9 hours ago
Mega U.S. Solar Project Threatened by Unlikely Foe
WILLIAM PENTLAND *Clean Beta*

Yesterday
Someday Berkshire Hathaway May Split Its Stock, Pay A Dividend, Forsake Omaha
ROBERT LENZNER *StreetTalk*

Steve Denning

RETHINK

MY PROFILE MY HEADLINE GRABS MY RSS FEED

Scrum Is A Major Management Discovery

Apr. 29 2011 - 11:24 am | 2,584 views | 0 recommendations | 8 comments

A reader remarked about my recent article on Salesforce.com, "Frankly, this looks like a plug for Scrum." Well, yes.

If there was a Nobel Prize for management, and if there was any justice in the world, I believe that the prize would be awarded, among others, to Jeff Sutherland, Ken Schwaber and Mike Cohn for their contributions to the invention of Scrum. Until recently, like most people, I had never heard of Scrum. This is not surprising, as it is rarely mentioned in general management textbooks or discussed in business schools. I came across Scrum several years ago, almost by accident.

My discovery of Scrum

At the time, I was working on a book focused on resolving an old management conundrum: how do you combine rapid innovation with disciplined execution? I was proceeding by asking people if they knew about any such workplaces where this was already happening.

I surprised to find that an unusually high proportion of the workplaces

MY ACTIVITY FEED

Show all activity

ACTIVE CONVERSATION 7 hours ago
Scrum Is A Major Management Discovery
8 Comments in the last 2 days **4 Called-out**

STEVE COMMENTED 7 hours ago
"Dear Craig, Thanks for these helpful clarifications. Obviously the few sentences in my brief article about Scrum do not give, and..."
Posted to **SCRUM IS A MAJOR MANAGEMENT DISCOVERY**

STEVE CALLED OUT 8 hours ago
craiglarman
Commented on **SCRUM IS A MAJOR MANAGEMENT DISCOVERY**
"steve, hi, and thx for article. as a certified scrum trainer and one of the early scrum writers/coaches (and as coincidence,..."

STEVE COMMENTED 13 hours ago
"Dear hwallop, Thanks for these comments. "It's good to view the world as two leans." I agree very much. That's a..."
Posted to **SCRUM IS A MAJOR MANAGEMENT DISCOVERY**

MOST POPULAR

that I heard about were in software development. Initially I didn't pay it any attention. After all, these were geeks, and they talked in a strange, barely comprehensible vocabulary. What could I possibly learn about management from people who had, I imagined, gone into computing because they preferred machines to people?

A colleague, Hans Samios, a manager at Intergraph Corporation in Alabama, contacted me and suggested that I check out what was happening in software development firms, under the names of Agile and in particular, the practices known as Scrum. I had never heard of Scrum, but I decided to check it out.

The history of Scrum

I learned that the first full implementation of Scrum had occurred in 1993 when Jeff Sutherland along with John Scumniotales and Jeff McKenna implemented Scrum at the Easel Corporation.

They drew on the inspiration of the classic 1986 HBR article "The New New Product Development Game," where Takeuchi and Nonaka had compared a new holistic approach to innovation to the sport of rugby, where the whole team "tries to go to the distance as a unit, passing the ball back and forth". This paper in turn drew on a long experience of iterative methods.[1]

In 1995, Sutherland and Schwaber jointly presented a paper, "The SCRUM Development Process," at Object-Oriented Programming, Systems, Languages & Applications (OOPSLA) Conference '95 in Austin, Texas, its first public appearance.

In 2001, Sutherland, Schwaber, and fifteen colleagues got together in Snowbird, Colorado, and drafted the Agile Manifesto, which became a clarion call to software developers around the globe to pursue this radically different type of management.[2]

Since then, Sutherland, Schwaber, and their colleagues have gone on to generate thousands of high-performance teams in hundreds of companies all around the world under the labels of *Scrum* and *Agile*.

Other important contributions to the Scrum practices were made by Mike Cohn with the development of user stories as the principal tool for describing client-oriented goals of work, along with the development of story points as a way of measuring the quantity of work and the velocity of teams.

What are the practices of Scrum?

When I checked out what was going on in these companies, I could see that underneath the cover of an esoteric terminology, these software developers had discovered a solution to the problem of combining disciplined execution of high-level intellectual work with continuous innovation.

If you extract the practices of Scrum from the esoteric vocabulary in which it is expressed for software developers ("sprints", "burndown charts", "product owner", "scrum-master") it comprises in essence the following core practices:

1. Organize work in *short cycles*:
2. The management *doesn't interrupt* the team during a work cycle.
3. The team reports to *the client*, not the manager:
4. The team estimates *how much time* work will take:
5. The team decides *how much* work it can do in an iteration:
6. The team decides *how* to do the work in the iteration:

7. The team *measures its own performance*:
8. Define work goals *before* each cycle starts:
9. Define work goals through *user stories*:
10. Systematically *remove impediments*:

None of these practices is by itself new. What is new is doing all the practices together in a disciplined way of getting all work done.

When the practices are generalized in this way, beyond software development, they can be collectively described as dynamic linking, to distinguish them from the traditional practices of hierarchical bureaucracy, where individuals reports to bosses to produce outputs.

Teams using the practices that Sutherland and his colleagues had pioneered have been unexpectedly productive. These were not just improvements where the teams were just *slightly* better than the norm. The best teams routinely obtain productivity increases of 200 to 400 percent, changes that are potentially *industry-disruptive* improvements.

Some mixed implementation results of Scrum

Nevertheless, despite the *enormous potential* that individual teams and departments have shown with Scrum, the overall picture of *implementation* has been quite mixed. More than 70% of Scrum implementations have failed to achieve their goals.

Most of these implementations with mixed results, which Sutherland derisively calls “Scrum-butt”, are examples of a failure to implement the full array of Scrum practices. When only some of the practices are implemented, such doing the work in short cycles but interrupting the team during the cycle, the potential gains in productivity don’t occur.

In part these problems of implementation have flowed from the way Scrum is sometimes viewed and introduced.

For instance, according to the Wikipedia, “Scrum is an iterative, incremental framework for project management often seen in agile software development, a type of software engineering.”

When you try to embed Scrum as a project management framework within a larger setting of a traditional management of hierarchical bureaucracy, there are inevitable tensions. Usually the prevailing culture of hierarchical bureaucracy is triumphant.

This experience sometimes leads people to believe that Scrum is just another management fad that didn’t work.

What the experience really shows is the opposite: that hierarchical bureaucracy is a work culture that no longer fits the marketplace of 2011. Scrum works. It’s the traditional management culture that doesn’t work.

The potential of Scrum as a different way of managing

The real question is: what can you accomplish if you execute Scrum well? The answer is now apparent with the results shown by Salesforce.com, which has been growing by 41% over a sustained period, and whose CEO, Marc Benioff, has been identified by Forbes as the most valuable CEO on the planet.

A turning point for Salesforce.com came in 2006, when the leadership realized that as the firm had grown, innovation in the firm had started to slow down. Instead of doing what most firms do, i.e. trying harder with more-of-the same kind of management, Salesforce.com adopted a

different kind of management: Scrum.

That was the point of my article Six Common Mistakes That Salesforce.com Didn't Make.

Unlike many firms that have tried to implement Scrum, the leadership at Salesforce.com saw that Scrum involved not just the adoption of a new business process, or a framework for managing software development, but rather as a fundamental transformation of the way work was managed in the company. They realized that they were introducing a new way of thinking, speaking and acting in the workplace for both managers and workers. They committed to it boldly and the results have been extraordinary.

Application of Scrum beyond software

The success of software development at firms like Salesforce.com [CRM], along similar customer-driven iterative methods in auto manufacture at firms like Toyota, has led to the spread of this different way of managing to related fields.

- The Quality Software Engineering group at IBM [IBM] is responsible for software development processes and practices across the company. As part of the effort to promulgate Scrum in developing software, an iterative process of working was adopted for doing change management.
- At the Chicago software firm Total Attorneys, iterative work patterns were so successful that they spread to the staff of call centers: small cross-functional teams work in cycles of three weeks.
- At the Danish software firm, Systematic, iterative methods have been spreading from software development to other parts of the firm.
- At the Swedish software firm Trifork, iterative methods have spread from software development to conference management.
- And OpenView Venture Partners, a Boston-based venture capital firm, has expanded client-driven iterations into consulting and finance.

Once a firm sees the dramatic benefits of small client-driven iterations in one area, it becomes natural to ask: Why not do all work in this fashion?

What to call these radically different management practices? In manufacturing, they are known as Lean. In software development they are known as Scrum and Agile. When they are applied to management in general, none of those terms is really applicable, as they carry the baggage of their origins in software and manufacturing. The Agile Manifesto for instance dwells on the goal of work as "working software", which is fine for software development, but not relevant to other sectors.

Scrum, Agile and Lean are in effect subsets of a radical shift in management more generally, or what I have called *radical management*.

Steve Denning's most recent book is: *The Leader's Guide to Radical Management: Reinventing the Workplace For the 21st Century* (Jossey-Bass, 2010).

[1] The early history of iterative approaches in software development is described in detail by Craig Larman and Victor Basili in "Iterative and Incremental Development: A Brief History." Computer, 2003, 36(6), 47–56. Iterative approaches to work build on the 1930s work of Walter Shewhart, a quality expert at Bell Labs who proposed a series of short plan-do-study-act (PDSA) cycles for quality improvement: Shewhart, W.

Statistical Method from the Viewpoint of Quality Control. New York: Dover, 1986. (Originally published 1939.)

[2] For the twelve principles behind the manifesto: http://agilemanifesto.org/principles.html.

Previous Post:
How Chicken Soup For the Soul Dramatically Expanded Its Brand

Next Post:
Bursting The Defense Bubble: End The Entitlement Mentality

More on Forbes Right Now

Related Posts

What On Earth Is Scrum?

Six Common Mistakes That Salesforce.com Didn't Make

How Marc Benioff of Salesforce.com Became The Most Valuable CEO Of All

The Changing Geometry of Organizations: From Economies of Scale to Economies of Small

Does Radical Management Apply To Manufacturing?

LIVE STREAMS

Related streams

Entrepreneurs
Leadership
Lists
Tech

Comments

DISPLAY

Called-Out Comments | **All comments**

Active Conversation

4 Called-out Comments, 8 Total Comments

Post your comment »

3:09 pm on 04/29/11

bennyflint

Very thoughtful article. Having worked in a Scrum process for several years, I have often thought about how useful it would be for pursuits other than software development.

I would, however, like to point out that traditional Scrum teams intentionally do not decide how much time work will take. Rather, work is estimated in terms of relative effort, and tasks are most frequently measured with what we call 'story points.' There are good reasons for this, which I outline below, but first it's important to know how such a process works.

Frequently, a well-known, routine task can be used as a baseline and assigned an arbitrary story point value, say 5. Other tasks are then assigned points based on subjective assessment of its relative effort. So something that is "about twice as hard" as a 5 might be scored as an 8 or 13 (we geeks like to use funny scales like the Fibonacci sequence).

It's important to note that the team–not management–decides story points by consensus, and their decisions are subjective, taking into account past experience, the quality of the task's definition, and any known or potential impediments.

So why is this distinction between time and story points important?

First, it encourages communication and collaboration within the team. The process of assigning story points is a thoughtful and empowering exercise that allows a team to share knowledge and better define a problem and its potential solutions.

Second, estimates based on relative effort are more accurate than arbitrary time estimates. Tasks with large point values are a good indication that a task ought to be broken down into smaller, more well-defined efforts. It's much easier to estimate smaller units of work than large, undefined projects. With traditional time-based estimates, managers tend to throw out a ballpark number to higher-level managers, often padding the time they allot their team, making their estimates wildly inaccurate. Smaller, point-based estimates make projections and planning more accurate overall.

Also, using a relative effort scale explicitly takes into account the risks involved with a project, something that is sorely missing from time-based estimates. Risks and unknowns are necessarily confronted and dealt with up front. This is like handing a team–and management–a map that indicates

"There be danger here," rather than equipping them with a weak flashlight and hoping for the best as they sally forth. This not only makes estimates more accurate, it results in a better end product because impediments are be identified removed in a quality manner, as opposed to being surprised and having to either take shortcuts around an impediment or move a delivery date.

Additionally, a relative scale like story points is measurable, especially over time. If someone–an executive perhaps–wants gauge productivity, they might ask the question, "How much can this team accomplish in three weeks?" If you're using time-based estimates, the answer would be, "about three weeks of work." Pretty meaningless. With point, however, it's possible for a team to look back, measure their accuracy and adjust accordingly (as a side note, the exercise of "looking back" is an explicit part of Scrum called a 'retrospective.'). Over time, this becomes a measurable velocity of a team's productivity. And as the saying goes, "What is measured tends to improve."

This discussion may cause people to think, "Holy cow, that sounds complicated!" The truth is, it's not. With a little practice, the process becomes quick, easy, engaging, and (dare I say?) fun. And, as the author said, it works.

-Ben Flint
Software Engineer, eHarmony.com

Log in to Reply Flag for abuse

3:14 pm on 04/29/11

STEVE DENNING
RETHINK

Ben,
Thanks so much for the comments.
All terrific points.
Steve

Log in to Reply Flag for abuse

8:22 pm on 04/29/11

whallop

Hi,

I agree with the comment about the estimating. The rummaging around for a common estimate intuitively generates a handle on the complexity of the story. Too big a story or too much rummaging are indicators that the story needs to be reshaped and shrunk.

Dynamic linking, eg software dynamic linking, isn't value added. There's a value-added baton passed here. This links to the PDSA mentioned – a dynamic learning cycle looping around "go look and see". Sutherland, ex-airforce pilot uses Boyd's OODA, all 4 bits firing at the same time – dynamically, like sitting in the fighter seat fighting.

Agile pattern matches startups. A common startup term is pivot. One learns new and pivots the startup to grab the new seen opport.

It's good to view the world as two leans.
Toyota lean empowers bottom up with an always improving kata. Other lean tends to empower coaches who add gadgets/tools as part of imparting their expertise. Toyota often invented the tools and has coaches, but doesn't foster the coaches so much and uses the tools to support the process.

Other lean implicitly adds 'priests' managing the process with 'their arcane' gadgets/tools.

Finally, kicking at the ignored cat in the corner, nobody remembers Johnson. Eg google: "Manage by Means, Not Results" by Johnson, H. Thomas. There are stories in the cultures' forgetting/ignoring.

The startup, agile, Toyota lean, manage by means – how things are done. The other rotters use results – manage by results. Priests end to fudge lean with tools towards results. Or for scrum, fudge it towards kanban(ed) results.

Log in to Reply Flag for abuse

9:29 am on 05/01/11

STEVE DENNING
RETHINK

Dear hwallop,
Thanks for these comments.
"It's good to view the world as two leans." I agree very much. That's a nice way of putting it. Clearly I am talking about the former kind, where people are acting as coaches and enablers, rather than priests managing the process.
On whether it is managing by means or managing by results, this deserves a longer answer. I tend to think that this is a false choice. It's a question of having both the right goal or results in mind (delighting clients) and the right means (enablement rather than control). It's a systemic shift in thinking, speaking and acting that is involved. I will write more on this.
Thanks again for the very interesting comment.
Steve

In response to another comment. *See in context »*

Log in to Reply Flag for abuse

3:32 am on 05/01/11

lancerkind

Scrum can also be applied to producing works of art such as novels: http://www.LancerKind.com/using-agile-to-write-agile-noir/

Log in to Reply Flag for abuse

9:04 am on 05/01/11

STEVE DENNING
RETHINK

Dear lancerkind,
Thanks for your interesting comment and link. I found a Scrum-like process was very useful in the editing of my last book, The Leader's Guide to Radical Management.
I believe that Mike Cohn also used a similar process for his book, Succeeding With Agile.
Steve

In response to another comment. *See in context »*

Log in to Reply Flag for abuse

1:08 pm on 05/01/11

craiglarman

steve,

hi, and thx for article. as a certified scrum trainer and one of the early scrum writers/coaches (and as coincidence, the author you cite in footnote #1), i hope you don't mind if i (1) highlight a factual error, (2) make some deeper points about scrum, and (3) address some key "what's missing", to help reduce incomplete or incorrect scrum descriptions. because you are a management thought leader, the more accurately we can all describe the key scrum ideas to colleagues and clients, the better for all.

=== factual error?
"Define work goals through user stories"

in scrum, the Product Backlog (of feature goals for the product, that a team does) officially (i'm writing as someone empowered to communicate the official scrum description) contains "items" — a very general name on purpose. in scrum, one may use any requirement model seen fit — and change it too. stories, use cases, planguage, … no limitation. in scrum, a Team delivers an *item*. because "user stories" are popular and useful, it is easy to confuse what is officially in scrum or not. scrum is silent on the requirements approach.

=== deeper points?
i raise the above minor point about "user stories" because it leads a more important, deep scrum point: scrum is silent on the "how" of most practices, and emphasizes an empirical flexible improvable approach regarding techniques. this intentional lack of prescriptive techniques is a key strength of scrum — one to carefully protect and explain.

similarly (correcting what bennyflint wrote above in his comment), scrum officially does *not* do "story points" (relative effort points). one may use person days, story points, gummy bear units, or anything found useful. scrum is silent on the required estimation unit.

this is a critical idea: scrum is silent almost all specific techniques — on purpose. it is low-prescriptive framework that emphasizes empiricism: try any technique (in requirements, estimation, development, …) and try changing it.

there exists another agile method called "Extreme Programming" (XP). some of the ideas in *XP* include "user stories", "story points", "test-driven development", etc. some people have incorrectly attributed these (good) XP practices to scrum, which leads to some confusion about was is in/out of scrum, and they miss the deeper point of low-prescription-on-purpose in scrum.

i encourage any reader of this comment to pls not mis-read my point as "user stories are not good", etc, or mutate this into a discussion of favorite techniques. not the point.

so, scrum has principles and mostly avoids prescriptive techniques. but OTOH, you may be familiar with "frameworks" in org design or product dev that are just a set of high-level principles or vague meta-models (such as cmmi), which ultimately can be boiled down to "do skillful things; stop doing bad things." that's all well and good, but these are just vague warm fuzzies — like my grandma. but they have no teeth — also like grandma. (ok, i made that part up)

scrum strikes a middle ground and is a sweet spot as a framework between too prescriptive and too vague. it has a few unequivocal concrete techniques (the events, roles, artifacts) that provide enough concrete structure that it can be clearly grasped and made operational, and yet not too much. it's a Goldilocks framework

=== what's missing?
there's no single correct "what's missing" list related to the summary list you write above regarding scrum, but the following points come to mind as important to highlight in an introductory list:

ONE. cross-functional multi-learning team — early in any scrum introduction, i explore the name "scrum" itself for its key metaphorical implication: a cross-functional team (CFT) that moves the ball down the field together. i know you allude to this, but the implications need to be spelled out, rather than left metaphorical. and of course related to CFT idea in the NNPDG paper is "multi-learning" (not only having 1 specialization, and learning new ones), and the corollary that the only job title is "team member". i.e., in scrum there are no single-function teams that do partial work and related handoff of WIP — no requirements team, no business analysis team, no UI design team, no test team, no doc team, no architecture team, and no designers, no testers, etc. rather, a team does everything (without hand off to other teams) — analysis, design, development, testing, documentation — to deliver an *item*.

"CFT with multi-learning team members that do everything" is such a critical, central idea in scrum. i'm sure you know this, but the above scrum introduction is strengthened by making the point and implications very clear. and this point has *huge* organizational design implications; e.g., the dissolution of single-function departments (the architecture group, the test group, …), the end of HR policies that emphasize single job title and single-specialist career path, etc.

TWO. agility — scrum is an "agile" method, which is more than simply an "iterative" method. a key implication is "adaptive planning", meaning that the Product Owner (business client) can *change* the content and priority of items, each sprint. based upon learning and feedback (internal or external), the Product Owner can "decide as late as possible" to remove goals, add goals, or re-prioritize goals so that there is business agility in the product features (versus following a fixed content specification). this is a key scrum idea which also implies the need to lower the cost of change — to make change easy and then to embrace change as a competitive advantage.

THREE. transparency, inspect, and adapt — these principles inform and underly so many aspects of scrum. i'm sure you know this and you allude to it, but an introduction to scrum should highlight "the three legs of scrum". when one grasps scrum well, one realizes that most of its mechanisms are for increasing transparency with structured inspection mechanisms, and ways to adapt. e.g., the Daily Scrum, the Sprint Review, the Sprint Retrospective. etc

FOUR. the "work" is an increment of customer-centric end-to-end functionality that is "done" — you mention "work" in the scrum description, but what is meant by scrum "work"? this needs to be spelled out to someone new to scrum because "work" could easily be misinterpreted as "the UI design work", "the test work", etc. i.e., partially done work or WIP. rather, a work item in scrum means a complete end-to-end customer goal, that is shippable and "done" — it is implemented, tested, documented, and done in every imaginable, and ready for delivery. btw, all this implies the ability to *split* customer-centric requirements into very small increments of functionality (product breakdown structure), rather than splitting into tasks (work breakdown structure) — and that has deep implications

FIVE. done — "done" is a formal concept in scrum. any item "done" in a sprint needs to be completely done across all disciplines or functions (analysis, dev, test, doc, …). "done" is formally defined by the parties in scrum, and the team is required to meet this unambiguous definition of "done", so that when the Product Owner asks, "Is it done?" then both parties have the same definition, and it is really shippable — no more wrap up work.

===
you have successfully described the iterative and self-managing elements of scrum in the above introduction, and i appreciate that. i hope these other elements can also enter your description list when introducing scrum to colleagues.

regards, craig
craig@craiglarman.com
http://www.craiglarman.com
author of
-Practices for Scaling Lean & Agile Development Large, Multisite & Offshore Product Development with Large-Scale Scrum
-Scaling Lean & Agile Development Thinking and Organizational Tools for Large-Scale Scrum
-Agile and Iterative Development A Manager's Guide

3:31 pm on 05/01/11

STEVE DENNING
RETHINK

Dear Craig,

Thanks for these helpful clarifications.

Obviously the few sentences in my brief article about Scrum do not give, and do not purport to give, a complete description of the principles and practices of Scrum, let alone its spirit and philosophy, or its impact and its implications.

Nevertheless I hope that the article will alert people that there is something quite important happening here, and that it will encourage them to learn more elsewhere, for instance, in the wonderful books that you cite in your note.

I hope that your comments will also encourage them to do likewise.

Thanks again,
Steve

Log in to Reply Flag for abuse

DISPLAY

Called-Out Comments | **All comments**

Log in for notification options

Comments RSS