Do Better Scrum

An unofficial set of tips and insights into how to implement Scrum well

Written by Certified Scrum Coach and Trainer Peter Hundermark

Why this guide?

Certified Scrum Trainer and Coach Jim York says: **Scrum is Simple. Doing Scrum is Hard.℠**

Many people I meet in organisations say that they find it hard to know how to get started with Scrum. Others have teams that are following some Agile practices, yet are far from becoming what Jeff Sutherland terms *hyper-productive*.

I hope this little booklet may be a source of inspiration to help you do better Scrum and Agile. More importantly I hope it might encourage you to drag yourself, your team and your whole organisation away from the old ways of working that simply don't, well, *work* and find new ways that lead to greater quality, faster delivery and above all, more fun.

Let me know what you like and dislike so I can improve it.

Now go and do it!

Peter Hundermark
Cape Town, November 2009
Second edition

What is Scrum?

Derivation

Scrum is a management framework within which complex products can be developed. Scrum is derived from work in knowledge management, complex adaptive systems and empirical process control theory. It contains influences from observed software development patterns and the Theory of Constraints.

Scrum and Agile

Scrum is the most popular of the Agile methods. It is frequently used in conjunction with Extreme Programming (XP).

What is the problem?

- Releases take too long
- Stabilisation takes too long
- Changes are hard to make
- Quality is falling
- Death marches are hurting morale

For decades software developers have been trying to employ defined methods of working and managing projects. Defined methods are appropriate when the inputs are well-defined and the method of converting these into outputs is predictable. Software development and other forms of complex work are not suited to such methods. And the high rate of project failures and customer dissatisfaction illustrates this amply.

How does Scrum help to solve it?

Alistair Cockburn [Cockburn 2008] describes software development as ‘a cooperative game of invention and communication’.

Traditional development methodologies rely on documents to record and pass on knowledge from one specialist to the next. Feedback cycles are too long or even non-existent. Decades of project under-performance has shown these ways of working to be an outright failure.

Scrum provides a platform for people to work together effectively and and relentlessly makes visible every problem that gets in its way.

The Essence of Scrum

The essence of Scrum is:

- The team is given clear goals
- The team organises itself around the work
- The team regularly delivers the most valuable features
- The team receives feedback from people outside it
- The team reflects on its way of working in order to improve
- The entire organisation has visibility into the team's progress
- The team and management honestly communicate about progress and risks

This way of working is based upon values of self-respect, respect for others, trust and courage.

Why is Scrum silent on software development practices?

Scrum does not attempt to instruct teams how to do their work. Scrum expects teams to do whatever necessary to deliver the desired product. It empowers them to do so. Development practices and tools change and improve all the time and good teams will constantly work to take advantage of these.

Applicability

While Scrum was first applied to development of software products, it is suited to all types of complex work. It is used today to manage software and hardware development, support, advertising and marketing, churches and entire organisations.

How does Scrum map to traditional methods?

The short answer is that is does not. Agile and Scrum are based on a different paradigm. Founders Jeff Sutherland and Ken Schwaber have frequently stated that attempts to map defined to empirical methods are futile.

Will Scrum succeed in my organisation?

That depends on you! The implementation in your organisation may fail because of a lack of resolve by people to overcome the problems that Scrum will certainly expose. Yet thousands of teams on all continents and in every industry are succeeding in making their world of work better today than yesterday.

Agile Manifesto

In February, 2001, seventeen independent 'lightweight' software methodologists and thinkers met to talk and find common ground. Amongst them were Scrum co-founders Jeff Sutherland and Ken Schwaber, together with Mike Beedle, who worked on the initial Scrum patterns and co-authored the first book on Scrum. The group named themselves 'The Agile Alliance' and agreed on a *Manifesto for Agile Software Development*. They further defined a set of twelve principles behind the manifesto, reproduced in full on the opposite page.

Commentary

The Agile Manifesto and its twelve principles have stood up (so far) for nearly a decade. They remain the litmus test for all Agile methods and practitioners by which to evaluate their way of working. The most vocal criticism has been the bias towards software development, when Agile methods are more broadly applicable.

Note

Scrum is one flavour of Agile. Adherence to the Agile Manifesto and all its principles does not necessarily mean a team or organisation is following Scrum. However, non-adherence to ANY ONE of these principles does imply that you are NOT doing Scrum (or Agile)!

space for notes

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Principles behind the Agile Manifesto

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Highsmith (2001)

The Scrum Roles

Roles Introduction

There is no Project Manager role in Scrum. The responsibilities of the traditional project manager are divided over the three roles in the Scrum Team:

- The Product Owner manages the *product* (and return on investment)
- The ScrumMaster manages the *process*
- The team manages *itself*.

This is challenging to individuals who currently fulfil this role and to managers in organisations in which they work. Michele Sliger and Stacia Broderick have written a helpful guide to the transition from Project Manager to Agile Coach [Sliger and Broderick 2008].

There are no appointed leaders of the Scrum Team beyond the Product Owner and ScrumMaster; none is required. The need for line managers is reduced, as teams manage themselves to a great extent. It is not uncommon for 50 team members to report directly to a single line manager in an organisation that has made the transition to Agile.

Self-organisation

Self-organisation does not at all imply a *laissez-faire* approach; on the contrary, self-organised teams are highly disciplined. They are given full autonomy and carry correspondingly greater responsibility for delivery accordance with their own commitments. They are encouraged to take reasonable risks and to learn through failure and self-reflection. High trust and high commitment is an automatic outcome of truly self-organising teams.

Teams new to Scrum will require some encouragement to explore their new, broader boundaries and to take ownership. They frequently need to overcome strong 'muscle memory' of the poor ways in which they were managed and worked, sometimes for years.

Self-organisation is not an option in Scrum; it is a core principle. Without this, high-performing teams will not happen. *Caveat emptor!*

Product Owner

The responsibilities of the Product Owner role are:

- Working on a shared vision
- Gathering requirements
- Managing and prioritising the Product Backlog
- Accepting the software at the end of each iteration
- Managing the release plan
- The profitability of the project (ROI)

Metaphor: The Product Owner is a CEO.

ScrumMaster

The responsibilities of the ScrumMaster role are:

- Empowering and shepherding the team
- Removing impediments
- Keeping the process moving
- Socialising Scrum to the greater organisation

Metaphor: The ScrumMaster is a facilitator, coach, mentor and bulldozer!

Team

The responsibilities of the Team or Team Member role are:

- Estimating size of backlog items
- Committing to increments of deliverable software
- —and delivering it
- Tracking own progress
- Is self-organising—but accountable to the Product Owner for delivering as promised

- *Team members can be developers, testers, analysts, architects, writers, designers and even users. The team is cross-functional, which means that between all its members they possess sufficient skills to do the work. There is no dictated leadership hierarchy within the team members.*
- *The Scrum Team comprises all three roles: one Product Owner, one ScrumMaster and five to nine Team Members.*

The Sprint Meetings

The sprint is the heartbeat of the Scrum cycle. It is bookmarked by *sprint planning* at the start and by the *sprint review* and *sprint retrospective* at the end. The length of the sprint is fixed and is never extended. Most Scrum teams choose two, three or four weeks as their sprint duration. Each day during the sprint the team holds a *daily Scrum meeting*. Every meeting in Scrum is strictly *time-boxed*. This means that is has a maximum duration. It does not means that it needs to occupy this full time. For a 30 day (or four week) sprint the time boxes for planning 1 & 2, review and retrospective are set at four hours each. For shorter sprints they should be adjusted in proportion to the sprint length.

Some key attributes of the meetings are described in the following sections. First, though, I have collected a few experiences I think are worth sharing.

- *I find two-week sprints a good length to start with. After three sprints, let the team re-assess the sprint length.*
- *Teams need three sprints to grasp the new concepts, break down old habits and start to gel as a team.*
- *Never do sprint planning on a Monday morning. The team is not yet at its best and it is the most common day for holidays and sickness. Never hold reviews or retrospectives on a Friday afternoon. The team is tired and thinking about the weekend. Therefore choose sprint boundaries on Tuesdays to Thursdays.*
- *Teams running two-week sprints might be tempted to hold all sprint boundary meetings in one day. In other words, start the day with the review, then the retrospective; after lunch do sprint planning parts 1 and 2. The thinking is to get all the meetings out of the way and have 9 full days to do the work. In my experience there are two problems with this approach:*
  - *The team does not get that these meetings are part of the work—in fact the most important part to get right!*
  - *During the last part of the day—sprint planning 2—the team is brain-dead.*

  Yet, as always, let the team try it out if they so wish!

Sprint Planning - Part 1

Part 1 of sprint planning (SP1) is really a detailed *requirements workshop*. The product owner presents the set of features he would like and the team asks questions to understand the requirements in sufficient detail to enable them to commit to delivering the feature during the sprint. The team alone decides what it can deliver in the sprint, taking into account the sprint duration, the size and current capabilities of its members, its definition of *DONE*, any known holidays or leave days and any actions it committed to during the retrospective held just prior this meeting.

The product owner must be present during this meeting to lead the team in the right direction and to answer questions—and they will have many. The ScrumMaster must ensure that any other stakeholder needed to help the team understand the requirements is present or on call.

Any new backlog items for inclusion in the current sprint and not previously estimated will be sized immediately during this meeting. This not, however, an excuse to avoid grooming the backlog—see below!

At the end of SP1 the team commits to the Product Owner what they believe they can deliver in the form of *running tested features*. An experienced team may use historic velocity as a predictor (*'yesterday's weather'*). This is known as *velocity-based planning*. My recommendation to most teams is to do *commitment-based planning*. The backlog items the team has committed to is called the *selected product backlog*.

Sprint Planning - Part 2

If part 1 is a requirements workshop, part 2 of sprint planning (SP2) is a *design workshop*. In this session the team collaborates to create a high-level design of the features it has committed to deliver. An outcome of this session is the sprint backlog, or the list of tasks that the team collectively needs to execute in order to turn the items in the selected product backlog into running tested features. This set of tasks is called the *sprint backlog* and is most often represented on a physical *task board*.

During SP2 the team may have additional questions regarding the requirements. The ScrumMaster must ensure that the Product Owner and, if necessary, other stakeholders are on call to answer them.

Design, as everything else in Agile, is emergent. Also, the meeting is time-boxed. So it is normal that the team won't get the design perfectly done in this session and will discover more tasks during the sprint. This is not a sign that something is wrong. They will simply grab a post-it note and pen and create more tasks whenever necessary during the sprint.

✺ *You will know that SP2 is working when the team is gathered together around the white board discussing noisily or even arguing about the 'best' or 'right' way to implement a feature.*

Daily Scrum Meeting

The daily Scrum meeting is one of the three primary *inspect and adapt* points in Scrum. The team meets to communicate and synchronise its work. Since the team is collaborating, this is essential to ensuring continued progress and avoiding work blockages. The team will also continuously assess its own progress towards achieving its *sprint goal*.

The daily Scrum meeting is NOT for reporting progress to the ScrumMaster or Product Owner or anyone else. The Product Owner may attend provided he is well-behaved, which means speaking only when he is spoken to! The ScrumMaster makes sure, using all her skills, that each team member has signed up for some work for the next 24 hours, that this work is directed solely at helping the team as a whole deliver the next item in the backlog and that any impediments to doing this work are bulldozed out of the way as fast as possible. The ScrumMaster also ensures the meeting is restricted to 15 minutes, which, surprisingly is ample time.

Jason Yip [Yip 2006] provides a useful guide to help ScrumMasters to run this meeting well.

Sprint Review

The sprint review is sometimes, and wrongly, called the sprint demo. While it does include a demonstration of the new features the team has completed during the sprint, its primary purpose is to *inspect* what the team has delivered and gather feedback from the attendees to *adapt* the plan for the succeeding sprint. Thus it is much more than a demonstration.

The focus of the sprint review is the *product* the team is building.

When asked who should be invited to the sprint review, I answer ‘the whole world’. My intent here is to help the ScrumMaster and the entire organisation understand that the direct attention and feedback of a broad constituency of the organisation is crucial to maximising the value the team will deliver in succeeding sprints. Sprint reviews have many possible outcomes including cessation of the project. In most instances, the team is authorised to continue for another sprint and a goal for this next sprint is set.

space for notes

Sprint Retrospective

The sprint retrospective is the final meeting of the sprint. It follows immediately after the sprint review. It is never omitted!

Whereas the sprint review is focussed on the product, the retrospective is focussed on the *process*—the way in which the Scrum team is working together, including their technical skills and the software development practices and tools they are using.

And whereas the sprint review is open to the world, the sprint retrospective is restricted to the members of the Scrum team—that is the Product Owner, development team members and ScrumMaster. Outsiders, including managers at every level in the organisation are strictly excluded unless specifically invited by the team.

This rule is to be understood in the context of the goal of the meeting, which is to inspect at a deep level how the team is collaborating and performing and to take action to improve. This often requires deep introspection and sharing, which in turn requires a safe and secure environment. Norman Kerth's Retrospective Prime Directive undergirds this: 'Regardless of what we discover, we understand and truly believe that everyone did the best job they could, given what they knew at the time, their skills and abilities, the resources available, and the situation at hand.' [Kerth 2001].

Boris Gloger [Gloger 2008] offers a simple pattern called *Heartbeat Retrospectives* for new teams to learn to hold valuable retrospectives. Esther Derby and Diana Larsen [Derby and Larsen 2006] provide helpful activities for facilitators of Scrum retrospective meetings.

Estimation Meeting

This meeting is not mentioned in some of the Scrum literature, but is essential if you want to achieve a continuous flow of the most valuable *done* features from your teams.

During every sprint, the product owner convenes one or two meetings where the Scrum Team and, if required, other stakeholders, meet to size backlog items that have been added or re-size large items that need to be split into smaller ones for tackling in the next sprint or two.

Teams need to devote 5-10% of their time during the sprint to preparation for the next sprint or two. The estimation meeting described above is an example. Other examples are story-writing and release planning workshops. This is important to avoid a stop-start effect at the boundary of every sprint. The other implication, of course is that teams should spend 90-95% of their time doing the work of the current sprint!

Scrum Artefacts

Scrum mandates only four artefacts:

- Product Backlog
- Sprint Backlog
- Burndown Chart
- Impediment Backlog

Scrum is purposely silent about all other documentation and artefacts. This sometimes leads to the misunderstanding that Agile teams don't need to do any documentation. I coach teams to produce only those artefacts that are *really* valuable to themselves and to others in the future. This cuts out worthless work and unnecessary killing of forests!

Product Backlog

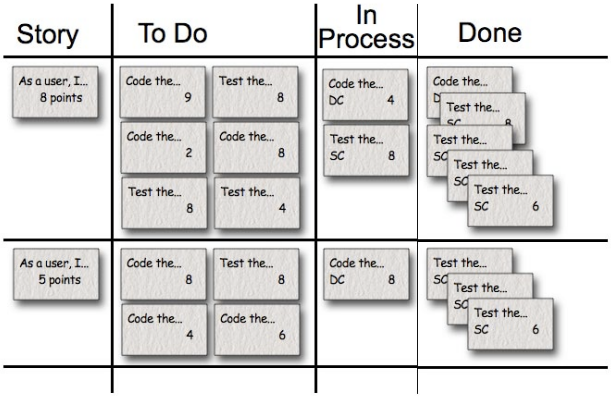
The product backlog is simply a list of work items that need to be done over time. Items may be added to the backlog by anyone, but only the Product Owner has the right to determine the order in which they will be executed by the team. Of course a good Product Owner will negotiate this with stakeholders and the team.

Requirements are *emergent*, meaning we do not and cannot know up front every detail about what we want in a product. Therefore the Product Backlog is a living document and requires constant *grooming* to keep it current and useful. Many new items will be added over time; existing items are disaggregated to multiple, smaller items; some items may be removed on realising that a desired feature is no longer needed. Moreover, items need to be sized in order to determine the likely relationship between value, time and cost. And, of course, the order of items in the backlog will change as the relative value between them is seen differently today from yesterday.

In nearly all cases it is sufficient, and generally preferable, to create and maintain the Product Backlog as a set of stories written on physical 150 x 100 mm (6" x 4") cards. Ron Jeffries [Jeffries 2005] coined the alliterative triplet Card, Confirmation, Conversation (the 3C's) to describe how to work with stories. The stories are commonly written from the perspective of a user of the product. Mike Cohn's book on user stories [Cohn 2004] will tell you all you need to know.

Sprint Backlog

Most teams will know the sprint backlog as the task board, which is the physical representation of the list of work they have committed to do during the current sprint. The task board is an example of a *kanban*, a Japanese word meaning sign or visible signal. It tells the whole team and anyone else what work they have planned or the sprint and their current status.



Sprint Backlog or Task Board

Adapted from http://epf.eclipse.org

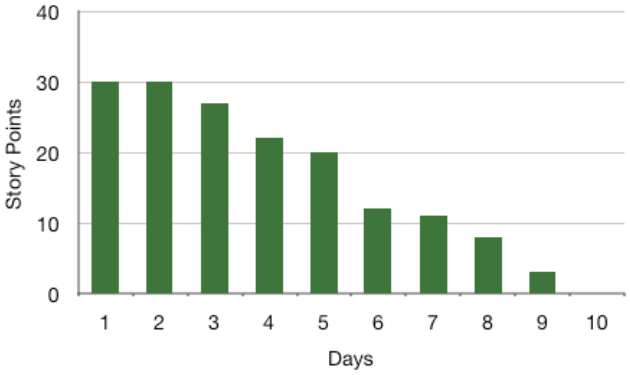
Sprint Burndown Chart

The sprint burndown chart is designed to help the team monitor its progress and be a leading indicator of whether it will meet its commitment at the end of the sprint. The classic format requires teams to estimate the duration of each task in hours and on a daily basis to chart the total remaining hours for all uncompleted tasks.

I coach teams to burn down their sprint in story points. The rationale behind this is:

- Estimating new tasks and re-estimating in-progress tasks requires effort
- Estimating tasks is inaccurate
- Estimating in units of time harks back to the bad old ways of working
- Completion of tasks delivers no value; only completed stories (features) deliver value.

So in my coaching world the sprint burndown is just like the product or release burndown, except the the X-axis scale is days rather than sprints.



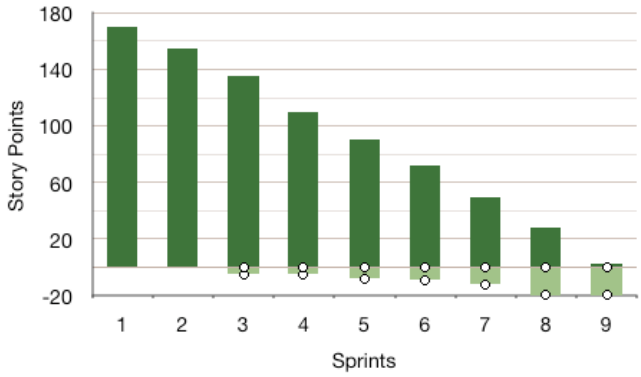
Sprint Burndown Chart

Product or Release Burndown chart

The product burndown chart, sometimes known as the release burndown chart, measures the rate of delivery of a stream of running, tested, features over time. This rate is know as the team's velocity. Because features vary in complexity, and therefore effort and time, we use a scale to compare their size. The most common scale is known as story points. Once a team has worked together for a few sprints, it generally establishes its velocity within a definable range. Product Owners then use this velocity to predict the rate at which the team will deliver features in the future, leading to credible release plans.

I coach teams to use an alternative form of the product burndown chart that simultaneously allows Product Owners to track changes to the product backlog. This is essential, given the dynamic nature of this list.

Using this chart Product Owners are able to report progress, determine release dates and predict release scope.



Product Burndown Chart

Impediment Backlog

The impediment backlog is simply the current list of things that are preventing the team from progressing or improving. These are things the ScrumMaster must bulldoze out of the way in her never-ending quest to help the team be the best they can. Impediments range from getting the coffee machine fixed to replacing the CEO! A good ScrumMaster will try to remove impediments within 24 hours of them being identified. (OK, perhaps not the CEO.)

Starting Scrum

Ken Schwaber [Schwaber 2007] says there is nothing that needs to be done before starting Scrum. I interpret this to say that there is no tailoring of the process needed in order to start. Nevertheless the literature is thin on how to get going and all of us struggled getting our first Scrum team going without outside help.

The best thing you can do is hire an experienced coach. Failing that, you can try using a pattern that has worked for me with dozens of teams.

Obviously (I hope) you need a Scrum team. This means a Product Owner, a ScrumMaster and five to nine team members. Then follow this sequence of steps, which will be detailed in the next pages.

1. Train the Scrum Team in the basics of Scrum
2. Establish the vision
3. Write user stories to form the product backlog
4. Order the backlog items by business value
5. Size the backlog items
6. Re-order the backlog, as necessary, by additional factors
7. Create the initial release plan
8. Plan the first sprint
9. Start sprinting!

- *I start new teams with a full day's training in the essentials of Agile and Scrum. This is sufficient to start a Scrum Team that has an experienced ScrumMaster/coach to support them during their first sprints.*
- *Steps 2 to 7 can be completed comfortably in a two-day product workshop attended by the entire Scrum Team. The best workshops are attended by additional stakeholders like enterprise architects, managers and business people.*

Training

In my team training I use a lot of group exercises and games to illustrate the principles. I cover all or most of the following topics:

- The power of self-organisation
- Empirical versus defined processes
- The value of collocation
- The importance of trust
- Agile principles (Agile Manifesto)
- The Scrum flow (cycle of meetings)
- Roles and responsibilities (3 Scrum roles plus more)
- Using user stories for requirements
- Agile estimation using planning poker
- Concepts of size and velocity
- Done!
- Using a task board
- Monitoring progress (burndown charts)
- Simulation of an entire sprint

Visioning

Katzenberg and Smith [2002] have confirmed that having clear goals is essential in the creation of high-performing teams.

The Product Owner will usually share his or her vision for the product. One technique I use is to get each member to write his or her personal version of the vision. Then members pair up and work to create a single shared vision statement from their initial efforts. The process of pairing up continues until the entire team collaborates to formulate a single statement.

This exercise utilises the power of the group and results in greater commitment to the resulting vision. The team will display the vision prominently in their work space. The visioning exercise may take one to three hours.

Creating the Product Backlog

The next stage is to hold a story-writing workshop. It is advantageous to involve business stakeholders here. Certainly the whole Scrum Team is involved. Of course, the ScrumMaster (or coach) facilitates.

Writing good user stories is simply a matter of practice. The Pareto principle (80/20 rule) applies here, as always.

Mike Cohn [Cohn 2004] has provided an excellent guide to writing good user stories. I encourage every Product Owner to have his personal copy always to hand!

> Template: As a **role / persona**, I want **function** so that **reason**.
> Example: As a **programmer**, I want **coffee** so that I can **stay awake**.

Bill Wake suggested the helpful INVEST acronym for the attributes of a good user story [Wake 2003].

> **Independent**—ideally can be implemented in any order
> **Negotiable**—and negotiated
> **Valuable**—to the customer
> **Estimatable**—enough to rank and schedule it
> **Small**—and with short descriptions
> **Testable**—I could write a test for it

At a minimum the backlog needs to contain enough items for the team to plan the first sprint. More commonly, the backlog contains all the items that make up the next planned (or hoped-for) product release.

As a guide, the backlog should be 80% complete (but not ordered or sized) by the end of day one. The first part of day two can be used to add the final 20% and to resolve any open questions. The overnight break is a very useful hiatus that may spark fresh thinking about the work.

Ordering the Backlog

The backlog is now ordered by business value. This appears easier to say than to do well. Mike Cohn [Cohn 2006] describes two methods for prioritising desirability: the Kano model of customer satisfaction and Wiegers' relative weighting approach. What I like about both is that they consider not only the benefit of having the feature, but also the penalty of excluding it. This is particularly helpful when the backlog includes technical items whose business value is not immediately obvious.

At the minimum, the Product Owner's subjective judgement of the value of one feature against another is good enough. Better is a somewhat quantitative assessment using a technique similar to planning poker.

However it is done, it is a core responsibility of the Product Owner to order the backlog.

Sizing the Backlog

Project planning has always had estimation as the lynch-pin. Project managers like myself have spent weeks calibrating complex models with historical data.

The simple reality is that the best of these techniques yield no better results than much simpler and faster techniques such as planning poker and affinity estimating. Planning poker works partly because has a solid basis in theory, but mostly because the people who estimate are those who will do the work. Who would have thought that?

Planning poker is fast. A practised team will estimate at an average rate of 3 minutes per work item. Planning poker is accurate. Estimates using planning poker are as good as the best traditional methods. And planning poker is fun. It takes away the pain usually associated with this topic. Commercial planning poker cards are obtainable from several sources at a cost of about $10 (including shipping) per set of four packs. You can print your own on plain card for next to nothing.

Affinity estimating is even faster than planning poker. It is great for getting started when we have an entire backlog to estimate and time is more important than great accuracy and information sharing.

However, we still need to understand a few key concepts about agile estimating:

- We size items relative to one another. Why? Because as humans we find this more natural and the results are more reliable. So it's easy to agree that 'this story is twice the complexity of that one' even though we don't know how long each will take to implement.
- We size items using units of complexity rather than time. Why? Because it allows us to separate the rate at which a team works from the size or complexity of the work. This shields us from having to change our estimates according to who does the work, or as the teams skills and capacity change over time. We use 'story points' as units.
- We use a non-linear sizing scale because the difference between a '1' and a '2' is obviously more meaningful, relatively, than that between '20' and '21'. My preference is to use the pseudo-Fibonacci numbers : 1, 2, 3, 5, 8, 13, 20, 40 and 100. And I define 1 to 8 or perhaps 13 as the size range of features a team can delivery in one sprint. The higher numbers are reserved for large stories ('epics') that will need to be split into smaller stories before they can be taken into a sprint.
- We relate our size estimates to time by means of *velocity*, which is the rate at which a team can deliver running, tested features to their product owner. We say that a team has a velocity of '25' when they are able to deliver at the end of every sprint 'done' stories whose sizes, on average, add up to 25 points.

> *One team's sizes are generally not comparable to those of another. This is immaterial unless we have multiple teams working off the same product backlog. This is an issue of scaling Scrum and is easily solved, but is not a topic for this little guide.*

Re-Ordering the Backlog

After sizing the backlog items we may find that some items should be re-ordered. Factors to take into account in addition to the business value include:

- Size: we will naturally choose to implement a simple (small) story ahead of a complex (large) story if they have similar business value.
- Learning: we may choose to implement a story earlier if it will help the team learn about the business domain or a new technology.
- Risk: we may choose to implement a story early because doing so will mitigate an identified risk. Obvious examples are items that will help establish performance and scalability limits.

We must always remember that the Product Owner has final say on ordering the backlog.

Release Planning

Once we have ordered and sized the backlog the next step is creating the initial release plan. To do this we need an estimate of our team's velocity. However we haven't done any work yet, so we use a simple technique known as *commitment based planning*.

First, of course, we must be sure to know the team size during the sprint—whether any member will be away on leave, training, etc. And we must choose the sprint length—I usually recommend two weeks for a new team. And we must create a definition of *DONE* for the team—what does it mean when the team says it has completed a story.

The ScrumMaster now picks the first item from the top of the backlog and asks the team 'can you complete this item during the sprint?'. She continues to do this until the team is no longer confident to add items. Counting up the story point values of all the committed items, the team has its initial estimate of velocity.

This velocity value is used to apportion the remaining backlog items (at least those that have been estimated) over succeeding sprints. This list of items attached to sprints is our initial release plan. It is accurate? Perhaps not, but it is probably more credible that anything a project manager could produce before at such an early stage of a project. And as we start work and complete one sprint after another, we will begin to chart our actual velocity and use this as a predictor of future output. So the release plan is a living thing that becomes more and more reliable as we progress.

Don't let the product workshop drag on. It is possible within two days for any team and any product to prepare enough backlog items for the first few sprints at least.

Collocation and Team Rooms

Alistair Cockburn [Cockburn 2007] defines software development as a *cooperative game of invention and communication*. The Agile Manifesto says developers and business people must work together daily and that face-to-face conversation is the best way to transfer information.

In order to collaborate effectively, there is no single factor that delivers more value than collocation. A study of collocated software development teams [Teasley, Covi, Krishnan, & Olson, 2000] has shown that they are twice as productive as non-collocated teams.

The definition I use for collocation is that team members sit no further than 6 m (20') from their farthest member. It follows that there is a limit to the number of people that can fit into such a space. In practice this number is 8 to 10, which happily matches the maximum size of a Scrum team.

Furthermore, every team member must be able to see every other member without doing more than turn her head or swivel in her chair. This means no dividers between desks—good-bye Dilbertville!

For no good reason beyond habit, organisations—note not teams—resist changing their environment to facilitate effective collocation. This is despite evidence that collocated team spaces require no more floor area, on average, than present layouts.

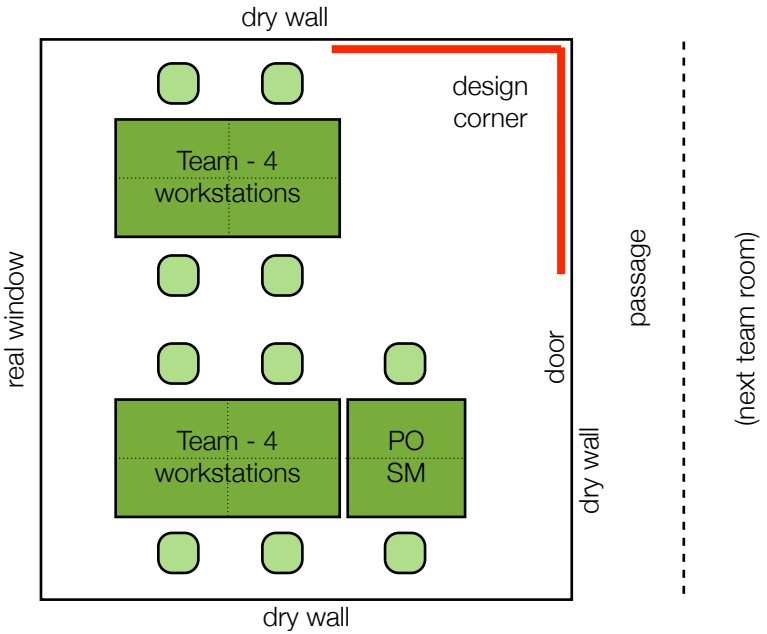
Management will challenge you about building dry walls, believing them to inhibit flexibility. This is simply not true—flexibility is required in the organisation structure, not team spaces!

My experience is that 90% of Scrum teams in organisations new to Agile struggle on with their miserable work spaces for a year before their management is persuaded to make the necessary changes. And once they have been made, everyone agrees that the improvement was immediate and measurable. Why is this??

In the hope that this absurd resistance will diminish, I offer a simple layout of a collocated team room on the next page. It goes beyond the scope of this little guide to provide all the reasoning behind this design—you'll have to hire me if you want that!

- *Team rooms significantly reduce the need for meeting rooms—teams can do sprint planning, daily meetings and reviews in their room.*
- *Be warned that your architect or office designer won't be much help to you. She has not been trained in the design of team spaces.*
- *The cost of the changes (drywalls and furniture) will be recovered in one week to one month! Sounds unbelievable, doesn't it?*

(next team room)



(next team room)

- *Some team space layout parameters:*
  - *Desks must be rectangular to facilitate pairing*
  - *1.5-1.8 m x 0.8 m (5-6' x 2'8") is a good desk size*
  - *1.2 m (4') space from desk to wall*
  - *Design corner 3 x 3 m (10' x 10')*
  - *Typical room size 7 x 8 m = 50 to 60 m² (23' x 27' = 550 to 650 sq.ft.)*
  - *ScrumMaster can see the whole team*
  - *Product Owner has a floating (not permanent) seat in the room*
  - *Dry walls with internal windows create noise barriers and enough space for information radiators without reducing light.*

Metrics

It is reasonable to expect of any manager that she will want, within a reasonable period, to be able to measure the results of a team or organisation's transition to Agile methods.

Happily, there are metrics that are easy and quick to implement. Based on my own research and that of other Agile practitioners, I have described a set of metrics that your team can and should implement [Hundermark 2009]. As soon as the team understands the basics of the Scrum framework and has some idea of its velocity—usually by the end of their third sprint—it is probably time to start measuring. In fact, you can run customer and team surveys before you even start Scrum!

Without further explanation here, the set of metrics I recommend for initial implementation is:

- Customer and team surveys
- Velocity chart
- Burnup / burndown chart
- Running automated tests
- Technical debt
- Work-in-process
- Story cycle time

And once you are able to put in some financial measure for business value, add the following:

- Cost per sprint or story point
- Real value delivered
- ROI or NPV

space for notes

Coaching

What does a coach do?

> *'Scrum coaching is defined as an engagement with one or more organizations/ teams during which the coach acts as a mentor or facilitator for those teams to improve their understanding and application of Scrum to reach their stated objectives. The engagement includes one-on-one and team mentoring, process facilitation, organizational development, alignment consultation, and interaction with all levels of leadership within an organization. It may also include mentoring in methods related to the effectiveness of Scrum, principles described in the Agile Manifesto, Lean principles, and Extreme Programming practices.'*
>
> *Certified Scrum Coach Application, Scrum Alliance*

The coach guides people in client organisations in the following ways:

- Advisory and consultation to enhance and speed up the self-discovery process
- Facilitation of client adoption, implementation, and learning Scrum
- Agile leadership, based on a servant leadership style
- Organisational development that enhances the client's existing skills, resources, and creativity.

Why does my organisation need coaching?

Software development and other kinds of work performed by knowledge workers relies on *tacit knowledge* as opposed to *formal* or *explicit knowledge*. Tacit knowledge is difficult to transmit from one person to another and is mostly gained through personal experience. An example is learning to ride a bicycle.

The ScrumMaster role must provide process leadership to the rest of the Scrum Team and to the rest of the organisation. Classroom training, such as the Certified ScrumMaster course, is an insufficient means to gain the required tacit knowledge to master Agile practices and the Scrum framework. The team and the organisation need the skills of an experienced practitioner to guide them through the maze of challenges they will inevitably face in transitioning to Agile.

The bottom line

When asked what they would change if they did their transition to Agile and Scrum over, the most common comment heard from managers in organisations worldwide is 'more coaching'!

References

1. Cockburn, Alistair (2007). *Agile Software Development: The Cooperative Game (2nd edition)*. Addison Wesley
2. Cohn, Mike (2004). *User Stories Applied*. Addison Wesley.
3. Cohn, Mike (2006). *Agile Estimating and Planning*. Prentice Hall.
4. Gloger, Boris (2008). *Heartbeat Retrospectives*. http://borisgloger.com/2008/04/24/heart-beat-retrospectives-1-introduction/.
5. Highsmith, Jim, et al (2001). *Manifesto for Agile Software Development*. http://www.agilemanifesto.org/.
6. Hundermark, Peter (2009). *Measuring for Results*. http://www.scrumsense.com/coaching/measuring-for-results.
7. Jeffries, Ron (2001). *Card, Conversation, Confirmation*. http://www.xprogramming.com/xpmag/expCardConversationConfirmation.
8. Katzenberg, Jon R. And Smith, Douglas K. (2002). *The Wisdom of Teams*. Collins.
9. Nonaka, Ikujiro and Takeuchi, Hirotaka (1986). *The New, New Product Development Game*. Harvard Business Review, Jan-Feb 1986.
10. Sliger, Michele and Broderick, Stacia (2008). *The Software Project Manager's Bridge to Agility*. Addison Wesley.
11. Schwaber, Ken (2007). *The Enterprise and Scrum*. Microsoft Press.
12. Schwaber, Ken (2009). *Scrum Guide*. http://www.scrumalliance.com/resources].
13. Teasley, Covi, Krishnan, & Olson (2000). *How Does Radical Collocation Help a Team Succeed?* Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work (pp. 339 - 346). New York: ACM.
14. Wake, William (2003). *INVEST in Good Stories, and SMART Tasks*. http://xp123.com/xplor/xp0308/index.shtml
15. Yip, Jason (2006). *It's Not Just Standing Up: Patterns of Daily Stand-up Meetings*. http://martinfowler.com/articles/itsNotJustStandingUp.html.

About Scrum Sense

Scrum Sense is a small Agile consultancy situated at the southern tip of Africa providing coaching and training to people in organisations who have a burning desire to find better ways to work together.

We offer the following services:

- A complete menu of coaching services that help you succeed in transitioning your organisation to Agile and Scrum
  - Starting Scrum
  - Product Owner skills
  - ScrumMaster skills
  - New responsibilities of managers
  - Enterprise adoption
- Public Certified ScrumMaster (CSM) training courses
- Public Certified Scrum Product Owner (CSPO) training courses
- Private uncertified Scrum training courses for team members (in conjunction with coaching services)
- Private CSM and CSPO courses
- Software development skills training (in association with Factor10)
- Design of collaborative work spaces for collocated teams
- Assessment of Agile practices in organisations
- Agile evangelism talks to any interested audience (free)

ScrumSense

www.scrumsense.com
mobile +27 842613680
skype peterhundermark
twitter peterhundermark