



# SELF ORGANIZATION IN SCRUM

## HOW IT WORKS AND WHEN IT DOESN'T

With help from Google, Yahoo, Microsoft, IBM, Oracle, MySpace, Adobe, GE, Siemens, BellSouth, GSI Commerce, Ulticom, Palm, St. Jude Medical, DigiChart, RosettaStone, Healthwise, Sony/Ericson, Accenture, Trifork, Systematic Software Engineering, Exigen Services, SirsiDynix, Softhouse, Philips Medical, Barclays Global Investors, Constant Contact, Wellogic, Inova Solutions, Medco, Saxo Bank, Xebia, Insight.com, SolutionsIQ, Crisp, Johns Hopkins APL, Motley Fool, Planon, OpenView Venture Partners, Juske Bank, BEC, Camp Scrum, DotWay AB, Ultimate Software, Scrum Training Institute, AtTask, Intronis, Loyalty Lab, Version One, OpenView Labs, Central Desktop, Open-E, Zmags, eEye, Reality Digital, DST

Jeff Sutherland, Ph.D.  
Co-Creator of Scrum  
+1 617 606 3652  
[jeff.sutherland@scruminc.com](mailto:jeff.sutherland@scruminc.com)

CEO, **scruminc**  
powered by OpenView Labs  
Chairman, Scrum Training Institute



# Jeff Sutherland, Ph.D. [jeffsutherland.com/scrum](http://jeffsutherland.com/scrum)

- **CEO Scrum, Inc. and Senior Advisor, OpenView Venture Partners**
  - Scrum coach, mentor, and trainer to venture group and over 20 portfolio companies
  - 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**
  - **Achieved hyperproductive state in all companies**
- **Signatory of Agile Manifesto and founder of Agile Alliance**



© Jeff Sutherland 1993-2008

# Abstract

- Scrum was designed for hyper-performing teams that operate at 5-10 times the velocity and quality of waterfall teams. It is linearly scalable across geographies to any size.
- High performance depends on the self-organizing capability of teams. Understanding how this works and how to avoid destroying self-organization is a challenge.
- The Secret Sauce from complex adaptive systems provides some guidelines:
  - Shock therapy
  - Choice uncertainty principle
  - Punctuated equilibrium

# Agile Development

adaptability

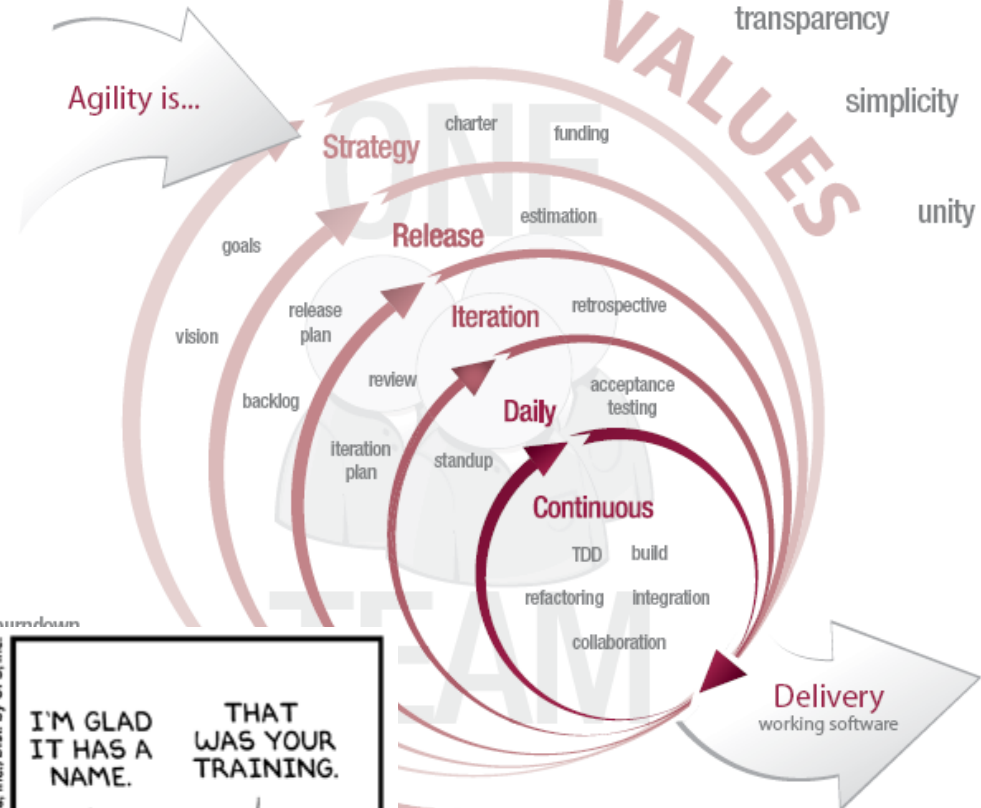
transparency

simplicity

unity

## VALUES

Agility is...



## ScrumButt



© Scott Adams, Inc./Dist. by UFS, Inc.

## Accelerate Success



# ScrumButt Checklist

## The Nokia Test by Bas Vodde

- Are you doing iterative development?
  - Sprints must be time boxed to four weeks or less
  - Software features must be tested and working at the end of an iteration
  - Sprints must start with an Agile specification
- Only 50% of Scrum teams worldwide meet these criteria



<http://www.slashphone.com/media/077129.html>

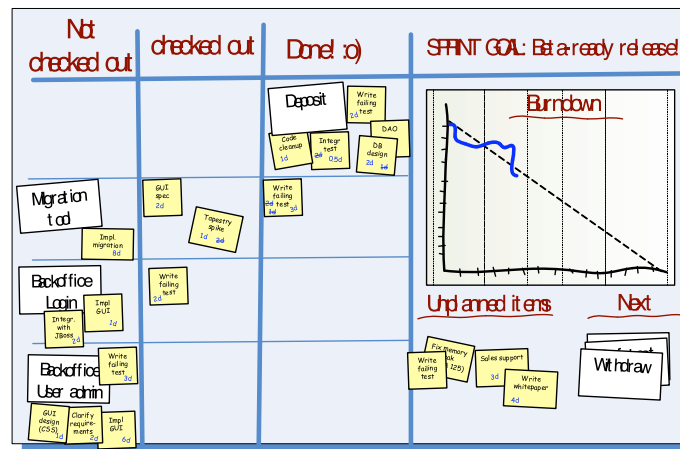
© Jeff Sutherland 1993-2008

# Are you doing Scrum?

## The Nokia Test by Bas Vodde

- Do you know who the product owner is?
- Is there a product backlog prioritized by business value that has estimates created by the team?
- Does the team generate burndown charts and know their velocity?
- Is the team's work free from disruption by project managers (or anyone else)?

Only 10% of teams worldwide meet these criteria.



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

© Jeff Sutherland 1993-2008

# Another way to measure ScrumButt

- Can you monetize extreme performance?
- Excellent Scrum - annual revenue up 400%
  - PatientKeeper
  - Others in Scandinavia I can't talk about
- Good Scrum - revenue up 300%
  - Companies in Scandinavia I can't talk about
- Pretty Good Scrum - revenue up 150% - 200%
  - Systematic Software Engineering - 200%
  - Google - 160%
- ScrumButt - revenue up 0-35%
  - Yahoo, most companies

# OpenView Venture Partners Strategy

- Investment partners practice Scrum
- Invest only in Agile projects
  - Use only market leading, industry standard processes – this means Scrum and XP
  - Ensure teams implement best Scrum practices
- Drive Scrum implementation at Board level
  - Ensure management is totally involved and understands Lean Product Development
- Many portfolio companies run senior management team, sales, marketing, client services, and support with Scrum.



# Investors want to know the “Secret Sauce”

- What is the secret recipe for building hyperproductive teams?
- First, implement basic Scrum practices and pass Nokia test.
- Second, management needs to get totally involved, understand team velocity, and remove impediments.
- Third, basic XP engineering practices need to be implemented
  - Test first development (maybe pair programming, TDD)
  - Continuous integration
- Then they are ready for the fun stuff

# Getting to hyperproductive state

- I define hyperproductivity as at least Toyota level of performance - 4 times industry average
- It used to take two years for a company to achieve 240% improvement
- We now see 300% improvement in 3 two-week Sprints in some portfolio companies
- The challenge is to consistently bring teams to a high performing state quickly and then use the secret sauce of self-organization to boot them to a hyperproductive state (5-10 times industry average)

## Velocity in Function Points/Dev month

	Scrum[1]	Waterfall[1]	SirsiDynix[2]
Person Months	54	540	827
Lines of Java	51000	58000	671688
Function Points	959	900	12673
Function Points per Dev/Mon	17.8	2.0	15.3

1. M. Cohn, User Stories Applied for Agile Development. Addison-Wesley, 2004

2. J. Sutherland, A. Viktorov, J. Blount, and N. Puntikov, "Distributed Scrum: Agile Project Management with Outsourced Development Teams," in HICSS'40, Hawaii International Conference on Software Systems, Big Island, Hawaii,

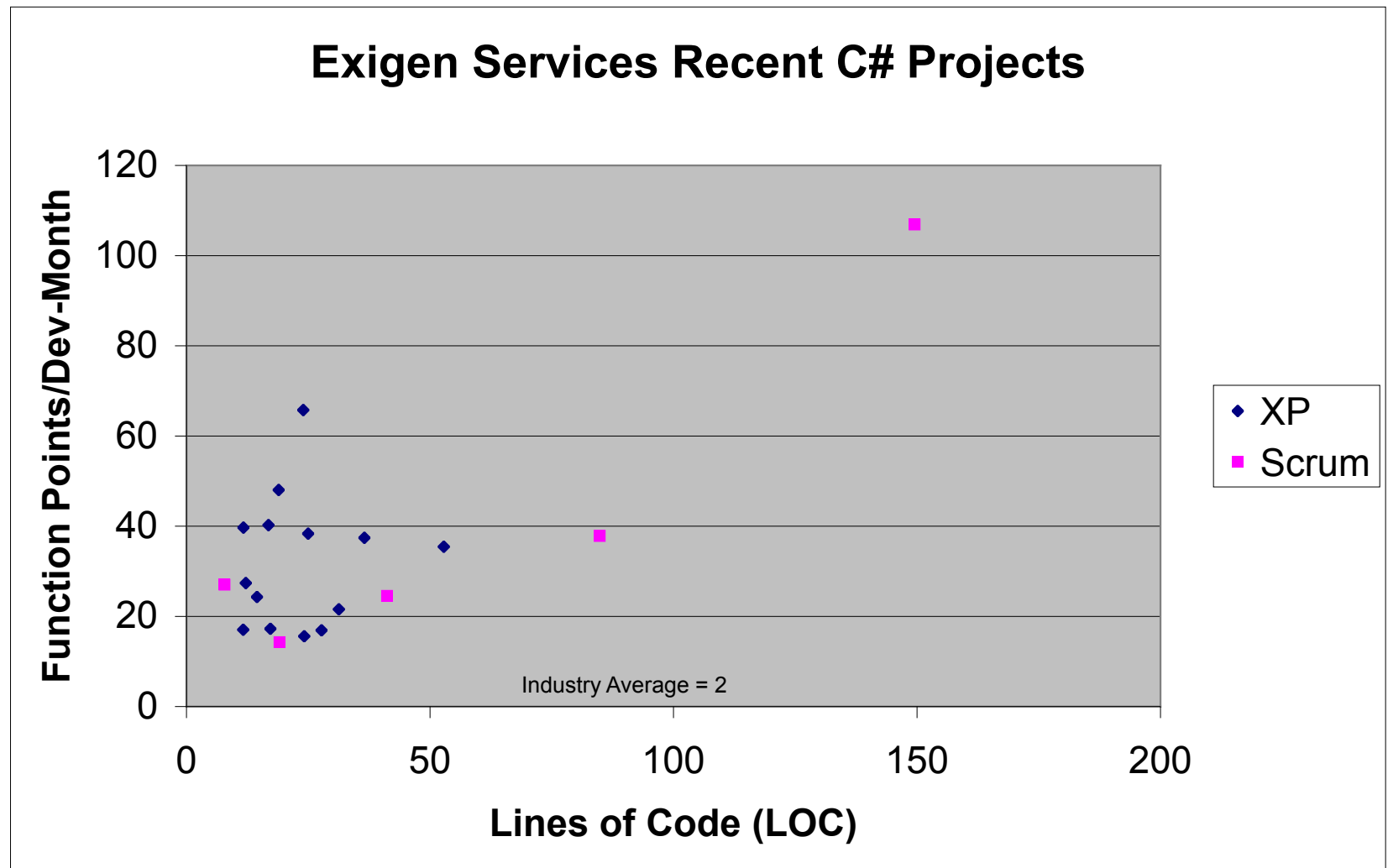
# Dutch Velocity vs. Russian Velocity

	SirsiDynix[2]	Xebia[3]
Person Months	827	125
Lines of Java	671688	<b>100000</b>
Function Points	12673	<b>1887</b>
Function Points per Dev/ Mon	15.3	<b>15.1</b>

1. M. Cohn, User Stories Applied for Agile Development. Addison-Wesley, 2004
2. J. Sutherland, A. Viktorov, J. Blount, and N. Puntikov, "Distributed Scrum: Agile Project Management with Outsourced Development Teams," in HICSS'40, Hawaii International Conference on Software Systems, Big Island, Hawaii,
3. J. Sutherland, G. Schoonheim, E. Rustenburg, M. Rijk. Fully Distributed Scrum: The Secret Sauce for Hyperproductive Outsourced Development Teams. Agile 2008, Toronto, Aug 4-8 (submission, preliminary data)

© Jeff Sutherland 1993-2008

# Russian projects velocity data suggests high velocity is not an accident



© Jeff Sutherland 1993-2008

# How can an Agile coach achieve hyperproductivity in a typical company?

- Typical company is waterfall or ScrumButt
- Management does not understand Agile and is not fully committed
- Multiple success points seen using an approach I call

## SHOCK THERAPY

- MySpace in Beverly Hills
- JayWay in Sweden
- Pivotal Labs in San Francisco

# Patterning

- Shock therapy is a form of patterning
- It is similar to martial arts training
- When you walk onto the mat in a dojo, the Sensei owns the mat
- You do exactly what he does, over and over again, until it is part of muscle memory
- Only when you have demonstrated mastery of the basic practices (and their principles) are you allowed to improvise
- Before you have gained discipline, centering and flexibility, you could become a danger to yourself and to others

# Case Study: MySpace

- MySpace has several hundred developers
  - about 1/3 waterfall
  - about 1/3 ScrumButt with Project Managers
  - about 1/3 pure Scrum
- Scott Downey, MySpace Agile coach repeatedly takes teams to high productivity state in a few weeks
  - Average time to 240% of the velocity of a waterfall team is 2.9 days per team member where the team includes the ScrumMaster and the Product Owner

# Starting up a Scrum Team

- Scrum, as a framework, permits teams a ton of options to customize it to their own environment. In Scott's experience, most teams just starting out are so overwhelmed with choices that they can't find a constructive way to start.
- It occurred to him one day that Scrum Teams are the customers of the ScrumMaster. We all know that customers of our enterprise don't really know what they want until they have seen it. So why do we expect Scrum Teams to know how to play Scrum if they haven't seen a prototype?
- So when Scott joins a team as their ScrumMaster, he issues a few non-negotiable rules (gently if possible, firmly if necessary).

# Starting up a Scrum Team

Scott describes it:

- My rules remain in effect until the team has met three criteria:
  - They are Hyper-Productive (>240% higher targeted value contribution)
  - They have completed three successful Sprints consecutively
  - They have identified a good business reason to change the rule
- The rules are roughly these:
  - Everyone on the team will attend a Scrum Training session.
    - I conduct an extremely condensed “Scrum at MySpace” course in about four hours, and the entire team comes together in one session. Until everyone has been trained, we won't begin our first Sprint.
  - Sprints will be one week long.
    - I justify this by pointing out that there is a reason geneticists study mutations in Fruit Flies instead of Elephants – they want to see the mutations quickly and adapt their studies accordingly. So do I.

# Teams hate one week Sprints (at first)

- I have been able to coax every team into giving me at least 4 one-week Sprints as a trial. Here's a favorite exchange of mine that almost always comes up – feel free to use it:
  - Engineer: "But I can't do anything in a week!"
  - Scott: "Then simple math suggests that you can only do four nothings in a month."
- Interestingly, by the time the teams have met the three criteria for changing this rule (Sprint length), only one team so far has ever elected to change it

# They will use my definition of “Done.”

- This is often one of the thorniest issues to iron out with a team, so I take it off the table until they have some shared success as a foundation.
- My initial definition of "Done" is this:
  - 1. Feature Complete
  - 2. Code Complete
  - 3. No known defects
  - 4. Approved by the Product Owner
  - 5. Production Ready

# All estimates are in Story Points

- Again, this one is sometimes met with a broad rolling of the eyes, but – so far, anyway – they have all eventually come around to this.
- It's usually at about week three when I can intentionally spark a debate over whether a card is a 3 or a 5, and then have the pleasure of watching the passion with which they debate these recently-meaningless values.
- I also make a point of shouting "BAA!" whenever they all vote the same value for a given card. My intent is to show them how often they actually agree in their vote.
- As the mood on the team lightens up, some teams begin scanning the other votes and "baa"ing like sheep when that happens.
- Only one has returned my "Baa!" with a "Humbug!" In any event, they all start having fun with it and that's important.

© Jeff Sutherland 1993-2008

# **We use a physical information radiator**

- Not only do I insist on a physical Information Radiator, but I have a basic template that I use initially for all teams.
- I choose the location of the Scrum Board unilaterally and use it as the focus of the Daily Stand-Up Meeting.
- When the team is first formed, I let them focus on the interaction with their teammates (the three Daily Scrum questions) and I move their cards across the Radiator's surface myself.
- Within a couple of weeks, they start moving cards themselves without being asked. This is usually my first indication that I can begin slowly stepping back and relaxing my demands.

# The “Sprint Meeting” is four hours once a week

- The first complaint of most Engineers is that they perceive Scrum imposing a highly disruptive schedule on them, with more meetings than they somehow think they have ever had before.
- To minimize this common concern, I consolidate everything but the Daily Stand-Up meetings into a single four-hour meeting (covering Sprint Review, Retrospective & Sprint Planning).
- Within a few weeks, the teams usually only need a couple of hours. And by the end of about eight Sprints together, the meeting is becoming ninety minutes or less in duration for a one week Sprint.

# My initial “Sprint Meeting” agenda

## is:

1. Demonstration: where the Team shows the Product Owner the working software they completed and “earns” Story Points if everyone concurs.
2. The Retrospective comes next, aided by a bunch of metrics that I track. I only track whole-team metrics, never individual metrics.
3. Product Backlog Presentation follows, during which the Product Owner discusses the content of the Backlog at that point in time.

The Team is free to question motives, suggest alternatives and add requirements at this point. I reject any improperly formed User Stories on behalf of the team.

# Sprint Planning:

4. Estimation and Negotiation signals that the meeting is nearly complete. They happen in a single motion with my new teams, though more mature teams eventually choose to split these activities into unique meetings.

The Product Owner participates in an *advisory-only* role during this phase of the meeting. I spend most of my time in this phase trying to keep the teams from unnecessarily breaking Story Cards down into task sequences, which some tend to do. The INVEST mnemonic is really handy here.

5. Sprint Backlog Commitment is the final act of the Sprint Meeting. In the first few Sprints, I literally read aloud what "Commit" does and does not mean so that there is no doubt in anyone's mind. Once the team commits to the work, the meeting adjourns.

# Multi-Tasking is Forbidden

- Work must be done in Priority Order.
  - Some Engineers understand this right away. Others feel most productive or fulfilled when they have multiple projects in progress and they don't appreciate my pointing out that there is no value in incomplete work – but point it out, I do. And often. I insist and enforce that they work on cards without multi-tasking and in priority order.
- I also have standard layouts that I use for their initial Sprint Planning Boards, User Stories, Story Cards, Burndown Charts and Velocity tracking.
  - I take full responsibility for entry and management of the (horrible) Scrum tool that we have at the moment so that they don't have that misery to deal with on top of learning everything else.

# Friendly smile and a “please”

- As I said in the beginning, I do try to get all of this done with a friendly smile and a "please" but generally have to insist -- sometimes quite forcefully -- to get all of them moving.
- Although the beginning is rocky, we usually start laughing and having fun in the meetings within a week.
- And as they become more comfortable and competent with Scrum, I relax my grip on some of the rules and let them redesign their environment to their own liking – so long as they continue to respect the principles of Scrum.

# Three key reasons for success ...

1. I find the biggest, nastiest problem that the team has and solve it within a day or two of the first Sprint Meeting.
  - Some teams quickly volunteer this problem to me in their first Retrospective while other problems require observation, careful listening and behind-the-scenes reconnaissance to tease out.
  - Especially for those teams who haven't worked directly with me yet, having that very large problem go away underscores that they are important to me, that I take them seriously and that I am working hard to make their world a better place.

# Key reason #2

2. Since I am the Master Scrum Master/Scrum Evangelist for the entire company, I am almost never their permanent Scrum Master.

- This gives me the freedom to create a bit (but only a very small amount) of "Us vs. Him" atmosphere at first.
- It causes the team to bond in an entirely new way than they have before, and also sets up their permanent Scrum Master to be the "good cop" down the road.
- This also allows me to be more firm about, for example, standing during Stand-Up, keeping SP estimates private until the laydown during estimation, etc.

# Hyperproductivity is the target

- I generally have to bow out and move on to another team after 6-12 weeks, by which point they are functioning very well and are (on average) around the 500% mark.
- Overall, most teams tolerate this approach very well and learn good habits more quickly – even if it does leave me feeling a bit a schoolmarm.

# Key reason #3 - facilitating self-organization

3. I like Socrates' approach. When I see something going wrong – say, someone sitting during the Daily Stand-Up – I don't always address the transgressor directly. Instead, sometimes I stop the meeting and ask the team, "Team, do any of you see something going wrong with our meeting right now?"

Ironically, it is almost always the most skeptical person who is the first to correct the insolently perched teammate. Soon, they start calling one another on leaning or sitting long before I stop the meeting and ask what's going wrong. It helps them begin to police themselves so that I don't always have to be around to elicit good behavior.

# Hyperproductive in four weeks

- This is roughly how I have pulled teams into hyper-productivity in as few as four weeks.
- [Note: One of Scott's co-workers calls him "The Scrum Whisperer."]
- I have one team that has achieved 1,650% higher targeted value contribution per week after just four months (16 Sprints) together. We are pretty proud of those numbers.
- I've also noticed that teams using this "quick format" approach tend to hit their Velocity elbow much sooner, giving Product Owners a greater and more stable view of the roadmap than teams who use longer Sprints or spend their inaugural period hashing out "where do we want the Scrum Board to be located".

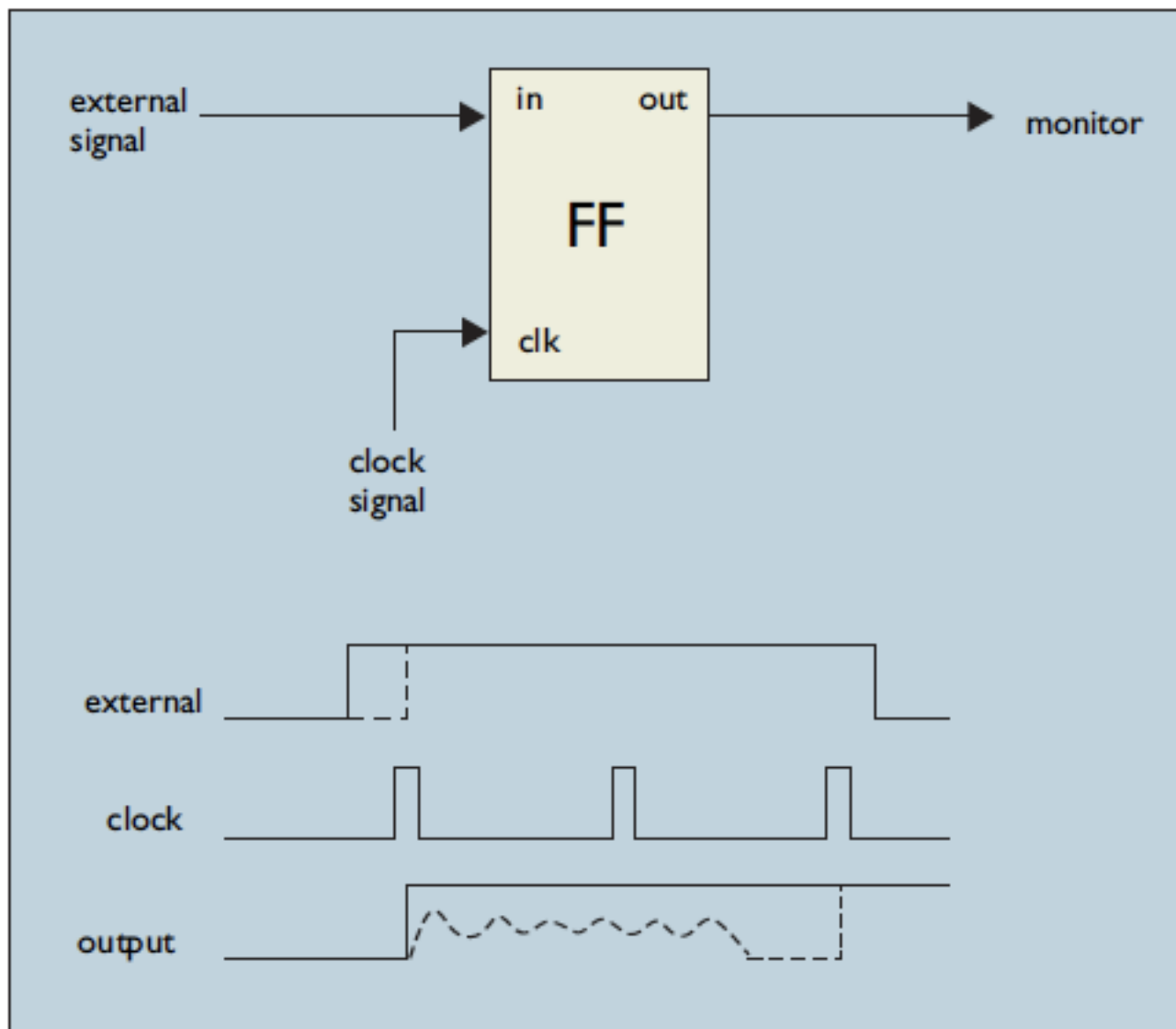
# Followup on the kick start ...

- It's a fairly large culture shock for most teams and doesn't yield a lot of "let's go to lunch" invitations at first.
- But, per my VP of Engineering, "They only hate you for about 2-3 weeks. Then they're indifferent to you for another few weeks."
- "Then they scream bloody murder if I try to take Scott away from them."
- I do stay in touch with teams that I've kick-started like this and, with one notable exception, they have all continued their trend of improvements in my absence.
- My techniques are always evolving ...
  - Scott Downey
  - <http://www.MySpace.com/PracticalScrum>

# Cosmic Stopping Problem

- Early computers crashed randomly. It was attributed to “cosmic rays” that corrupted memory.
- Computer logic circuits occasionally locked up, requiring a complete restart of the computer. These lockups could cause the loss of considerable work on long computations. Worse, they made machines unreliable for real-time, safety-critical applications.
- These lockups never occurred when the interrupts were off.

Denning, Peter. The Choice Uncertainty Principle. Computer 50:11, p. 9, November 2007



If Flip Flop is in indeterminate state at clock signal bad things happen.

# Metastable State

- At less than 200MHz modern flipflops have zero chance of metastable states
- At 300MHz they disrupt computer every two weeks
- At 400MHz there is a problem every three minutes
- Solution was to force computer not to read unless state was stable

# **Extremely valuable, in some situations, to know which state you are in**

- Equivalent: two people passing on a sidewalk not knowing which way to pass. They must stop until they decide

# Choice Uncertainty Principle is a Cosmic Problem

- It occurs at every level of the universe (Heisenberg Uncertainty Principle)
- Well-run Scrums handle this at every step
  - Don't accept backlog that is not ready
  - Minimize work-in-progress
  - Stop the line when bad things happen and fix it so it can never happen again
  - If it is not "Done" put it back on the product backlog
- Where was Scott Downey handling the Cosmic Stopping Problem in his Scrum implementation at MySpace?
- This requires forceful stopping. Could you do this at Google?

# Forceful stopping leads to punctuated equilibrium

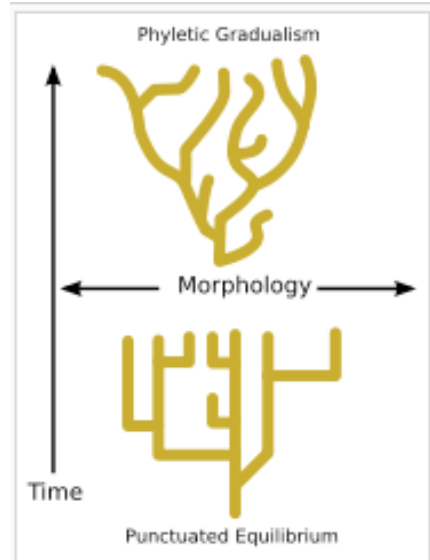
## Punctuated equilibrium

From Wikipedia, the free encyclopedia

**Punctuated equilibrium** is a theory of [evolutionary biology](#) which states that most [sexually reproducing](#) populations experience little change for most of their geological history, and that when phenotypic evolution does occur, it is localized in rare, rapid events of branching speciation (called [cladogenesis](#)).

Punctuated equilibrium is commonly contrasted against the theory of [phyletic gradualism](#), which states that evolution generally occurs uniformly and by the steady and gradual transformation of whole lineages ([anagenesis](#)). In this view, evolution is seen as generally smooth and continuous.

In [1972](#) paleontologists [Niles Eldredge](#) and [Stephen Jay Gould](#) published a landmark paper developing this idea. Their paper was built upon [Ernst Mayr's](#) theory of [geographic speciation](#), I. Michael Lerner's theories of developmental and genetic homeostasis, as well as their own empirical research. Eldredge and Gould proposed that the degree of gradualism championed by [Charles Darwin](#) was virtually nonexistent in the fossil record, and that stasis dominates the history of most [fossil](#) species.



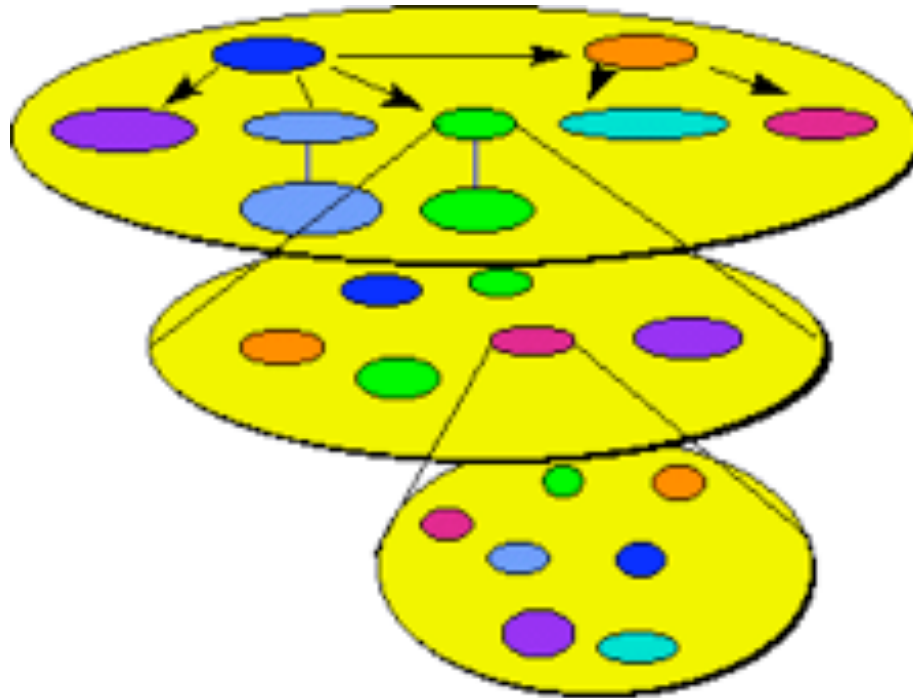
Punctuated equilibrium, bottom, consists of morphological stability and rare bursts of evolutionary change.

© Jeff Sutherland 1993-2008

# Software evolution depends on punctuated equilibrium

- Many components must make small changes and only when they appear together in the right configuration does the software system jump in functionality.
- The jump is often fast and surprising after a long period of stability.
- The team (and individuals on the team) must avoid many behaviors and execute on a few carefully selected behaviors.
- Thus self-organization for performance is dependent on not doing many things and carefully doing a few things in right timing.

# First Scrum Team Component Model

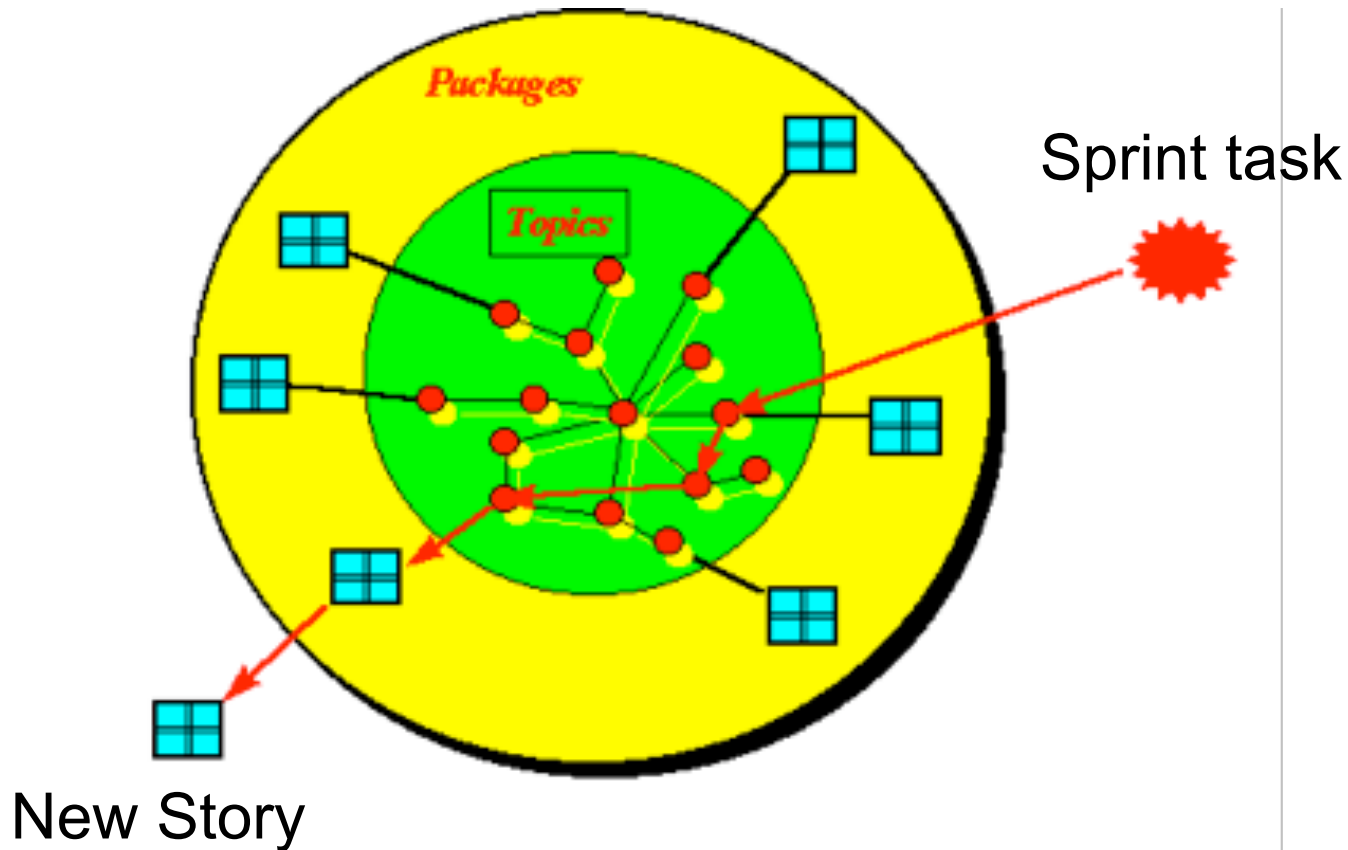


## Business Object Component Architectures: A Target Application Area for Complex Adaptive Systems Research

[Jeff Sutherland](#), SVP Engineering & Product Development, IDX Systems Corp., 1998

© Jeff Sutherland 1993-2008

# Next task must maximize probability of user seeing system change behavior



Dennett, D. C. *Darwin's Dangerous Idea: Evolution and the Meanings of Life*. Simon & Shuster, 1995.

Sutherland, Jeff. *Agile Can Scale: Inventing and Re-Inventing Scrum in Five Companies*. The Scrum Papers, 208.

© Jeff Sutherland 1993-2008

# The Road to Hyper-Productivity

- Based on Complex Adaptive Systems theory
- Need an architectural metaphor which allows rapid refactoring at all levels of granularity without introduction of defects
- All developers must understand architecture and state of all components
- Make the right changes to the right components in the right order to maximize the speed of appearance of new features - punctuated equilibrium
- Solve the Cosmic Stopping Problem by not working on requirements that are not ready, avoiding work that is not done, minimizing work in progress, avoiding obstacles through self-organization, and ...

*Shock therapy can reduce time to initial target velocity to 3 or 4 Sprints (a few weeks)*



**All Black Video**

© 1993-2008 Jeff Sutherland v8.7

# Questions?



© Jeff Sutherland 1993-2008