly
    - Is not estimated properly
    - Does not have enabling specifications
- The Product Owner is AWOL during the Sprint

© Jeff Sutherland 1993-2007

## Product Owner is worth 20% of revenue

- Proper ordering of Product Backlog will increase revenue by at least 20% (see Software by Numbers)
- Product Backlog not “ready” will cut productivity of team by at least 50%

© Jeff Sutherland 1993-2007

## 2. Sprint Planning failure point

- The Product Owner does not communicate clearly:
  - The vision, the business plan, the release roadmap
  - The Product Backlog is not ready
- The Team:
  - Takes too much off the Product Backlog
  - Does not break down features into Sprint tasks with good estimates – Sprint Backlog
- The ScrumMaster does not make sure the original estimates in the Product Backlog equals the more detailed estimates in the Sprint Backlog
- Trust, transparency, and truth is not present
- The plan does not meet the finger test

© Jeff Sutherland 1993-2007

## No Useless Code Strategy

- Not writing code the user will never use is the best coding possible
  - No code has no bugs
  - Maintenance cost is zero
- Rule at PatientKeeper for Product Backlog not ready
  - All developers go surfing
  - No useless code means no refactoring, no rework, no integration costs, and no future problems in the field!

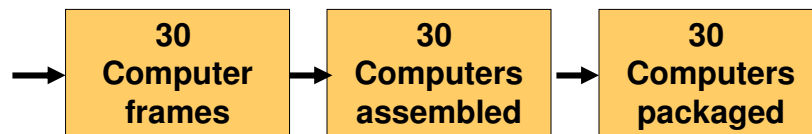
© Jeff Sutherland 1993-2007

### 3. Daily Meeting failure point

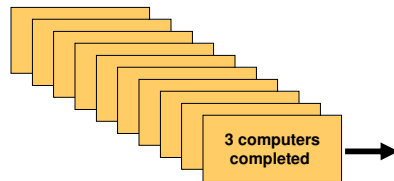
- Team is more than 9 people
- Every person does not speak
- Meaningful information is not communicated
  - Tasks started, stopped, completed
  - Estimates expanding
  - Impediments including personal issues
- Team does not self-organize
  - Must replan work based on information heard
  - Use 60 second rule to eliminate most impediments in the meeting
- ScrumMaster is a poor facilitator
  - Meetings longer than 15 minutes
  - Lack of dynamic facilitation

© Jeff Sutherland 1993-2007

### Small Teams, Small Batches



One failure means retest 90 computers



One failure means retest 3 computers

© Jeff Sutherland 1993-2007

## 4. ScrumMaster failure point

- ScrumMaster not dedicated and focused on team
  - Update burndown daily, remove impediments daily, deal with personal issues daily
- CEO or Product Owner is ScrumMaster with conflict of interest
- Lack of good personal and facilitative skills
  - Communicate, communicate, communicate
  - Listen, listen, listen
- Lack of good leadership skills
  - Tolerates distrust, lying (even if by omission), and hiding information
  - Fails to have a prioritized impediment list and eliminate impediments
  - Fails to deal with personal dynamics and other personal problems

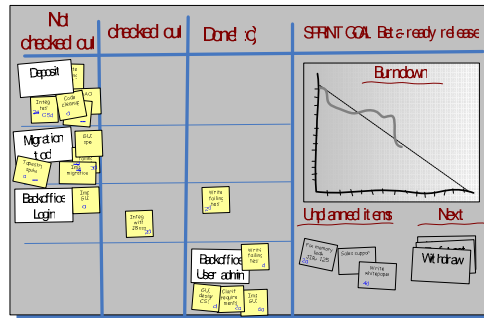
© Jeff Sutherland 1993-2007

## 5. Team failure points

- Lack of required technical or domain knowledge
- Not working on product backlog in priority order
- Failure to produce burndown, remove impediments, and increase velocity
- Working on anything not on the Sprint backlog
- Individual multitasking
- Team generating excessive work in progress
- Failure to test early
- Failure to improve engineering practices
- Lack of focused and dedicated resources

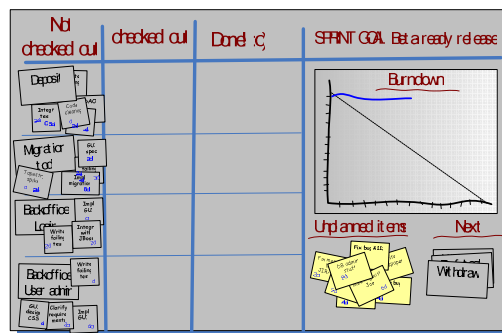
© Jeff Sutherland 1993-2007

## Taskboard warning sign #1



© Jeff Sutherland 1993-2007

## Taskboard warning sign #2



© Jeff Sutherland 1993-2007



## 7. Finally – management failure!

- Failure to have a business model that works
- Failure to provide adequate resources
- Failure to smooth out flow - Mura
- Failure to avoid stressing system - Muri
  - Violating sustainable pace
  - Disrupting teams during Sprint
- Failure to eliminate waste – Muda
- Failure to eliminate any impediments the team cannot eliminate
- *Failure to challenge teams to move beyond mediocrity*

*Note: Waterfall plans and time sheets are addictive. Managers may need rehabilitation and are at high risk of relapse.*

© Jeff Sutherland 1993-2007

## Why time sheets are waste ...

- *they demotivate developers, 10-15% loss of productivity is the minimum*
- *developers have to fake the time to fill them out properly*
- *erroneous data causes management to make bad decisions*
- *customers are deceived*
- *they focus the organization on phony data instead of production*
  - *there is no correlation between developer time and software production*
  - *there is no correlation between time spent and quality of code*
  - *there is no relationship between "quality people" and code production*
  - *for a single project worst/best coding times are 1/10*
  - *across many projects worst/best coding times are 1/25*
  - *the ratios above are the same for the worst and best Yale students*
  - *the quality of the code produced is completely independent of time spent*

See <http://www.joelonsoftware.com/articles/HighNotes.html>

© Jeff Sutherland 1993-2007

## Don't worry

- Doing Scrum wrong is better than not doing it at all
- Acknowledging the problem is the first step to the solution
- Fix one thing at a time
  - Measure effect of change
  - Bottlenecks change after removing one impediment
  - Don't waste resources fixing low priority impediments

© Jeff Sutherland 1993-2007

## Questions!



© Jeff Sutherland 1993-2007