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

Scott Downey
Owner, Rapid Scrum LLC
Scott@RapidScrum.com

Jeff Sutherland, Ph.D.
CEO, Scrum Inc.
Jeff@ScrumInc.com

Abstract

Scrum Teams use lightweight tools like Story Points, the Burndown chart, and Team Velocity. While essential, these tools alone provide insufficient information to maintain a high energy state that yields Hyperproductivity. More data is required, but data collection itself can slow Teams. This effect must be avoided when productivity is the primary marker of success.

Here we describe nine metrics that can develop and sustain Hyperproductive Teams—Velocity, Work Capacity, Focus Factor, Percentage of Adopted Work, Percentage of Found Work, Accuracy of Estimation, Accuracy of Forecast, Targeted Value Increase, Success at Scale, and the Win/Loss Record of the Team. The unique contribution of this paper is to demonstrate how a light touch and lightweight strategy can be used to compare Teams with different Story Point reference scales.

1. Background

A fighter aircraft is inherently unstable and must constantly correct to stay within the flight envelope—those parameters where the plane flies properly. Recent work with Hyperproductive Teams shows they are much like modern jet fighters. They have two engines that produce velocity – alignment of the Team, and Team spirit. Just like the cockpit gauges of a fighter aircraft, Scrum Teams need a set of reliable, lightweight metrics so that Team Performance can be easily monitored and quickly corrected if problems arise. These metrics must be collected with a minimum of disruption to the Team, as we know that the act of measurement alone may serve to slow Teams down. Failure to collect and monitor these metrics put you in danger of crashing from Hyperproductivity back to a state that is no more productive than waterfall Teams.

Historically, most Scrum teams have done a poor job of collecting quality data over time on team performance. The first Scrum team was carefully measured using tooling provided by consultants from Software Productivity Research. Subsequent Scrum teams deployed at dozens of Scrum companies led by Sutherland have captured even better data and these data have been compared to ongoing research by productivity expert, Capers Jones, the founder of SPR. As a result we have some of the best data in the world across many companies that precisely define the expected performance of Scrum teams under varied conditions.

For example, the Scrum teams initiated at Yahoo by Scrum Foundation founders Sutherland, Deemer, and Benefield delivered an average 35% improvement in velocity at Yahoo [1] whereas Teams properly coached on how to achieve performance delivered 300-400% increases. As Agile Coach at MySpace, Downey had teams that peaked at 1680% of initial velocity after 20 weeks and averaged 450% increase in velocity over 10 Sprints. The highest performing Team ever recorded was a Borland Team audited by Bell Labs. They were 50 times faster than waterfall Team industry average [2]. Clearly, large performance gains are possible.

Currently, 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 worked directly with projects in the U.S. [3], Russia [4], the Netherlands and India [5], and compared results with Software Productivity Research data on agile Teams [5]. Capers Jones data has been almost exactly equivalent to ours giving us significant confidence in our findings. The problem addressed in this paper is that over 90% of Scrum Teams never deliver the capability seen in most of our teams and Capers Jones teams [6].

Agile Teams have trouble measuring performance. Global surveys by the authors show 50% of Teams do not know their velocity of production and have difficulty finding ways to improve and measure this rate. Even when Teams know their velocity, management cannot compare the performance of two Teams with current metrics.

Velocity on Agile Teams is typically measured in Story Points. Teams pick a small reference story and assign it an arbitrary number of points. All other stories are estimated relative to the reference story using the wide-band Delphi estimation technique

1530-1605/12 $26.00 © 2012 IEEE
DOI 10.1109/HICSS.2013.471

commonly known as "planning poker" [7] 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 [8]

The lack of adequate attention to metrics can prevent Teams from systematically improving and reaching a Hyperproductive state, at least 400% better than the average waterfall Team. Today we have many documented Hyperproductive Teams running even faster than this [4, 9-11].

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 Hyperproductive state [12]. Simple rules can drive self-organization at all levels in an organization. [13]

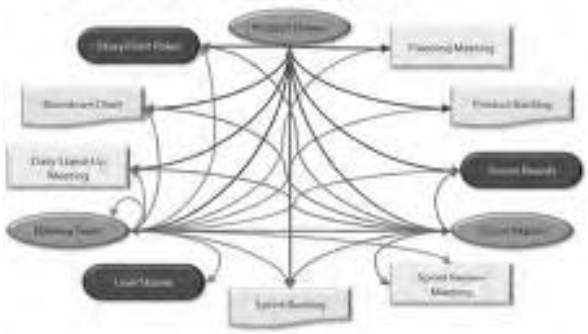

Figure 1. Scrum is an ecosystem.

Any system will settle into the lowest possible energy state. Consider a spinning top. In its natural, unaffected state, it is motionless and lies tilted on its side. When you introduce energy into the system by spinning it, it becomes upright and stable for a time. Then friction and gravity overcome the spinning motion and it returns to a motionless, inert state.

The difference between the highest and lowest performing software development Teams is 1:2000 [14]. This is more than two orders of magnitude greater than the difference between the best and worst developer on a project [15]. The average software development Team is in a placid state (like the top in its unaffected 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.

Scrum meetings are designed to raise the quality of communication within the Team, 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 coordination of the parts requires daily inspection in order to maintain a high energy state. 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 of Agile teams

People often measure 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 have broken through the barrier of mediocre performance. As an example, we have data on five Teams from MySpace in California. Teams at MySpace worked on a variety of projects, from SEO and framework standards to internal tools and user features that manage profiles and accounts for 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 [7]. 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 sum of the original

estimates for the approved work is the baseline Velocity.

Velocity is defined as:
V = ∑ of original estimates of all accepted work

At MySpace, the Baseline Velocity is often perceived by the Team as being too low. This derives from the fact that they have been allowed and encouraged to expect reward for motion alone, not exclusively for completion. In Scrum, we do not recognize Value Creation until the work is accepted by the Product Owner as Done. So Team Members who spend time on an initiative which is not completed by the end of the Sprint initially feel slighted when no points are accepted for their work. Scrum Masters who strive to reward motion in the absence of completion are doing a disservice to their Team, as this delta serves to highlight the scale of suboptimization that will be overcome with successful application of the Scrum framework.

3.2. Daily Stand-Up Modifications

In order to collect data indicating progress during the Sprint and get new Teams operational more quickly, a few modifications to the standard Daily Stand-Up format were necessary.

The first is to structure the meeting around the Sprint Backlog. Most Teams use a standard format wherein each individual answers the three Scrum questions:

1. *What did you do yesterday?*
2. *What are you going to do today?*
3. *What, if anything, is blocking you?*

We shift the focus of the meeting from the individuals to the Sprint Backlog. Starting with the highest priority Sprint Backlog Item (SBI) that is not yet completed in each Daily Stand-Up, the entire Team discusses their collective contribution toward completing that SBI. They then estimate their collective contribution's complexity in Story Points as if the previous day's contribution had been presented during the Sprint Planning meeting as the entire goal of the body of work. The Team then collectively plans the fastest and most effective way to share the work in order to move that SBI into the Done column as quickly as possible. Finally, we discuss anything that blocks the work or has the potential to slow it down for any reason. So the restructured Daily Stand-Up questions become:

1. What did WE achieve yesterday on Priority 1?
2. What was OUR contribution on Priority 1 worth in Story Points?
3. What is OUR plan for completing Priority 1 today?
4. What, if anything, is blocking US or has the potential to slow US down today?

These questions are then repeated for each lower Priority remaining in the Sprint Backlog until either all SBIs have been discussed or the 15 minute allotted time has elapsed, whichever comes first.

These modifications serve several purposes. Shifting the focus from the individual to backlog priorities helps people to function more as a Team. It encourages consideration of how to effectively subdivide the work for quicker completion, overcoming the technical silos that specialists tend to prefer.

We also find better quality updates and more attentive participation from all Team Members as a result of question 2. Because each Team Member now has a need to understand the complexity that has been resolved in order to vote on it, updates on the order of *"Yesterday, I worked on SBI 1. Today, I will keep working on SBI 1. No impediments."* are no longer tolerated by the Team. They become a self-policing group, both demanding quality updates and full attention from all Team Members to keep the meeting efficient.

Through the daily repetition of Story Point estimation, we also find that both the quality and the speed of estimation in Story Points improves more quickly using this method than with Teams who only experience Story Points during their Sprint Planning Meetings and use alternate metrics during the Sprint. This is true for all Teams, but is especially true for those that may have previously been unfamiliar with Story Points.

The more detailed discussions of achievements aide in cross-training the entire group more quickly, as they will hear and be asked to estimate the work of Teammates with specialties that may differ significantly from their own. This also quickens the Team's learning so that they can move through the *Forming, Norming, Storming, Performing* [16] phases rapidly.

Finally, and critically, it overcomes fractional thinking. For example, it is typical for an engineer who is working on a task estimated as 5 Story Points to report 1 point per day for 5 days if s/he feels that the work is progressing smoothly. This creates a false sense of uniformity in the rate of complexity resolution and often masks estimation inaccuracies that could be discussed in Retrospectives to help the entire Team become better at the initial estimates.

3.3 The INVEST criteria for SBIs

The common model for handling work in Scrum is to have a Product Backlog (PBL) populated by User Stories and a Sprint Backlog (SBL) populated by some derivative from User Stories, typically referred to as Tasks or simply Sprint Backlog Items (SBIs). There is a common expectation that Product Backlog Items (PBIs) are estimated in Story Points and may vary widely in scale, while SBIs are designed to be a uniform series of 2-hour blocks of time. The commonly accepted justifications for this behavior are threefold:

1. To control batch size of work with a goal of providing granular visibility and a consistent sense of progress to those outside of the Team
2. To push the Team to spend more time investigating the work with a goal of creating a higher degree of certainty that their Sprint Forecast is completely accurate.

While both of these goals are good ones to pursue, we find that the suggested approach to achieving them is too heavy-handed.

Though a consistent batch size is known to help speed Team performance [17] our model's goal is to spend a minimum of the *Team's* time and effort on digesting work and instead maximize the time and energy available for achieving it. Using a slight modification of the INVEST mnemonic [15], we ask the Teams to accept the *largest* piece of work that they believe they can achieve in the coming Sprint with ~80%+ confidence, that is:

- **I**mmediately Actionable
- **N**egotiable
- **V**aluable
- **E**stimable
- **S**ized to Fit (Max of ~50% of Velocity)
- **T**estable

We are not concerned with uniformity of scale among the SBL, nor on the formatting of the items it contains. We seek uniformity instead on the PBL. With a goal of minimizing the Team's time and energy on everything that doesn't *directly* create Value, it is a natural shift to ask the Product Owners to include this extra rigor in their PBI creation.

We then extract our external visibility by first normalizing the units of measure between the PBL and SBL into Story Points and, second, communicating to all stakeholders outside the Team exclusively in the unit of Sprints or percentages of Sprints. This avoids the very dangerous situations that arise when Story Points are used as the unit of measure in external communications to stakeholders who do not understand them, as Story Points are exclusively meaningful to their Team of origin.

Further, the reformatted Daily Stand-Up Meeting as described in section 3.2, above, provides us with a consistent sense of progress for each SBI. We can then clean up our Information Radiator [18] by reducing the number of SBIs represented thereon while keeping them in the language easily understood by external stakeholders. The Information Radiators are then returned to their original intent, which is to clearly and quickly communicate status to stakeholders not involved in the daily lives of the Team. So the sense of what is happening comes from the SBIs represented on the Information Radiator, while the sense of progress comes from the Burndown Chart in concert with the metrics described below.

3.4 MySpace Team Data

Data on five Teams at MySpace is summarized in Figure 2.

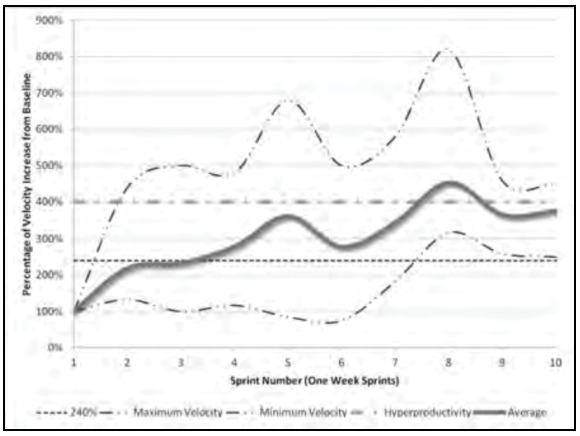

Figure 2. Velocity of MySpace Teams

The solid curve in the middle of the graph is average Velocity for all five Teams for each Sprint. The upper and lower curves show the maximum and minimum achievement from the data.

The lower dotted line marks 240% percent of baseline Velocity. This threshold was used to recognize that Teams had achieved a level of proficiency with the Scrum Framework so that the Agile Coach could begin gradually returning control from the Shock Therapy [16] model to the Team Members. This threshold was usually crossed in 3-5 one-week Sprints. Teams that achieve this typically went on to surpass 400% (upper dotted line) into a Hyperproductive 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 Scrum Master role. The permanent Scrum Master failed to enforce constraints.

These Teams were all monitored by the set of metrics described below, which were used to analyze 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 can cause a Hyperproductive Team to spiral out of control. The result is Velocity that descends to baseline level.

4. Good Metrics Promote Improvements

Good metrics help the Team measure their own performance and make changes based on facts, not just on feelings or guesses. Since, unlike Story Points, these metrics are meaningful outside of their Team of origin, they help management compare the performance of multiple Teams with apples-to-apples data. They also lay down a consistent framework for data collection so that measured hyperproductivity is clear and broadly understood.

Good metrics give the Scrum Master a solid foundation for the advice that s/he offers the Team. They make clear the impact of any modification, from the introduction of new tools and technologies to changes in process or even Team composition. This can help Scrum Masters who may need to justify requests for additional resources make the case for how those resources would be applied and the impact that the company can expect if the requested investment is approved.

The formulas for these ten essential metrics are as follows:

1. Velocity
   ∑ of original estimates of all accepted work
2. Work Capacity
   The sum of all work reported during the Sprint, whether the SBI toward which the work was applied finished or not.
3. Focus Factor
   Velocity ÷ Work Capacity
4. Percentage of Adopted Work
   ∑(Original Estimates of Adopted Work) ÷ (Original Forecast for the Sprint)
5. Percentage of Found Work
   ∑(Original Estimates of Found Work) ÷ (Original Forecast for the Sprint)
6. Accuracy of Estimation
   1-(∑(Estimate Deltas) ÷ Total Forecast)
7. Accuracy of Forecast
   (∑Original Estimates) ∑ (∑Original Estimates + ∑Adopted Work + ∑Found Work)
8. Targeted Value Increase (TVI+)
   Current Sprint's Velocity ÷ Original Velocity
9. Success at Scale
   For each Point on the Fibonacci Scale (F_p), the formula is:
   (∑No. Accepted Attempts of scale F_p) ÷(No. of All Attempts of scale F_p)
10. Win/Loss Record
    Each Sprint is a Win only if:
    a) A minimum of 80% of the Original Forecast is Accepted
    b) Found + Adopted Work During the Sprint remains at 20% or less of the Original Forecast.

When we say Targeted Value Contribution is up 200%, we want it clear and demonstrable what we mean: a doubling in the Team's ability to successfully resolve requested complexity. TVC+ (Targeted Value Contribution increase) allows us to compare the increase in profitably applied horsepower of the Team with the increase in revenue generated by the Product Owner's backlog.

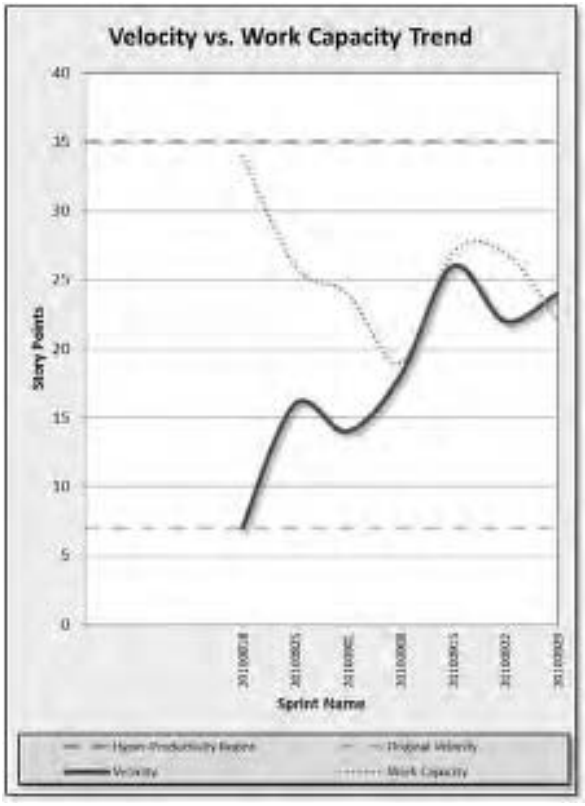

Figure 3. Velocity and Work Capacity

We naturally expect that the Team's Work Capacity (the measure of their full horsepower) will be higher than their Velocity (the measure of their ability to turn their effort into requested and approved Value). In Figure 3. note that the dotted line which marks Work Capacity is usually equal or above Velocity.

Work Capacity may, on rare occasions, drop below Velocity. This is because Velocity is calculated based on the Original Estimates of work while work Capacity is calculated based on the sum of actual work reported. In this rare inversion scenario, it indicates that the Team has been overestimating the complexity of the work requested.

The Focus Factor, calculated as the ratio of *Velocity ÷ Work* capacity, should remain in the neighborhood of 80% on average for a healthy Team. In Figure 4. , we see a Team that was struggling for the first three Sprints. These data points below 80% indicate a Team that is disrupted by external events or otherwise incapable of turning their Forecast work into Accepted Work. When the Focus Factor goes too high, it generally indicates that either the Team have been under forecasting their ability in order to appear "perfect", or are ignoring other organizational responsibilities which may blow up in the near future.

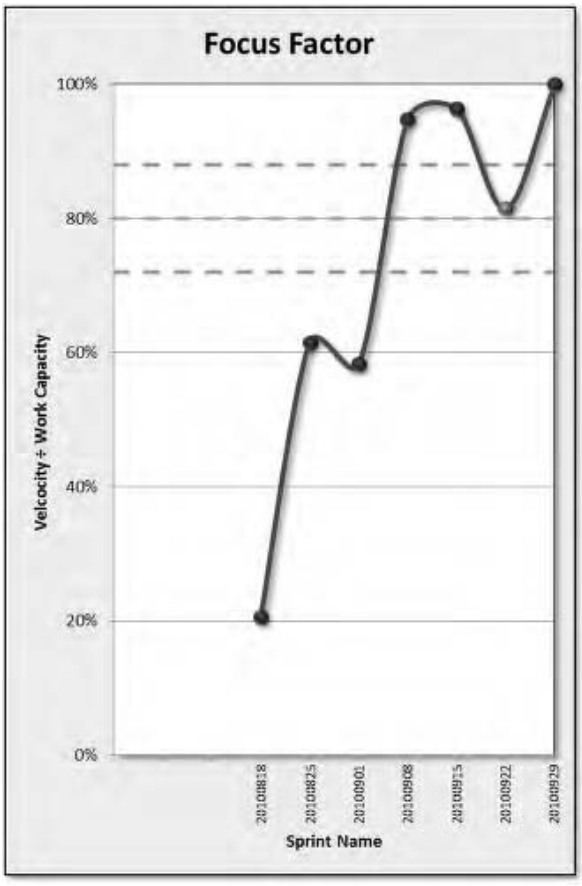

Figure 4. Focus Factor

In Figure 5. , below, we see an example of raw data from the RoboScrum workbook.

Found Work is work associated with a piece of Forecast Work which is above and beyond what was initially expected but which must be completed to deliver the original work item.

Adopted Work is work that is brought forward from the Product Backlog at any point during the Sprint because the Team has completed their original Forecast early.

As a percentage of the original Forecast, these two values when added together should not exceed 20% of the original Forecast in an average case. As you see in the examples, the Author uses a rolling 10-Sprint window to evaluate the average performance of the metrics presented herein.

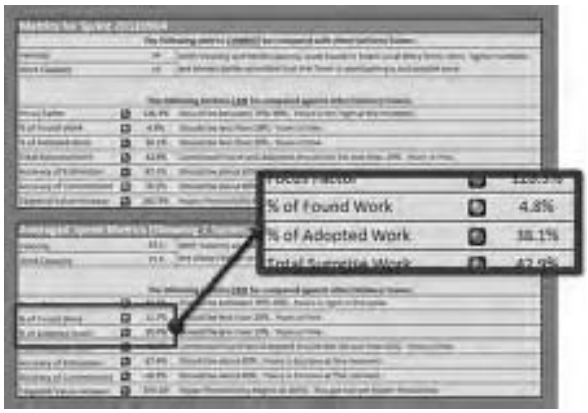

Figure 5. Adopted & Found Percentages

Figure 6. and Figure 7. below, show the trending accuracies of Forecast and Estimation for a new Team with only seven one-week Sprints under their belt are shown.

Forecast Accuracy refers to the Team's ability to come together in the Sprint Planning Meeting, select and devote themselves to a body of work that, with 80% accuracy, represents what they can achieve during the coming Sprint. Any Team that achieves 100% Accuracy is likely under some form of external pressure and is, therefore, underforecasting work because of fear of reprisal or some similar, dysfunctional dynamic. When this number goes above 90%, the Scrum Master needs to evaluate the environment of the Team to be sure that they feel safe making a good faith effort at more work, even when the bottom 20% of the Forecast is not accepted by the end of the Sprint.

When Forecast Accuracy dips below 75-80%, it is generally because the Team is heavily randomized or is not being adequately protected by the Scrum Master during the Sprint. Especially in scenarios where multiple Teams do Sprint Planning on the same day, it is often the case that a Product Owner of one Team is failing to coordinate their Team's needs

with the Product Owner of another Team, resulting in an unplanned quantity of work landing on a Team's shoulders early in the Sprint. A good Scrum Master will be sure that the Sprint is protected from this behavior. A great Scrum Master will work with the Product Owners to be sure their PBLs are coordinated ahead of Sprint Planning so that all Teams' Sprint Planning Meetings are effective predictors of the organization's achievement in the coming Sprint.

Accuracy of Estimation reflects the Team's ability to correctly estimate the body of work during Sprint Planning. This number, again, should remain around 80% in healthy Teams who are challenged by their work.

When Accuracy of Estimation goes too high (above 88% on average), it is likely that the Team is being overly conservative and spending an inordinate amount of time planning, digesting, researching and so on. In those scenarios, we advise the Team to accept a bit more risk and spend more time achieving the work than studying it for planning purposes. This generally results in shorter meetings, higher productivity and (for those who prefer shorter meetings) happier Teams.

When Accuracy of Estimation dips too low (below 72% on average), the Scrum Master should begin investigating pressures on the Team. It is often the case that the User Stories/PBIs are too poorly understood, that the Product Owner is unavailable to the Team during the Sprint, that the Team does not understand the technology or product that they are being asked to build/modify, or that the requirements are changing during the Sprint. There is also the potential that you have a Knowledge Vampire on the Team who is hoarding system-critical knowledge and keeping everyone else in the dark.

All of these structural deficits are correctable, and this metric lets the Scrum Master know when such corrections are necessary and when the corrections have yielded the desired state of confidence.

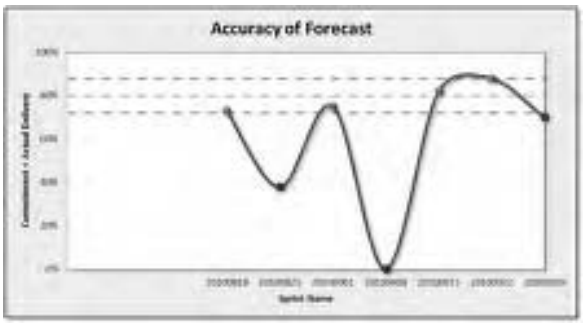

Figure 6. Forecast Accuracy

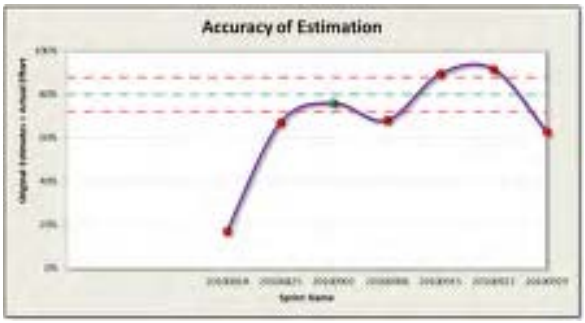

Figure 7. Estimation Accuracy

In Figure 8. , the sample data taken from RoboScrum indicates that the Team has achieved a 242.9% TVI+. This indicates that their Velocity for the Sprint which just concluded is just over twice their Original Velocity. In Shock Therapy [16], this would be a Team that is ready to begin taking back some control over their Scrum adoption from the Shock Therapy Coach provided that their numbers stay up and the changes they propose adhere to the principles and ethics of the Scrum Framework.

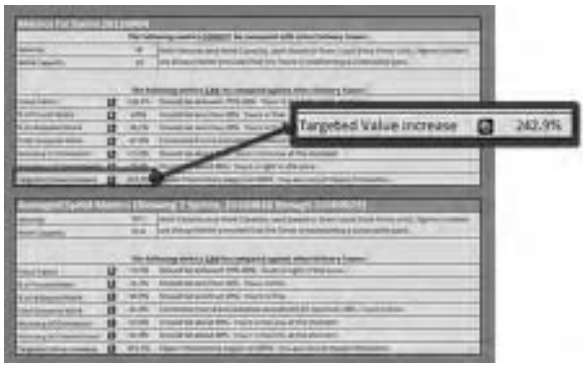

Figure 8. Targeted Value Increase

To give Teams a sense of confidence during Sprint Planning Meetings, as well as to help them select the granularity to which work should be digested while using INVEST, the Success at Scale data is invaluable. In Figure 9. , below, you see that the data indicates a high degree of competency with SBIs at positions 1 and 2 along the Fibonacci sequence. Those SBIs estimated as a 3, 5 or 8 are also in the range of acceptably successful; however, 13 is too risky for anything that may represent too large a commitment for the coming Sprint. In the example below, no SBIs of a scale above a 13 had been tried.

Suppose your Team has a Velocity of 40 points per Sprint and is just beginning a new Sprint Planning meeting. The "S" in INVEST (Sized to Fit) suggests that it is unwise to accept a single SBI larger

than 50% of your Velocity or, in this scenario, 20 Points.

Further suppose that the first PBI up for consideration has previously been estimated as a 34. Holding all other variables consistent, and though clearly small enough to fit into the Team's expected Velocity, a quick glance at the Success at Scale chart lets us know that the Team has not historically been successful with individual SBIs larger than an 8.

It is never recommended that Teams be denied the opportunity to try. If they feel confident, they should be allowed to proceed. But a Scrum Master who is advising a Team should ask a few questions before the Team simply adopts the PBI as its highest priority SBI. These questions may include, "What is different about this 34 point card than the other cards smaller than it which have not succeeded?", "INVEST advises us against accepting any single SBI larger than 50% of our Velocity, or 20 Points in our current case. Why do you think this 34 Point PBI is safe to forecast for completion as-is?", etc.

As described in section 3.3, above, we advise our Teams to accept the *largest* piece of work for which the Team has about 80% confidence that they can achieve, and which has passed INVEST. Given the data in Figure 9. , the Team may choose to break the 34 point PBI into a series of 3, 5 and 8 Point SBIs before proceeding. But, again, it is important that a confident Team be allowed to try the unprecedented when it is organizationally safe and responsible to do so.

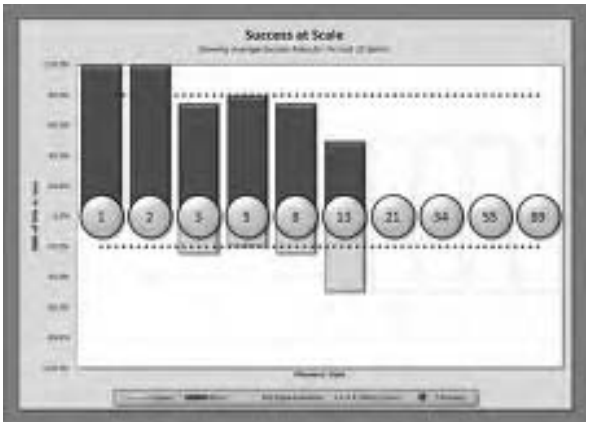

Figure 9. Success at Scale

Careful tracking of unplanned work is essential to detecting and removing waste from a Hyperproductive Team. As seen in Figure 10. , a Sprint can only be considered a Win if at least 80% of the Original Forecast was approved by the Product Owner, and the combined surprise work (Found + Adopted) remains at a level of 20% or less of the Original Forecast.

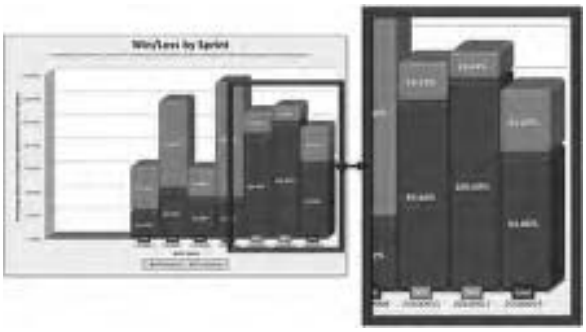

Figure 10. Win/Loss Record

5. Conclusions

Hyperproductive Scrum Teams, and the Scrum Masters who advise them, need a simple set of metrics to provide subtle control that maintains safe and consistent growth. Without these metrics, performance of the Team can be unstable and loss of control will result in lowered Velocity. We carefully avoid hours as a means of tracking progress as it introduces waste into the system, lowers Velocity, and reduces predictability. The resulting simple set of metrics are easy to implement and have a powerful effect on the performance of 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] J. O. Coplien, "Borland Software Craftsmanship: A New Look at Process, Quality and Productivity," in *5th Annual Borland International Conference*, Orlando, FL, 1994.

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

[4] 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.

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

[6] C. Jones, *Software Engineering Best Practices: Lessons from Successful Projects in the Top Companies*: McGraw Hill, 2010.

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

[8] J. Sutherland, *Scrum Handbook*: Scrum Foundation, 2010.

[9] J. Sutherland and I. Altman, "Take No Prisoners: How a Venture Capital Group Does Scrum," in *Agile 2009*, Chicago, 2009.
[10] 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.
[11] 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.
[12] 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.
[13] D. Sull and K. Eisenhardt, "Simple Rules for a Complex World," *Harvard Business Review,* pp. 69-76, 2012.
[14] L. Putnam and W. Myers, *Industrial Strength Software: Effective Management Using Measurement*: IEEE, 1997.
[15] J. Spolsky. (2005, Hitting the High Notes. *Joel on Software*.
[16] B. Tuckman and M. A. Jensen, "Stages of Small-Group Development Revisited," *Group & Organization Management,* vol. 2, 1977.
[17] T. Ohno, *Toyota Production System: Beyond Large Scale Production*: Productivity Press, 1988.
[18] A. Cockburn, *Crystal Clear: A Human-Powered Methodology for Small Teams*: Addison-Wesley, 2004.