

Scrum Metrics for Hyperproductive Teams: How They Fly like Fighter Aircraft

Abstract

Scrum teams use lightweight metrics like story points, the burndown chart, and team velocity. The inventor of Scrum was a fighter pilot and used the Scrum burndown chart to help teams land a sprint properly. Recent work with hyperproductive teams shows they are like modern jet fighters in multiple ways. They have two engines that produce velocity--alignment of the team and team spirit. A hyperproductive team uses careful measurement of aspects of performance and prioritization to make slight adjustments in flight. Just as modern jet fighters are inherently unstable without computers to fine tune flight parameters, hyperproductive teams require daily adjustment based on key metrics.

Careful attention to the metrics described--velocity, work capacity, focus factor, percentage of found work, percentage of adopted work, original commitment, final commitment, commitment accuracy, estimate accuracy, and target value contribution increase can develop and sustain hyperproductive teams.

1. Background

The average Scrum team delivered a 35% improvement in velocity at Yahoo [1] where teams properly coached delivered 300-400% improvements. The best Scrum Master at MySpace peaked at 267% of initial velocity after 12 weeks and averaged 168% increase in velocity over 12 Sprints. Most teams were less successful.

We define Hyper-Productivity here at 400% higher velocity than average waterfall team velocity with correspondingly higher quality. The best Scrum teams in the world average 750% gains over the velocity of waterfall teams with much higher quality, customer satisfaction, and developer experience. We have seen this in the U.S. [2], Russia [3], the Netherlands and India [4], and from Software Productivity Research data on agile teams [5]. The problem addressed in this paper is that over 90% of Scrum teams never deliver this capability.

Agile teams have trouble measuring performance. Over 50% of teams do not know their velocity of production and have difficulty in finding ways to improve and measure this rate. Even when teams

know their velocity, at the management level it is difficult to compare the performance of two teams.

Velocity on Agile teams is typically measured in story points. Teams pick a small reference story and assign it an arbitrary number of points [6]. All other stories are estimated relative to the reference story using a wide-band delphi estimation technique commonly known as "planning poker" on Agile teams. Planning poker provides faster and more accurate estimates with less variance than hourly estimates but has the disadvantage that it is not usually comparable across teams. While function points are the preferred metric for productivity research they require more training, expertise, and time than is usually available to Agile teams.[7]

The lack of adequate attention to metrics can prevent teams from systematically improving and reaching a hyperproductive state. This state is defined as at least 400% better than the average waterfall team. We have many hyperproductive teams today running at 4-8 times the performance of the average waterfall team [3, 8-10].

2. Scrum is an Ecosystem

Experienced agile coaches recognize that Scrum is based on complex adaptive systems theory. It is not a methodology, process, or procedure. It is a framework based on enforcement of simple constraints that will cause an average team to self-organize into a hyper-productive state [11].

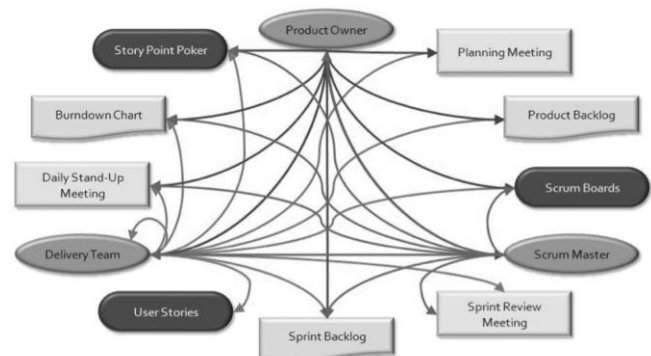


Figure 1. Scrum is an ecosystem.

Any system will settle into the lowest possible energy state. Consider the water in a toilet. It is without motion and flat. When you flush the toilet you introduce energy into the system and enforce constraints which cause the water to swirl into the same motion every time. As soon as the energy input stops, the water returns to a flat and motionless state.

The difference between the highest and lowest performing software development teams is 1:2000 [12]. This is more than two orders of magnitude greater than the difference between the best and worst developer on a project [13]. The average software development team is in a placid state where velocity is slow, quality is low, customers are unhappy, and management is upset. We want to introduce energy into the team and enforce constraints that systematically produce high velocity, high quality, happy managers, and ecstatic customers.

The Scrum meetings are designed to raise the communication saturation level of a team in order to align their focus and facilitate team spirit. This introduces an energy flow into the system which is constrained by the ordering of the product backlog, the required ready state of user stories, a strong definition of done, and continuous process improvement through removal of impediments. Velocity of the team, quality of the software, satisfaction of the users, and revenue for the company will always increase several hundred percent if communication saturation goes up and Scrum constraints are properly enforced. Waste will be flushed from the system and the team will go from strength to strength.

When implementing Scrum, it is therefore essential to understand Scrum as an ecosystem of interdependent parts. The balance of each part with the other needs to be inspected daily. A simple set of metrics provides a dashboard similar to an aircraft cockpit. Watching altitude, direction, speed, and rate of descent can keep you on track even in heavy weather.

3. Current state

People are often measuring hours of work accomplished or tasks completed without being able to clearly demonstrate forward progress on the product owner's roadmap or demonstrate process improvement that increases value contribution. Management cannot compare performance of Agile teams straightforwardly. Productivity and quality are less than 25% of what they could be with properly functioning teams.

There are, however, a few teams that heavy broken through the barrier of mediocre performance. As an example, here we have data on five teams from

MySpace in California. Teams at MySpace were implementing a web framework and tools to support hundreds of millions of users building their personal web pages.

3.1. Establishing baseline velocity

The baseline velocity (100%) is established for a team during the first Sprint. The Product Owner presents the prioritized Product Backlog in the Sprint Planning meeting. This is estimated using Planning Poker and story points [6]. The team selects what can be accomplished during the Sprint and the Product Owner determines exactly what is "Done" at the end of the Sprint. The number of story points completed is the baseline velocity.

Velocity is defined as:

$$V = \sum \text{of original estimates of all completed work}$$

At MySpace, the baseline velocity is often significantly higher than their previous chaotic implementation of waterfall, so the baseline is conservative.

3.2. MySpace Team Data

Data on five teams at MySpace is summarized in Figure 2. The solid curve in the middle of the graph is average velocity for all teams for each Sprint. The upper and lower curves show the maximum and minimum achievement from the data.

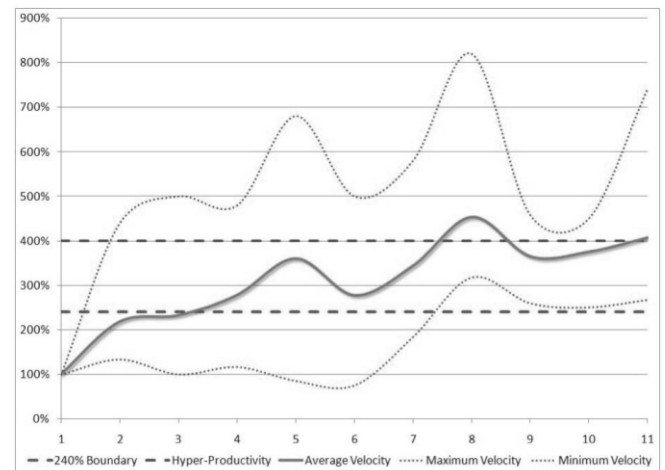


Figure 2. Velocity of MySpace Teams by Sprint

The lower dotted line is 240% percent of baseline velocity and the goal at MySpace was to achieve this in three one-week Sprints. Teams that achieve this typically go over 400% (upper dotted line) into a

hyper-productive state in later Sprints. The low data points were from the only team in this data set where the MySpace Agile Coach did not assume the ScrumMaster role. The existing Scrum Master failed to enforce constraints.

These teams were all monitored by a carefully selected set of metrics that was used to analysis performance in real time. Flying these teams into the hyperproductive state required careful balance of the altitude, speed, direction, and rate of descent on the burndown chart at all times. Failure to do this will cause a hyperproductive team to spiral out of control. This results in velocity that descends to baseline level.

4. A simple set of metrics can help create and maintain a hyperproductive state for all teams

Good metrics help the team to measure their own performance. They help management compare the performance of multiple teams with apples to apples metrics. They lay down a framework for consistent data collection so that measured hyperproductivity is clear.

Good metrics also give the ScrumMaster a clear framework as a basis to advise the team. They measure the impact of modification of the environment (removal of impediments) on team performance.

When we say team value contribution is up 200%, we want it clear and demonstrable what we mean. TVC+ (targeted value contribution increase) allows us to compare the increase in horsepower of the team with the increase in revenue generated by the Product Owner's backlog. Team measurements show how well the team satisfies the product owner requirements.

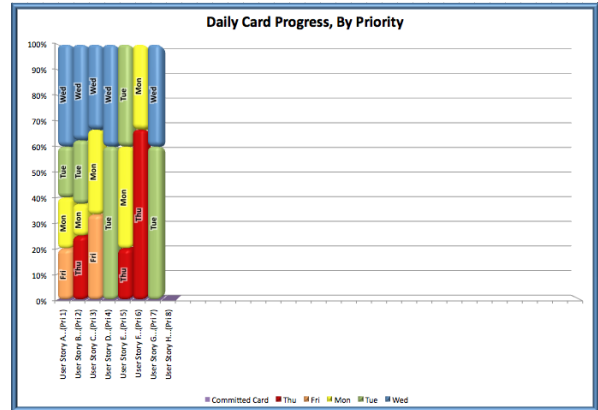


Figure 4. Daily Card Progress

Daily card progress allows us to see the rainbow of time. Red is first day of sprint and blue is last day. Why did the team start priority 2, 5, and 6 before priority 1 or 3? It enables the ScrumMaster to facilitate a discussion on priorities.

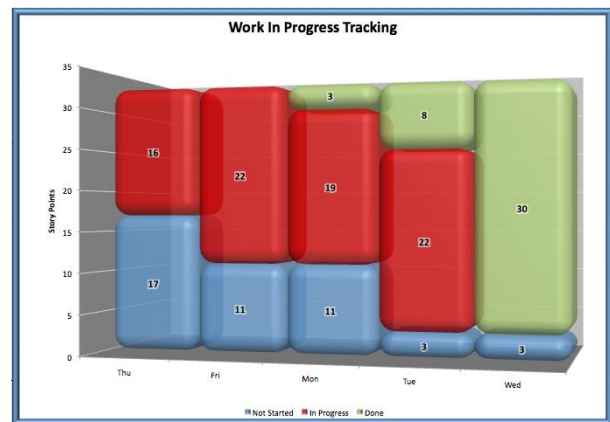


Figure 5. Work in Progress

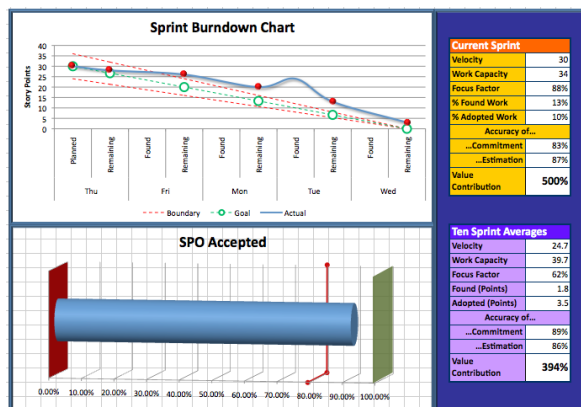


Figure 3. Sprint Burndown Chart

Blue is not started. Red in progress. Green is done.

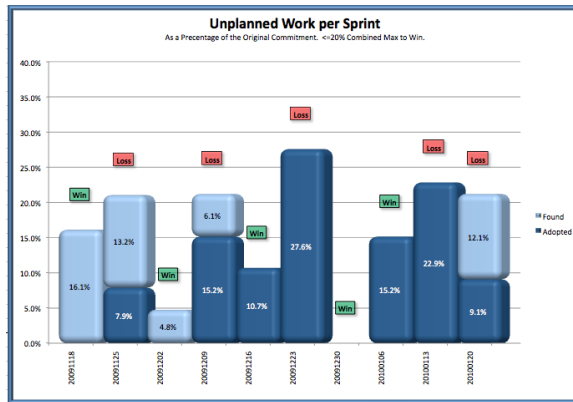


Figure 6. Unplanned Work

Careful tracking of unplanned work is essential to detecting and removing waste from a hyperproductive team.

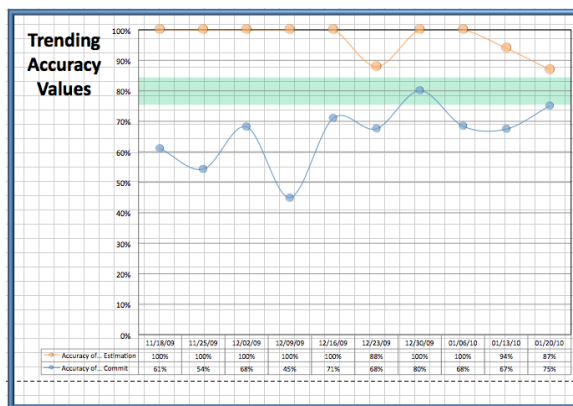


Figure 7. Trending Accuracy Values

5. Conclusions

Hyperproductive Scrum teams need a simple set of metrics to provide subtle control to the team. Without these metrics, performance of the team can be unstable and loss of control will result in lowered velocity. Super-performing teams typically abandoned hours as a means of tracking progress as it introduces waste into the system, lowers velocity, and reduces predictability. The simple set of metrics introduced here are easy to implement and have a powerful effect on performance of hyperproductive Scrum teams.

6. References

[1] G. Benefield, "Rolling Out Agile at a Large Enterprise," in *HICSS'41, Hawaii International*

Conference on Software Systems, Big Island, Hawaii, 2008.

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

[3] 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: IEEE, 2007.

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

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

[6] M. Cohn, *Agile Estimation and Planning*: Addison-Wesley, 2005.

[7] J. Sutherland, *Jeff Sutherland's Scrum Handbook: Scrum Training Insitute*, 2010.

[8] J. Sutherland, G. Schoonheim, and M. Rijk, "Fully Distributed Scrum: Replicating Local Productivity and Quality with Offshore Teams," in *42nd Hawaii International Conference on Software Systems*, Big Island, Hawaii, 2009.

[9] J. Sutherland, G. Schoonheim, N. Kumar, V. Pandey, and S. Vishal, "Fully Distributed Scrum: Linear Scalability of Production Between San Francisco and India," in *Agile 2009*, Chicago, 2009.

[10] C. Jakobsen and J. Sutherland, "Scrum and CMMI – Going from Good to Great: are you ready-ready to be done-done?," in *Agile 2009*, Chicago, 2009.

[11] 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. Boston: Addison-Wesley, 1999, pp. 637-651.

[12] L. Putnam and W. Myers, *Industrial Strength Software: Effective Management Using Measurement*: IEEE, 1997.

[13] J. Spolsky, "Hitting the High Notes," in *Joel on Software* New York: Fog Creek Software, 2005.