Teams that Finish Early Accelerate Faster: A Pattern Language for High Performing Scrum Teams

Jeff Sutherland
Scrum Inc.
jeff@scruminc.com

Neil Harrison
Utah Valley University
Neil.Harrison@uvu.edu

Joel Riddle
Scrum Inc.
joel@scruminc.com

Abstract

Recent surveys show that 42% of Agile projects are successful. While this is three times better than traditional projects, 49% of Agile projects are late or over budget and 9% are total failures [1]. There is a better way to help Agile teams to implement Scrum. At the 2013 Scrum PLoP Conference held in Tisvildeleje, Denmark thought leaders in the Agile community reviewed a set of Scrum Patterns that together generate a high performing Scrum team. During this editorial process it became apparent that a combination of nine Patterns in conjunction with the Scrum framework could help teams achieve Hyper-Productivity, more than a 400% increase in velocity over a team's initial velocity.

1. Introduction

Many years before the writing of the Agile Manifesto [2], Mike Beedle was influenced by the online description of Scrum [3]. He then implemented the process in his company, and led the effort to drive Scrum through the Pattern Languages of Programming Design conferences. The result was **Scrum: A Pattern Language for Hyperproductive Software Development**, a groundbreaking work that established a pattern foundation for Scrum, the most widely deployed Agile processes in the world [4].

Recent work by Jim Coplien shows that Scrum is deceptively simple while compressing a complex array of organizational patterns [5]. While Scrum incorporates at least 33 organizational patterns, it can be superficially explained in just 2 minutes.

One of Scrum's design goals was to encapsulate best practices from 40 years of software development into a process that was simple enough for the average developer to use with less than 2 days of startup time. Coplien's analysis [6] indicates that this goal was accomplished.

In recent years the Scrum Pattern Community has written a comprehensive set of patterns for Scrum [7] that allow teams to try proven approaches that have worked in many companies. While the Scrum Guide [8] provides the basic rules of Scrum, the patterns give teams the tools to solve problems when implementing Scrum in specific contexts.

2. Hyper-Productive Software Development

Only a small percentage of Scrum teams achieve Scrum's design goal of five to 10 times traditional project productivity with a corresponding increase in quality. Some of these Hyper-Productive teams include Mike Beedle's [3] and Jeff Sutherland's companies [9], as well as organizations in the U.S. [10], Russia [11], the Netherlands and India [12], and from Software Productivity Research data on agile teams [13].

Systematic, a CMMI Level 5 company in Denmark, has shown how to systematically produce a Hyper-Productive team by focusing on a high standard for "Done" at the end of a sprint and "Ready" at the beginning of a sprint [14]. They noticed that it was impossible to achieve Hyper-Productivity if they changed members of the Scrum team at the beginning of every project, showing that the pattern **Stable Teams [5]** is a requirement for high performance. Similar results were observed consistently for a style of Scrum called "Shock Therapy" in the U.S. and Europe [15].

The Systematic and Shock Therapy approaches to consistently generating a Hyper-Productive team have been too disciplined or too aggressive for most teams to implement. However, a venture capital group with over 30 companies suggested a better approach. OpenView Venture Partners decided to implement Scrum internally in 2006 for all departments in the company [16]. After running hundreds of sprints with carefully documented metrics, they discovered that **Teams that Finish Early Accelerate Faster** [17]. This insight provided a way for the average team to approach Hyperproductivity. If a stable team could accelerate

978-1-4799-2504-9/14 $31.00 © 2014 IEEE
DOI 10.1109/HICSS.2014.580

faster by finishing early, what other simple steps could be taken by any team to achieve Scrum: A Pattern Language for Hyperproductive Software Development [4]?

3. A Generative Pattern Language for Hyper-Productivity

A Pattern Language is an attempt to express deeper wisdom through a set of interconnected expressions arising from contextual knowledge. It moves beyond a list of processes, to seek activities or qualities that repeat across many of those processes, in an effort to find what works. It is an interconnected whole, that when applied coherently, creates "the quality that has no name" (QWAN) [18]. Combining multiple patterns creates a whole greater than the sum of the individual patterns.

The investors at OpenView Venture Partners were surprised when they discovered **Teams that Finish Early Accelerate Faster**. They observed that Scrum is not about velocity, it is about acceleration. An accelerating team will soon outperform a team with flat-lined velocity.

This pattern seemed counterintuitive to the investors, so the authors and others experimented with it in other companies and found that it consistently worked. The next question becomes how to get it to work well enough to generate a Hyper-Productive team. What set of *generative* patterns will feed off one another, generating unexpected side effects that keep teams accelerating?

Generative patterns work indirectly; they work on the underlying structure of a problem rather than attacking the problem directly. Good design patterns are similar: they encode the deep structure of a solution and its associated forces, rather than cataloging a solution [19].

We already knew from the Systematic data [14] that **Stable Teams** were necessary for hyperproductivity. We decided to systematically investigate every other major problem that blocks a team from finishing early.

4. The Patterns

A Scrum Pattern is a general reusable solution to a commonly occurring problem within the Scrum framework. The structure of Scrum is simple and designed to help Teams adapt to change as it occurs but Scrum doesn't solve every problem. As Scrum has been implemented and improved upon over time, a number of practices evolved to address common pitfalls.

Every year at the Scrum PLoP conference, new Patterns are proposed and go through a round robin editorial process by some of the most influential minds in the Scrum community. Eventually, if the Pattern is seen as having value, it is approved and added to the Pattern spreadsheet.

As more and more Patterns emerge, they can be used together. A subset of the Scrum patterns are the nine Patterns listed below, which form in essence a vocabulary of a *Pattern Language for Hyper-Productive Teams*.

The Patterns are:

1. Stable Teams
2. Yesterday's Weather
3. Swarming: One Piece Continuous Flow
4. Interrupt Pattern: Illigitimus Non Interruptus
5. Daily Clean Code
6. Emergency Procedure
7. Scrumming the Scrum
8. Happiness Metric
9. Teams that Finish Early Accelerate Faster

The first two patterns help the team get ready for a successful sprint. Patterns 3-6 help the team deal with the most common disruptive problems in a sprint. Patterns 7-8 will drive a team to the Hyper-Productive state by causing Pattern 9 to emerge as a side effect.

5. Patterns that Help Teams Get Ready

Stable Teams: *Keep teams stable and avoid shuffling people between teams. Stable teams tend to get to know their capacity, which makes it possible for the business to have some predictability.*

The Scrum framework is built around a team of three to nine members. Research at Harvard University and elsewhere has shown that the optimum size is five people [20, 21]. Small teams keep communication paths simple and allow for communication saturation, a key to hyper-productivity [22]. However, just having a small team doesn't mean it will be successful. If members are pulled off the team to work on other projects or are unable to participate regularly in rituals, the team's Velocity will suffer. To solve this problem, practitioners realized they needed small, stable teams.

At PatientKeeper [23] during 2005-2007 all teams were Hyper-Productive except an offshore waterfall team. Careful data collection during this period showed the onshore teams were 10 times as productive as the offshore team. A key feature was the stability of the onshore teams with almost no changes in team members during this period. We did

discover, however, that adding a new person to the team about every 6-12 months helped to bring in fresh ideas.

6. Patterns that Help Teams Finish the Sprint

Stable teams tend to reach a consistent Velocity, which helps the Team predict how many Points they can accomplish, each Sprint. That enables them to use the first pattern that helps prevent failed Sprints.

Yesterday's Weather: *In most cases, the number of Estimation Points completed in the last Sprint is the most reliable predictor of how many Estimation Points will be completed in the next Sprint.*

Yesterday's Weather allows teams to build a more accurate Sprint Backlog, limiting the possibility of the team ambitiously pulling in too many Estimation Points and endangering the Sprint. *Stable Teams know their capacity, which enables them to use Yesterday's Weather*.

Once stable teams have built a realistic Sprint Backlog using *Yesterday's Weather*, they start their Sprint. They then encounter numerous forces that can cause a Sprint to fail. The following four Patterns are designed to address the most common Sprint pitfalls.

Swarming: *Focus maximum team effort on one item in the Sprint Backlog to get it done as soon as possible. Whoever takes this item is Captain of the team. Everyone must help the Captain if they can and no one can interrupt the Captain. As soon as the Captain is Done, whoever takes responsibility for the next priority backlog item is the new Captain.*

When Teams struggle to finish Sprints, it is usually because they have too much work in progress and aren't swarming on high value Sprint Backlog items. Swarming helps teams move items to "Done" quickly, increasing Velocity. *Yesterday's Weather allows Swarming Teams to increase Velocity because the team is building a realistic Sprint Backlog.*

The next most common problem Scrum teams face is interrupts to work on the Sprint Backlog. Many requests come to the team which are not on the subset of the Product Backlog accepted into the Sprint. Research at Carnegie Mellon and 20 years of experience with Scrum teams has shown that teams that plan for interruptions do significantly better than teams that do not, even when they experience no interruptions [24].

Interrupt Pattern: *Allot time for interruptions and do not allow the time to be exceeded. Set up three simple rules that will cause the company to self-organize to avoid disrupting production*:

1. The team creates a buffer for unexpected items based on historical data. For example, 30% of the team's work on the average is caused by unplanned work coming into the sprint unexpectedly. If the team velocity averages 60 points, 20 points will be reserved for the interrupt buffer.
2. All requests must go through the Product Owner for triage. The Product Owner will give some items low priority if there is no perceived value relative to the business plan. Many other items will be pushed to subsequent Sprints even if they have immediate value. A few items are critical and must be done in the current Sprint, so the Product Owner puts them into the interrupt buffer.
3. If the buffer starts to overflow, i.e. the Product Owner puts one point more than the 20 points allocated to the buffer into the Sprint, the team must automatically abort, the Sprint must be re-planned, and management is notified that delivery dates will slip.

The Interrupt Pattern, like Swarming, allows teams to finish their Sprints because they have developed a process to deal with found work. Examples of how to use these patterns to solve common problems were found in many of the OpenView Venture Partners portfolio companies [16].

Balihoo, a company that automates local marketing campaigns for companies such as Wendy's, Ace Hardware, and New Balance, failed to deliver half of its planned stories for 18 two-week sprints in a row. The management was not happy with their Scrum team.

The first problem addressed was that almost all stories were open on their Scrum Board every day. Excessive "work in progress" delays testing and makes it extremely difficult to get things done in a Sprint. We fixed that by Swarming, which caused the whole team to focus on completing a least one story on the board every day. At the same time we implemented the Interrupt pattern. All of the next 18 Sprints, were successful, none were aborted, and velocity more than tripled. *The Interrupt pattern generates a side effect that causes the entire company to self-organize to avoid sprint aborts.* This means the buffer is never completely used up and teams tend

to finish early and pull forward from the next Sprint's backlog. This increases yesterday's weather and the team accelerates.

Finishing at least one story every day allowed the team to focus on the second value in the Agile Manifesto – working software with no bugs. This minimizes the amount of undone work at the end of the sprint and maximizes velocity. All great Scrum teams implement the *Daily Clean Code* pattern.

Daily Clean Code: *Fix all bugs in less than a day. Aim to have a completely clean base of code at the end of every day.*

If a Team isn't creating daily clean code, a lot of time will be wasted going back to fix bugs. Errors can be limited by building quality control into the development process so that issues are discovered and corrected at the point of origin. Research in Silicon Valley at Palm, Inc. in 2006, showed that a bug that is not fixed the same day it is created can take as much as 24 times longer to correct three weeks later.

Despite their best efforts, even a great team may find themselves behind on implementing the Sprint Backlog with no clear way to complete the Sprint successfully. In this case, by mid-Sprint they should execute the *Scrum Emergency Procedure*.

Emergency Procedure: *When high on the burndown try a technique used routinely by pilots. When bad things happen, execute the emergency procedure designed specifically for the problem. Do not delay execution while trying to figure out what is wrong or what to do. In a fighter aircraft you could be dead in less time than it takes to figure out what is going on. It is the responsibility of the Scrum Master to make sure the team executes the Scrum Emergency Procedure, preferably by mid-sprint, when things are going off track.*

Emergency Procedure Steps: (do only as much as necessary)

1. Change the way the work is done. Do something different.
2. Get help, usually by offloading backlog to someone else.
3. Reduce scope
4. Abort the sprint and replan. Inform management how release dates will be affected.

7. Getting Hyper-productive

Stable Teams and *Yesterday's Weather* set the team up for success by helping it get in a ready state. *Swarming, the Interrupt Pattern, Daily Clean Code,* and the *Emergency Procedure* help the Team deal with Impediments as they arise during the Sprint. The next three Patterns take advantage of the previous Patterns and allow the team to attain a Hyper-Productive state.

Scrumming the Scrum: *Identify the single most important impediment from the previous Sprint during the Sprint Retrospective and remove it before the end of the next sprint. To remove the top impediment, put it in the Sprint Backlog as a user story with acceptance tests that will determine when it is Done. Then evaluate the state of the story in the Sprint Review like any other story.*

If the team is able to capitalize on *Scrumming the Scrum* they should create at least one process improvement per sprint. The pattern calls this process improvement the *Kaizen*. This contributes to increasing Velocity. If the team is using *Yesterday's Weather*, than they have a good chance to finish their sprint early because they will have one less impediment dragging down their Velocity. (The Kaizen may not be a direct process improvement. It may deal with strong personalities, management impeding the Sprint, or a variety of sticky human issues. These impediments should be treated like process improvements and should be resolved as quickly as possible.)

Happiness Metric: *Happiness is one of the best metrics because it is a predictive indicator. When people think about how happy they are they are really projecting out into the future about how they feel. If they feel the company is in trouble or doing the wrong thing, they will be unhappy. Or if there is a major roadblock or frustrating system they have to deal with, they will be unhappy.*

A powerful way to take the pulse of the Team is by finding out how happy they are. The Scrum Master asks just 2 questions:

- How happy are you with the company?
- How happy are you with your role?

Team Members are asked to rate their feelings on these questions on a scale from one to five. These numbers are kept in a spreadsheet and tracked over time. If the average changes significantly it's important to talk and see how Team happiness can be improved. By monitoring the team's happiness, the Scrum Master can anticipate drops in Velocity and make adjustments.

Teams That Finish Early, Accelerate Faster: *Teams often take too much work into a Sprint and cannot finish it. Failure prevents the Team from improving. Therefore, take less work into a Sprint (see Yesterday's Weather for guidance Then implement the four Patterns that reduce Impediments within the Sprint, which will systematically deal with any interruptions and help you finish early. On early completion pull work from the Product Backlog which will increase the baseline of Yesterday's Weather.*

8. Implementation Example

A new Scrum team was started up in 2010 to run an entire company with one week Sprints. Backlog was pulled into Sprints based on the average Velocity for the previous three Sprints. An interrupt buffer was used to handle unplanned work. The team minimized work in progress focusing on daily clean completion of stories. The emergency procedure was used to handle difficult problems.

The team used the Happiness Metric as a way to identify and prioritize process improvements. On a scale of 1-5 they asked (1) how they feel about their role in the company and (2) how they feel about the company. They then shared what would make them feel better. The team used planning poker to estimate the value of things that would make team members feel better. The team estimated the value (as opposed to effort) of backlog items as well. The entire product backlog was estimated at 50 points of value in an early Sprint.

"Better user stories" was the top priority improvement for the team. Removing this impediment was estimated at over 60 points of value. The Chief Product Owner wondered if removing that impediment might double velocity, as the impediment value was higher than the entire product backlog value for the sprint.

"Improve User Stories" was put into the Product Backlog and pulled into the next sprint with a definition of Done. That definition of Done included acceptance tests with metrics that were calculated at the next Sprint Review. They included:

1. How many stories got into the sprint that did not meet the INVEST criteria (immediately actionable, negotiable, valuable, estimable, sized to fit, and testable)?

2. How many times did members of the Team have to go back to the product owner to clarify a story during a sprint?

3. How many times did dependencies force a story into a hold state during a Sprint?

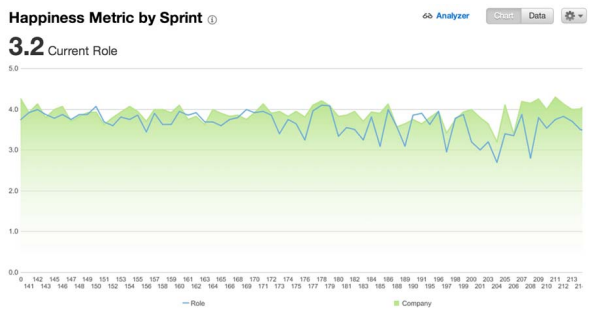
4. How many stories had a process efficiency of over 50%? (process efficiency = actual work time/calendar time)

5. How many stories were not clear to the team members? Measure by number of team members that complained about a story.

6. How many stories implied technical implementation rather than clarifying desired user experience?

7. For how many stories did team members understand the linkage between the story, the theme that produced the story, the epic that generated the theme, and the business need that generated the epic? This was measured by number of team members complaining that they did not understand why they were doing a story.

Resulting Context: While improving the quality of user stories is never ending, the sprint review demonstrated significant improvement on this backlog item as measured by the acceptance tests. Significant improvement resulted in an increase in velocity sprint to sprint for three sprints. After velocity had tripled this impediment fell off the top of the impediment list and another impediment took its place.

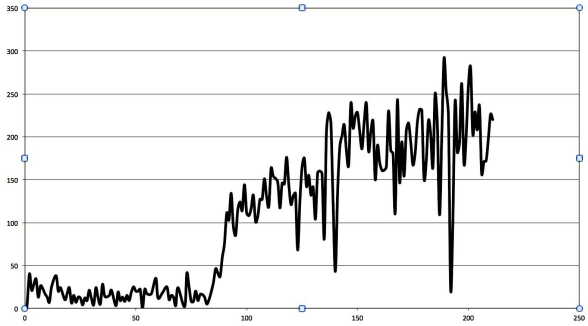


The graph above is team happiness data for weekly sprints 140-212 where the solid line is happiness about the individual's work and the shaded area is happiness about the company. While happiness had some normal variation, work on the Kaizen kept it hovering around 4.

The graph below shows the raw velocity of the team. In Sprint 86 the team's size was doubled and velocity rose to 37 during Sprint 88. In Sprint 89, "Improve User Stories" was put in the backlog of each sprint for three sprints. By Sprint 91 velocity was 111, up 300% from Sprint 88.

Velocity continued to increase for the next two years using the Scrumming the Scrum pattern and by Sprint 211 output was up 1200% while the team had tripled in size. This is the first documented, sustainable, hyper-productive company (400% improvement in velocity), as the data include all

work for the entire company. The low points on the velocity graph are when individuals or the whole company were on vacation.



Velocity in Points. Source: Scrum Inc. Company Data 2010-2013, weekly sprints 1-214

9. Conclusions

By implementing and executing all nine Patterns, teams dramatically increase their ability to finish the Sprint early. This allows them to pull more Product Backlog Items into the Sprint from the Product Backlog. This will increase Velocity and establish a higher baseline for *Yesterday's Weather*, setting the team-up for the next Sprint. Teams that finish early also tend to have a higher Happiness Metric because they feel confident about their ability to complete Sprints. This initiates a virtuous cycle of continuous improvement eventually leading to Hyper-Productivity.

The generative nature of these patterns is not obvious to those who have not tried them. Unanticipated side effects cause unexpected positive results. Therefore, it is recommended that all teams try these patterns, particularly in combination, to see if they help improve performance, quality, and happiness of the team.

10. References

[1] K. Schwaber and J. V. Sutherland, *Software in 30 days : how agile managers beat the odds, delight their customers, and leave competitors in the dust*. Hoboken, N.J.: John Wiley & Sons, Inc., 2012.

[2] M. Fowler and J. Highsmith, "The Agile Manifesto," *Dr. Dobbs,* July 13 2001.

[3] M. Beedle. (2010, June 15). *Mike Beedle on the Early History of Scrum*. Available: http://scrum.jeffsutherland.com/2010/08/mike-beedle-on-early-history-of-scrum.html

[4] M. Beedle, M. Devos, Y. Sharon, K. Schwaber, and J. Sutherland, "Scrum: A Pattern Language for Hyperproductive Software Development," in *Pattern Languages of Program Design*. vol. 4, N. Harrison, Ed., ed Boston: Addison-Wesley, 1999, pp. 637-651.

[5] J. O. Coplien and N. Harrison, *Organizational patterns of agile software development*. Upper Saddle River, NJ: Pearson Prentice Hall, 2005.

[6] G. Bjornvig, J. Coplien, and J. Ostergaard, *Scrum Tuning Using Organizational Patterns*: Scrum Foundation, 2010.

[7] ScrumPloP. (2013). *Scrum Pattern Community*. Available: http://scrumplop.org

[8] K. Schwaber and J. Sutherland, "The Scrum Guide: The Definitive Guide to Scrum, The Rules of the Game," in *Software in 30 Days*, ed: John Wiley & Sons, 2011.

[9] J. Sutherland and K. Schwaber, *The Scrum Papers: Nuts, Bolts, and Origins of an Agile Method*. Boston: Scrum, Inc., 2007.

[10] M. Cohn, *User Stories Applied : For Agile Software Development*: Addison-Wesley, 2004.

[11] J. Sutherland, A. Viktorov, J. Blount, and N. Puntikov, "Distributed Scrum: Agile Project Management with Outsourced Development Teams," presented at the HICSS'40, Hawaii International Conference on Software Systems, Big Island, Hawaii, 2007.

[12] J. Sutherland, G. Schoonheim, and M. Rijk, "Fully Distributed Scrum: The Secret Sauce for Hyperproductive Offshored Development Teams," in *Agile 2008*, Toronto, 2008.

[13] C. Jones, "Development Practices for Small Software Applications," Software Productivity Research 2007.

[14] C. R. Jakobsen and J. Sutherland, "Scrum and CMMI Going from Good to Great," in *Agile Conference, 2009. AGILE '09.*, 2009, pp. 333-337.

[15] J. Sutherland, S. Downey, and B. Granvik, "Shock Therapy: A Bootstrap for Hyper-Productive Scrum," in *Agile Conference, 2009. AGILE '09.*, 2009, pp. 69-73.

[16] J. Sutherland and I. Altman, "Take No Prisoners: How a Venture Capital Group Does Scrum," in *Agile 2009*, Chicago, 2009.

[17] J. Sutherland. (2013). *Teams that Finish Early Accelerate Faster*. Available: https://sites.google.com/a/scrumplop.org/published-patterns/retrospective-pattern-language/teams-that-finish-early-accelerate-faster

[18] P. L. o. G. Process. (2013). *What is a Pattern Language?* Available: http://grouppatternlanguage.org/What_is_a_Pattern_Language

[19] J. Coplien. (1995). *Generative Pattern*. Available: http://c2.com/cgi/wiki?GenerativePattern

[20] J. R. Hackman and D. Coutu. (2009) Why Teams Don't Work. *Harvard Business Review*.

[21] J. R. Hackman, *Leading Teams: Setting the Stage for Great Performances*. Cambridge: Harvard Business Review Press, 2002.

[22] J. O. Coplien, "Borland Software Craftsmanship: A New Look at Process, Quality and Productivity," in *5th Annual Borland International Conference*, Orlando, FL, 1994.

[23] J. Sutherland, "Future of Scrum: Parallel Pipelining of Sprints in Complex Projects," presented at the AGILE 2005 Conference, Denver, CO, 2005.

[24] B. Sullivan and H. Thompson, "Gray Matter: Brain, Interrupted," in *New York Times*, ed. New York City: New York Times Company, 2013.