emergn

Value, Flow, Quality®

SCRUM

FLOW

This publication forms part of Value, Flow, Quality® Education. Details of this and other Emergn courses can be obtained from Emergn, 20 Harcourt Street, Dublin, D02 H364, Ireland or Emergn, One International Place, Suite 1400, Boston, MA 02110, USA.

Alternatively, you may visit the Emergn website at http://www.emergn.com/education where you can learn more about the range of courses on offer.

To purchase Emergn's Value, Flow, Quality® courseware visit http://www.valueflowquality.com, or contact us for a brochure - tel. +44 (0)808 189 2043; email valueflowquality@emergn.com

Emergn Ltd.
20 Harcourt Street
Dublin, D02 H364
Ireland

Emergn Inc.
One International Place, Suite 1400
Boston, MA 02110
USA

First published 2011, revised 2013 - printed 3 March 2016 (version 1.7)

Printed and bound in the United Kingdom by Apple Capital Print.

Contents

INTRODUCTION

The Agile Manifesto begins with a bold statement: 'We are uncovering better ways of developing software'. Many of these 'better ways' have spawned their own language, artefacts and associated industries of publishing, training and conferences. One of the most famous and successful is Scrum, which describes itself as a 'framework' for developing and sustaining complex products.

Popularity carries a certain risk – the more people adopt or use an idea, the more likely it is to be mistranslated, misunderstood or misapplied. There's a danger that people try Scrum, find it doesn't quite work for them or live up to its promises and declare the whole concept to be rubbish.

We have seen Scrum introduced into numerous companies and fail ignominiously. In almost all cases, it has been because those implementing the methodology did not understand the whys. This encouraged differing problems: complete rejection or resistance; a slavish copying of processes rather than principles leading to disappointment if the expected benefits failed to materialise; and a half-hearted adoption of some elements by senior management, without an appreciation of which formed the essential foundations and which were decorative. Whichever problem lay at the root of the failure, we are convinced that the solution lies in understanding *why* a practice has been developed. This enables you to make conscious, intelligent choices and to create hybrids tailored to your company's precise needs.

This session provides a thorough, if pared-down, grounding in Scrum. While not precisely a 'how-to' manual, this session provides you with all you need to get started on implementing Scrum. It is intended to be used in conjunction with more detailed sessions on the thinking behind specific principles such as prioritisation, feedback and collaborative team working.

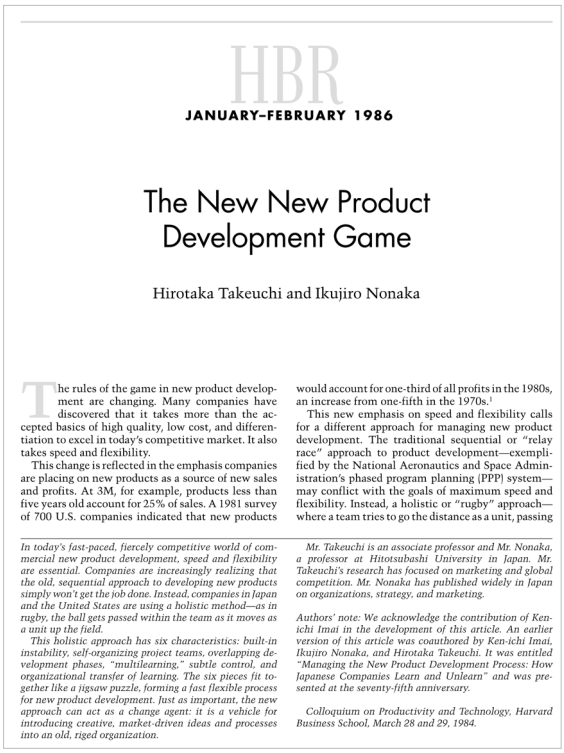
By the end of this session you will:

1. Have an appreciation of the key principles, practices and roles within Scrum.
2. Understand the key advantages and benefits the framework is intended to enable.
3. Identify common points of failure or difficulty and prepare your organisation for a successful implementation of Scrum.
4. Deal with problems that often arise.

1 SCRUM

If you described this methodology to a naming agency (yes, such places do exist) and asked them to come up with a name, we don't think you'd get Scrum.

Rugby was once described as a hooligan's game played by gentlemen. In this hooligan's game, the 'scrum' is the unforgettable spectacle of eight men bent over and pushing the opposing eight in order to gain possession of the ball. This may, or may not, remind you of IT. Nonetheless, it is a fairly niche game in global terms – especially in the USA and Japan, yet that's where the people with the best claim to have developed the Scrum methodology come from.



HBR

JANUARY–FEBRUARY 1986

The New New Product Development Game

Hirotaka Takeuchi and Ikujiro Nonaka

The rules of the game in new product development are changing. Many companies have discovered that it takes more than the accepted basics of high quality, low cost, and differentiation to excel in today's competitive market. It also takes speed and flexibility.

This change is reflected in the emphasis companies are placing on new products as a source of new sales and profits. At 3M, for example, products less than five years old account for 25% of sales. A 1981 survey of 700 U.S. companies indicated that new products would account for one-third of all profits in the 1980s, an increase from one-fifth in the 1970s.[1]

This new emphasis on speed and flexibility calls for a different approach for managing new product development. The traditional sequential or "relay race" approach to product development—exemplified by the National Aeronautics and Space Administration's phased program planning (PPP) system—may conflict with the goals of maximum speed and flexibility. Instead, a holistic or "rugby" approach—where a team tries to go the distance as a unit, passing

In today's fast-paced, fiercely competitive world of commercial new product development, speed and flexibility are essential. Companies are increasingly realizing that the old, sequential approach to developing new products simply won't get the job done. Instead, companies in Japan and the United States are using a holistic method—as in rugby, the ball gets passed within the team as it moves as a unit up the field.

This holistic approach has six characteristics: built-in instability, self-organizing project teams, overlapping development phases, "multilearning," subtle control, and organizational transfer of learning. The six pieces fit together like a jigsaw puzzle, forming a fast flexible process for new product development. Just as important, the new approach can act as a change agent: it is a vehicle for introducing creative, market-driven ideas and processes into an old, riged organization.

Mr. Takeuchi is an associate professor and Mr. Nonaka, a professor at Hitotsubashi University in Japan. Mr. Takeuchi's research has focused on marketing and global competition. Mr. Nonaka has published widely in Japan on organizations, strategy, and marketing.

Authors' note: We acknowledge the contribution of Ken-ichi Imai in the development of this article. An earlier version of this article was coauthored by Ken-ichi Imai, Ikujiro Nonaka, and Hirotaka Takeuchi. It was entitled "Managing the New Product Development Process: How Japanese Companies Learn and Unlearn" and was presented at the seventy-fifth anniversary.

Colloquium on Productivity and Technology, Harvard Business School, March 28 and 29, 1984.

Figure 1. Excerpt from "The New New Product Development Game" by Hirotaka Takeuchi and Ikujiro Nonaka, Jan-Feb 1986.

Hirotaka Takeuchi and Ikujiro Nonaka created a product development method which they called 'rugby' because the team 'tries to go the distance as a unit, passing the ball back and forth'. Ken Schwaber and Jeff Sutherland adapted and developed the idea for software development in the 1990s, giving their extensive series of ideas the name Scrum.

Since those early days a vast quantity has been written about Scrum, as even the briefest internet search will tell you. There are numerous Scrum courses, certifications for various roles, conferences, blogs, websites and forums. Much of this material is very good, a great deal of it goes into far more detail than we consider strictly necessary. We need to begin by boiling down to the essence of Scrum. What is it about, what problem does it intend to solve, and how does it go about it? From that we will go on to describe the framework – a series of practices, activities and roles associated with Scrum.

1.1. What is Scrum?

Scrum, as spelled out by Jeff Sutherland and Ken Schwaber, is neither a process nor a technique for building products. Scrum is positioned as a process framework within which small teams can design and sustain complex products to effectively deliver the highest value. All of which, we think, is a jargon-heavy way of saying that Scrum is designed to help focus the team's attention on flow and value.

You already know the problem. Software development projects often fail. They take too long. They cost too much money. The people who work on them often want to collapse or leave (or both). Requirements and ideas are often vaguely defined at the beginning of the process so that any plans made then, turn out to be meaningless by the end.

Such issues prompted Schwaber and Sutherland with others to try and create a way to deliver working software in a maximum of 30 days. Moreover, they wanted the pace of delivery to be sustainable – this should be a normal working week, not developers working 24 hours in shifts. Given the initial starting point, such an aim seemed (and still seems for many projects) to be almost absurd in its ambition.

Sutherland and Schwaber describe the Scrum framework they created: lightweight, easy to understand and difficult to master. It is composed of a number of specific Scrum roles, practices and 'rules of play'. That may sound worryingly formal, but Scrum's claim to be 'lightweight' is fair. Certainly the rules are easy to understand – we'd expect you to know exactly what you needed to do to implement Scrum in theory by the end of this session.

In practice however, Scrum often proves extremely difficult to implement correctly. It can highlight problems within an organisation and require significant changes to work culture and existing processes if you are to reap the full benefits promised.

Jim Highsmith, Director of the Agile Project Management Practice, suggested three 'readiness tests': will you spend money on change; will you spend time on change; will you remain committed to change?

Case study – Primavera: Why changing to Scrum is hard

In 2003 Primavera Systems – a company which sold portfolio management solutions (now owned by Oracle) – changed to Scrum. They might not have made the change if it weren't that things were going so badly. Sales were fine, but the internal process of developing new releases was appalling. The team were demoralised and exhausted after a final 'death march' on the last project. Everyone had worked weekends and late nights just in order to release something which senior management felt was incomplete (and overdue). In fact, despite the team feeling their efforts had been heroic, management were so disappointed, they decided not to pay the team their annual bonus.

The VP of Development decided that it was the development process that was at fault and that this had to change. Scrum was implemented. By the end of the first release, the team was able to demonstrate a particularly difficult and risky element of the next release. The company was instantly sold on the change, which over the next year proved itself time and again. As well as producing better quality work in a more timely manner, the team became energised and better connected to its own senior management, making the company an enjoyable, positive place to work.

Brilliant – it sounds like a magic wand!

But make no mistake, those at Primavera acknowledged how hard the process had been – because it required a commitment to change throughout the organisation. Change does not happen easily. Funnily enough, the obvious external changes are often the easiest ones to manage: changing to cross-functional teams, changing planning, changing to short feedback loops, changing engineering processes, changing approvals processes – Scrum has developed a framework which assists with these. Scrum has not been able to develop a framework which changes how people think and feel (although it tries). The team could become 'cross-functional', but becoming truly collaborative and co-operative rather than competitive is a much more painful shift. Everyone at Primavera needed to accept the new way of working – marketing and quality people had to embrace the changes as much as developers. The team noted ruefully how often, under stress, everyone reverted to past behaviours, and how quickly success could be forgotten or bogged down in fresh problems.

Perhaps Primavera would not have succeeded if it had not been so conscious of how awful and painful their previous way of doing things was. Sometimes it needs crisis to force a company to overcome inertia. But when they do – as Primavera discovered – the benefits can be extraordinary.

1.2. Why does Scrum work?

Scrum has certain ceremonies, and has even developed its own vocabulary. All of which can sound rather off-putting, as if we were being asked to join a cult or sign up to a political movement. That's why we think it's important to explain that Scrum does not work because it uses 'Sprints' or a 'Scrum Master', but because of certain underlying principles that encourage faster delivery cycles, increased customer responsiveness, ability to change direction and greater predictability or improved risk management.

Empirical process control

Scrum is based on real data. If you agree to create five features this 30 day sprint, by the end you can measure how many you have actually created: if you've created seven, then you can probably take on more the following month; if only two then the project may prove more difficult and take longer than you had anticipated. If you've completed none, then perhaps the project is misdirected and you need to think again. Real data enables good decisions and good estimation. For CEOs who have been used to signing off vast sums of money without knowing what results they will get for two or three years, this can prove one of Scrum's most attractive features.

Feedback and embracing change

Real data does not only refer to numbers (how many features written, how many to go, average burn rate, etc.). Data also incorporates feedback. Because Scrum runs on a short development cycle of 30 days, feedback is frequent. When a piece of working software is demonstrated ideas spark and develop. 'That's great' someone shouts, 'let's add that function to every page!' Suddenly the plan has changed. Instead of groans and complaints that a new idea is not in the plan, Scrum is designed to embrace change. If the new idea is considered more valuable, then it receives priority over other features. Far from needing to have a list of all requirements up front, Scrum can start with just one or two certainties, and then build on them as concepts become firmer.

Similarly, if the piece of software is not doing what the customer wants, then the team can go back and make changes. This short feedback loop not only increases value to the customer, but it encourages quality in the product and, equally importantly, cost effectiveness for your organisation. An investment of only 30 days is relatively low cost compared to a year of development. At the end of the first two sprints, if it has become clear that it's not going to deliver on the vision, you can decide to call a halt to the project.

Iterations

An iteration in mathematics and computing is the process of repeating the same function to get a result. Interestingly, the iteration of a quite simple function can produce complex behaviours and patterns. An iteration suggests a complete function, however small or simple – so rather than splitting a project into planning, development and testing phases, all three processes would happen within a single iteration to produce a piece of working software. In Scrum an iteration is referred to as a Sprint.

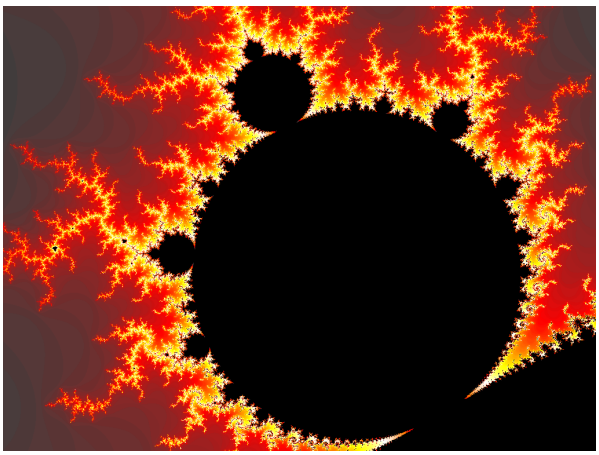


Figure 2. Hundreds of iterations over a simple formula create this complex fractal pattern

Valuable increments

Scrum insists on delivering a working piece of software at the end of each Sprint, which over time can build up into an extensive system. We will discuss a little later why this isn't always easy, but the concept behind it is that each increment should deliver a 'Potentially Shippable Product'. Let's take a very simple example. If we were building an e-commerce website, rather than first creating the entire architecture, stock levels, credit checks, etc., we might begin by launching a single page with information. The next month, customers might be able to see what goods are available, but you might still need to phone or email to order them. Each increment, however small, provides some value to both the organisation and the customer.

1.3. The three pillars of Scrum

We've described the underlying principles that explain why Scrum works. While Scrum is made up of a series of practical tools and practices, it also requires a mental or cultural attitude in order to flourish. Just as telling the work experience student to 'fix the quality problem' is not the same as empowering him, so simply renaming your process Scrum won't automatically produce software that's better, cheaper and faster.

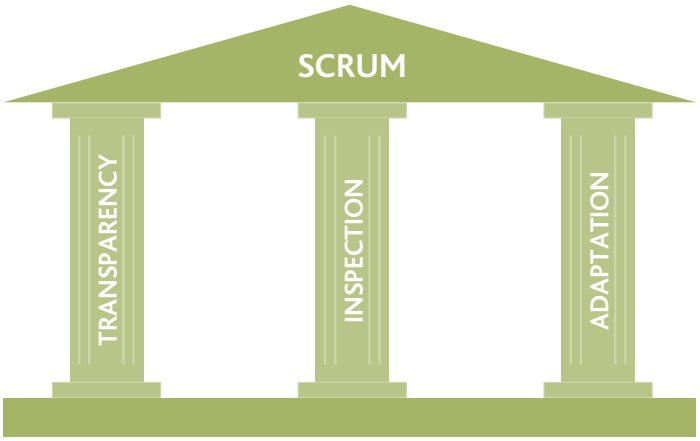


Figure 4. The three pillars of Scrum

Transparency

Everyone needs to understand the process – not just a couple of managers or the developers themselves. The team as a whole is going to be responsible for the outcome and therefore they need to have an overview of the entire process. Part of the reason for spelling out a Scrum way of working is to ensure that everyone has a shared vocabulary and understanding. This doesn't mean, by the way, that everyone has to be a complete convert before you begin. One of the key features of Scrum is that by working in short iterations and then reviewing progress and achievement, people can agree to put their scepticism on hold for just one or two months and then judge actual results.

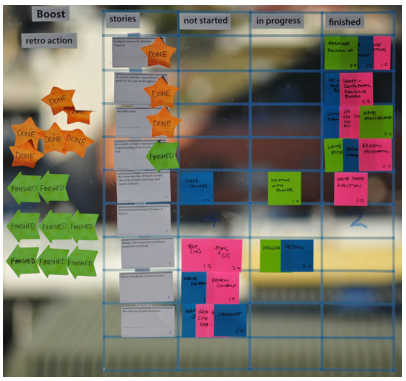


Figure 3. A very visible Scrum Board

Transparency is also part of how the work itself is viewed – there is a shared definition of completion (whether work is done or not done). You can't sneak a piece of work through at the end of the increment by saying – 'well, it's all done bar the testing'. That is 'not done'. Sutherland and Schwaber call partially done work 'opaque' because there is no way of knowing how much work and time will be required to complete it. As teams mature, the definition of 'done' becomes ever more stringent in the search for higher quality. Scrum artefacts must also be transparent – everyone should understand why decisions around prioritisation and risk control have been made. When they are transparent, each decision has a more sound basis, enabling it to be challenged or additional information to be brought to bear on it. The less transparent the decision, the more likely it is to be flawed leading to lower value and increased risk.

Activity 1 – Definition of Done

This activity should take about 20 minutes and is ideally done with a team.

One of the core pillars of Scrum is transparency; transparency about what a team is working on, how they progress and when their work is complete. Many practices in Scrum emphasise making what you're working on visible. In this activity we are going to focus on one aspect called the 'Definition of Done'.

The Definition of Done is a useful technique for identifying a checklist of what needs to be done to every single piece of work, to really know when it is complete. It is created by looking broadly at all aspects of development and capturing what completion means from the point of view of architecture, analysis, coding, testing, integration, etc.

Together with your colleagues consider the units of work that are being worked on in a typical project in your organisation. They may include individual features, use cases or user stories. Look for common elements that need to be considered and completed before a piece of work can be marked as done. For example you may want to consider things like "unit tests are written and passing", "documentation is completed", "source code is peer-reviewed", "required stored procedures are written, tested and approved", etc.

Take some Post-it notes and together write as many sensible criteria for completion as you can think of, each on a separate note.

Once you have at least 10 different items identified, gather all the notes, put them up on a whiteboard and review them together. Remove any duplicates and clarify those which are not obvious to everyone. Using dot voting (every person has 5 dots that they can distribute amongst the Post-it notes however they like) select the top 5 items. Re-write them on to a separate page as a list. This is now your Definition of Done – a checklist that every piece of work completed should be evaluated against. Obviously, put this in a place where it can be visible to everybody and encourage its use, regardless of whether you are currently using the Scrum framework or not.

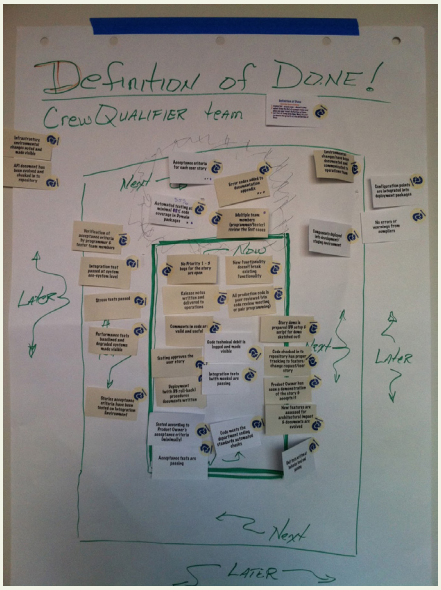


Figure 5. A sample Definition of Done (from an exercise by David Kootnz - Agile Complexification Inverter blog)

Apply your Definition of Done for the next few items you intend to complete and observe what effect it has on the quality of the work being done and on the visibility of what is being completed.

Commentary:

We often observe that using an explicit Definition of Done changes how and when we tell people the work is actually 'done'. It counters the all too common phrase of it's '90% complete' and stops you accidently leaving off crucial activities.

Inspection

Quality, appropriately leads on to the second pillar 'inspection'. But the term is rather misleading. The concept is not to have a dedicated inspector policing the team, checking their work and sniffing unpleasantly every time a bug is caught.

Figure 6. Inspecting new software

Inspection in the context of Scrum is about openly demonstrating the product to customers in order to get feedback. It is also an agreement to take responsibility both for one's own work and for the process itself. Scrum teams do not complain that the customer gave them confused requirements – they work with the customer to clarify, trial and test out ideas. They don't object to stupid managers overloading them with work, because they themselves have agreed what can be achieved and checked that the requirement has been appropriately broken down. Finally, they don't moan about what's not working within the process, because they are committed to the next pillar...

Adaptation

Scrum places a great deal of emphasis on the ability to adapt. In product development terms this means using feedback to refine the product vision, to change direction, or even to completely axe the project. Adaptation is not about massaging estimated numbers in order to get a project through a phase gate, it is all about demonstrating a product and then deciding how to improve it – both of which are functions of a commitment to transparency and inspection.

Figure 7. A Scrum meeting in action

The formalised Scrum meetings are all designed to encourage this behaviour, from the Retrospective where the team asks what worked, what didn't and what can be done better, to the daily question: 'what do I need to do my work today and what might be stopping me getting my work done?'

Activity 2 – Run a Retrospective

One of the key events in which the teams focus on inspection and adaptation in Scrum is called a Retrospective. Scrum teams hold their Retrospectives at regular intervals and focus mainly on the last Sprint. In this activity you will conduct a retrospective to reflect on the work a team has been doing over the last 30 days. It important to note however, that this is not a post-mortem (sometimes known as lessons learnt). We are not interested in lessons learnt, but rather lessons that can be quickly applied over the next 30 days. Again, this activity is not dependent on your organisation having adopted Scrum.

This activity will take 60 minutes and you must run it with a team.

Preparation:

- Pick a team of people with whom you work on daily basis.
- Choose the time and place for your meeting and invite everyone to participate.
- The meeting room will need some whiteboards or flipcharts and should allow for people to work all together and in small groups.
- Prepare Post-it notes, index cards, pens, markers, paper, etc. and bring them to the meeting.

Agenda:

In their book *Agile Retrospectives: Making Good Teams Great*, Esther Derby and Diana Larsen lay out a basic 5-step structure for running Retrospectives. We shall roughly follow this approach here: Setting the Stage, Gathering Data, Generating Insights, Deciding What To Do Next and Closing the Retrospective.

1. Setting the stage (5 minutes)

Explain the purpose of the meeting, agree the timeframe you will focus on in the Retrospective and what theme you may want to choose – for example how the team worked together, how the project work was carried out or technical challenges you might have encountered. This will help focus your thinking.

2. Gather data (20 minutes)

Use an activity called "Mad Sad Glad" to gather data. Draw three columns on a whiteboard or a flipchart and label them "Mad", "Sad", "Glad". Ask everyone to write on Post-it notes significant events over the period that made them feel mad, sad or glad. They should recall what happened and the effect it had on them. Ask everyone to put their notes up as they are writing them. Don't worry about repetitive, vague or overlapping ideas. Gather as many as you can.

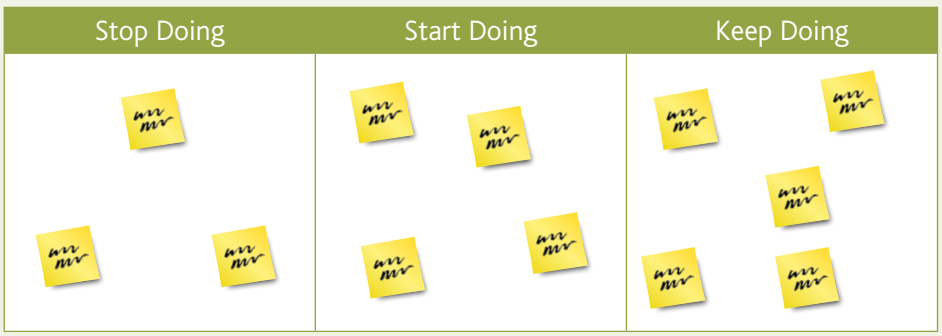
3. Generate insights (15 minutes)

Look at all the reflections and ideas you have collected in the three columns. In each column group the notes together when they relate to the same theme or topic. See what stands out. Are there any common patterns? Are there things that surprised the group?

Now try to dig deeper to understand some of the reasons behind the picture that you have painted. Go through the three columns again and use a technique called "5 Whys" for each of the identified themes, highlights and surprises. Explore the true reasons behind it by asking "Why?" – usually after about 5 times you get to the root cause of an issue.

4. Decide what to do next (15 minutes)

Based on the data you have gathered in step 2, and any further insights that it led you to in step 3, come up with ideas for actions that will help you make improvements. On a flipchart write three columns labelled "Stop Doing", "Start Doing" and "Keep Doing". Ask everyone to come up with ideas (one per sticky note) that would help you improve and that fit in these three categories. Note what didn't work too well and you should consider stopping; what things you missed and should start doing to improve; what works really well for you now and you should keep doing.



Together work through the three columns and pick two most important items from the first two columns as takeaways from the retrospective. Rewrite them, if needed, into actionable steps that the team can commit to. Also keep a list of the items from the third column as a reminder of what works for you.

Discuss how you plan to carry out these actions, in what timeframe and who will take responsibility for ensuring that they do happen. Consider assigning each action to one or two people so that everyone in the team owns some part of the improvement process.

5. Close the Retrospective (5 minutes)

Summarise the meeting and reiterate your agreed next steps. Discuss what you have discovered and how you expect this to help you learn and improve your work in the near future.

Commentary:

Hopefully the hour spent in the Retrospective provided you with some fresh ideas on how to improve your current approach. Teams who regularly run Retrospectives are often able to make sustainable changes to their process. Even if each meeting results in only a few minor tweaks, over a long time period these add up to significant improvements. It is possible to become complacent in the process, so it's important that everyone is committed to the improvement efforts and that any changes or actions proposed are carried out. You can reflect on how they worked in your next Retrospective.

2 THE PRACTICALITIES

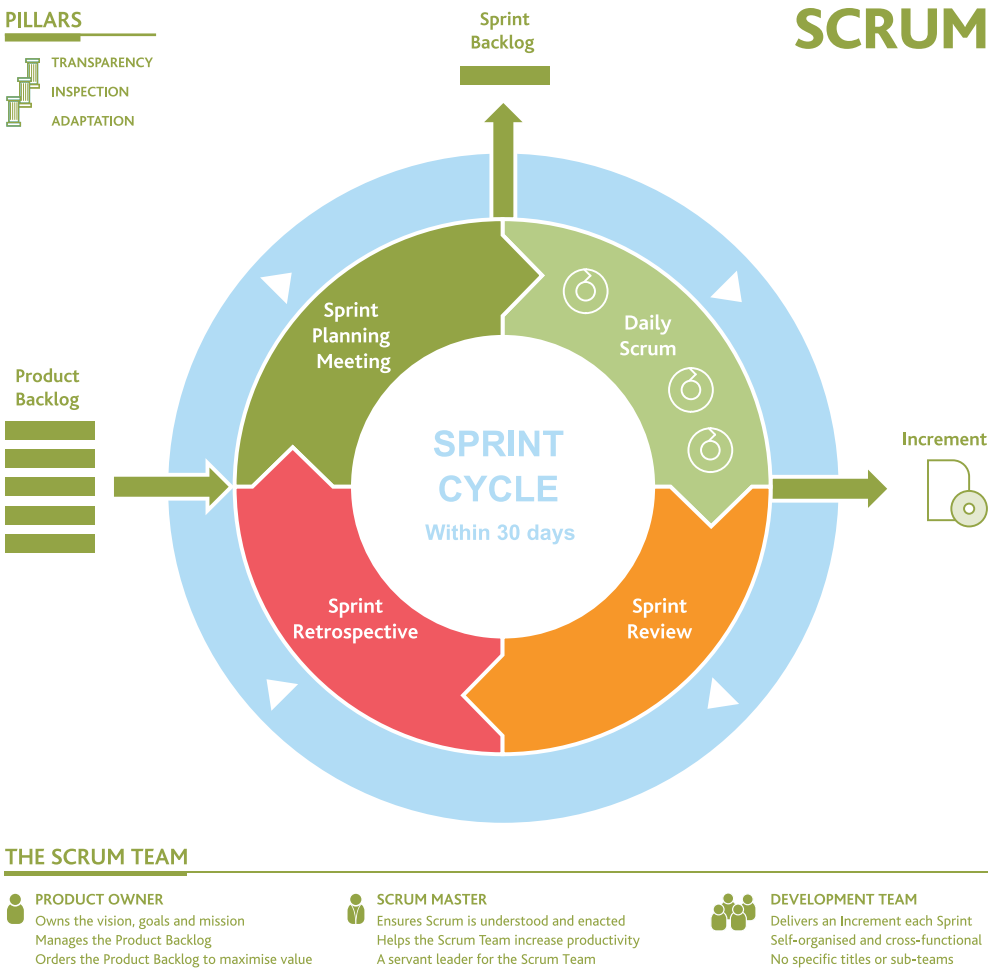


Figure 8. Diagram of Scrum

2.1. The Scrum Team

A Scrum Team relies on size for its agility and speed. It should be small, consisting of a Product Owner, a Scrum Master and between three and nine people in 'development'. The team is self-organising and cross-functional. They should have all the skills necessary to take the product from idea to shippable – that means that our term 'developer' is an elastic one, it might include coders, testers, analysts, database specialists... Their self-organised autonomy must also be genuine. If the team says they can complete three features this sprint, no senior manager is allowed to dictate that the number should rise to six, or try to sneak an extra, pet project into the team's workload.

Product Owner

The Product Owner represents the customer and as such is responsible for maximising the value of the product. To do this, he or she has the final authority to decide which requirements are the most important, the detail of each requirement and to judge whether the results have been delivered.

Scrum does not provide tools to help the Product Owner do this. It's one of the reasons Scrum is not a magic wand. A Product Owner can make the wrong decision: he might decide a feature is essential which turns out to be unimportant; he might accept a piece of work that has unexpected flaws; his prediction of users' actual behaviour could be mistaken... The point is that by delivering a working increment, such risk is managed, because the increment can be deployed and tested with real customers.

The Product Owner is not an easy role. Jeff Sutherland once wrote out his wish list for a Product Owner working in a healthcare software provider and it reads rather like a spoilt child's Christmas list.

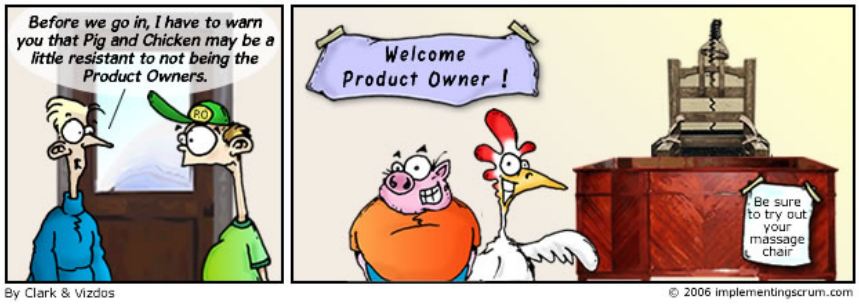


Figure 9. Shock Treatment for your Product Owner from Implementing Scrum

Even the best candidate can take some time to get used to such a role, but here are some of the most common issues to bear in mind:

- The Product Owner is responsible for the Product Backlog and the continual ordering of the features within it – it can't be done at the beginning and then left until the project is finished.
- The Product Owner may work across a couple of projects, but she must always be available to the team – especially during Sprint Planning Meetings and the Sprint Review Meeting. If the team needs to wait for a response from the Product Owner when seeking clarification over requirements then the capacity of the team to deliver the increment is compromised.
- The Product Owner does not manage the team. The team is self-organising and it is not for the Product Owner to decide who does what or how much they should do.

- The Product Owner cannot add more work when the Sprint is in progress, although the team may pull more if they wish to.
- As a single person representing the customer, the Product Owner is often faced with the hard choices required by balancing competing interests. The important point is that the Product Owner must make those hard choices during the planning meeting and then review whether they were correct or not afterwards.

Development Team

The Development Team are responsible for building the increment. As we discussed above, its members need to possess all the skills required within the team. Rather than giving each member a label – 'programmer,' 'tester', 'architect' – everyone is simply a 'developer'. They need to take shared responsibility and hold one another accountable for their progress towards the shared goal of that specific increment and for the project as a whole. For this to work effectively, the team should be co-located and sized between three and nine people. Team members need to be willing to take on work outside their specialised knowledge. Someone who has worked as a tester may indeed end up doing lots of testing, but if that is not required in a particular Sprint, then he should be prepared to turn his hand to other work. For some individuals, such a way of working is creative and inspirational, others may need coaching and help. Our session on Teams discusses this further.

Scrum Master

Wow – what a great title it is – reminiscent of Masters of the Universe, perhaps... But before you get too excited and sign up for a certification course, we should point out that the Scrum Master's title refers to his or her authority over the Scrum process, not over the team or the work. The Scrum Master is more like a coach or personal trainer, helping the team in the Sprint but not actually running the race himself. Indeed, Sutherland and Schwaber also rather less glamorously refer to the Scrum Master as the team's 'servant leader'.

The Scrum Master ensures that the Scrum values, practices and rules are followed. He organises and facilitates the meetings, helps deal with impediments or blocks to progress and may even assist in coaching to encourage cross-functional working. He or she helps the Product Owner work on long-term planning and on the kinds of changes of direction following feedback that are an essential part of agility.

Hmmm... If there's one role that companies newly adopting Scrum are most sceptical about, this is the one. It sounds like an increase in headcount and all to do something that isn't adding direct value. What often happens is that rather than seriously recruiting for someone with experience of working with Scrum, organisations pick a project manager and send them on a 2-day course. This is rarely sufficient and indeed lays an unfair responsibility for the success or failure of the implementation on one person.

Remember the part about how hard it can be to implement Scrum effectively? The Scrum Master is the answer to many of the problems. By creating strong discipline

around a framework that has proven results in many organisations, they increase its chance of success in your own. A Product Owner who misses one Sprint Review, or who adds some extra work into the Sprint 'just this once'; a Sprint that lasts 33 days instead of 30; a specialist who insists on being treated differently to the rest of the team... none of these may sound like sufficiently serious infractions to derail the project, but each is symptomatic of an organisation that wants the results without being able to commit to the principles behind Scrum. A Scrum Master who really understands the principles and acts as an intelligent advocate for the rules, is thus hugely advantageous.

2.2. Artefacts

Artefacts sound rather like objects from a Dungeons and Dragons game... in the context of IT methodologies, we might call them the 'things' that we produce or require.

Product Backlog

In its simplest form, a Product Backlog could be just a list of all the ideas that make up our vision of the product we would like to build. These might range from well-defined features, (a log-in function), to rather vague concepts, (something that makes the customer feel reassured when they put in card details). This list of ideas must be ordered – the most important and valuable coming at the top, the less important further down. Those at the top tend to be more detailed, well-defined and with a clearer idea of what each will achieve, those further down may still be quite vague. Each item on the list has a description, an order and also an estimate attached to it. While the ordering is the responsibility of the Product Owner, the development team must assign the estimate. This provides an idea of the amount of work it will take to develop the idea. Since only work items estimated to take 30 days or fewer can enter the Sprint, it is helpful because it can show you when an idea needs to be broken down into smaller, more manageable chunks.

Figure 10. An example of a Product Backlog

The Product Owner needs to be constantly checking the backlog – is everything in it still important? Have we changed our ideas? Have features like this one proved so tricky that I should discuss changing the estimate with the team? Are there developments in the market that mean a particular feature is now more important? Have we reached the end of what is really going to deliver value?

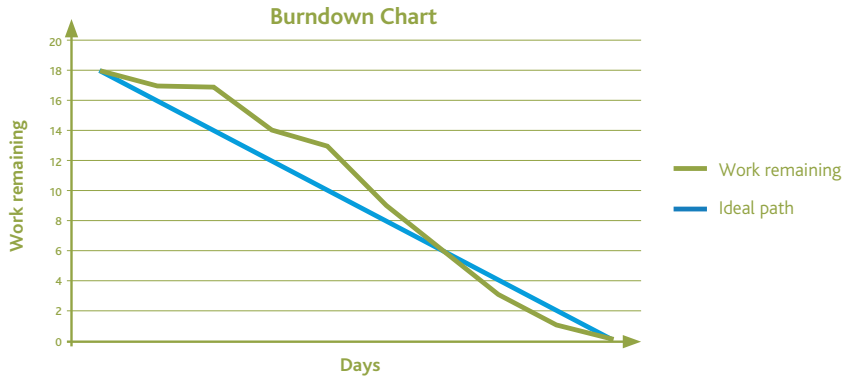


Figure 11. An example of a Burndown Chart

Sometimes the Backlog is supported by a release Burndown Chart which simply shows the total number of items in the backlog left to be completed. Of course, occasionally more items are put into the backlog between Sprints as new ideas are generated from customer feedback. The team can also predict when they will finish the project based on their previous results. The only issue with a Burndown Chart is that it can discourage an openness to change and adaptation. If the Product Owner feels they have delivered the majority of the value, it may be time to jettison the remaining items. A neat graph on the wall showing the number still to go until the finishing line can act as a psychological barrier to such decisions.

Increment

Increment

An increment is the amount of work delivered during a given Sprint. Each builds on the one before, which means that all increments must be fully integrated. There is no such thing in Scrum as a long period of 'stabilisation' at the end of a project where software is tested and integrated. This is all part of the Sprint, ensuring that each piece of working software is a potentially shippable unit. The Product Owner can, and indeed often does, choose to release an increment immediately following the Sprint, or she may wait until several increments have stacked up together. Read our session on Delivering Early and Often for a more in-depth discussion.

Sprint Backlog

Sprint Backlog

This is pretty simple – if the Product Backlog is the list of ideas and items that compose the entire project vision, the Sprint Backlog is the list of items that have been accepted for that Sprint in order to create an increment. Just as items in the Product Backlog have a description and an estimate, so the Sprint Backlog items have a plan for delivering the work with sufficient detail to ensure that progress can be understood on a daily basis. This means that the team don't congratulate themselves on finishing five out of the six items, only to suddenly learn that item six was a whopper!

The Development Team is responsible for managing the Sprint Backlog and refining the plan as they tackle the work. An important element of Scrum is that the work is visible to everyone – the whole team knows what is in the Sprint Backlog, which assists in self-organisation.

2.3. Events

We have mentioned several meetings and events already. In Scrum these happen regularly. This minimises disruption and reduces the need for ad-hoc meetings, which can waste development time. All Scrum events have a set maximum duration, which is commonly known as 'time-boxing'. The overarching rhythm of Scrum is set by the Sprint – all other activities happen within this beat.

The Sprint

The Sprint is the name given to each iteration designed to produce an increment. It cannot be longer than 30 days, although some Scrum teams work in 2-week Sprints. In traditional Scrum the duration of Sprints must be consistent – that is, all sprints are 30 days, or all sprints are 14 days.

There is no gap between Sprints – no need for the developers to take a one week break, for example. The whole point of Scrum is to maintain a sustainable pace that produces quality and eliminates the wasteful activities that misdirect and exhaust the development team's energies. To support this aim, Scrum insists: the team composition must remain constant for the Sprint and every Sprint must have a goal, a clearly defined vision of what that specific Sprint is going to deliver, which cannot be changed once the Sprint has begun. This offers coherence for the team's work during the Sprint.

These rules can sometimes sound rather inflexible, but anyone who has worked on a team where members are being pulled off to deal with sudden crises knows how disruptive and damaging it is. Similarly, changes of strategy at mid-point mean that nothing will get built – often more of a waste of time than learning from a misdirected effort and refining it.

Scrum does emphasise that scope can be clarified and renegotiated during the Sprint, which normally means an adjustment of what can be achieved as estimates are replaced by reality.

Time limit: 30 days

Sprint Planning Meeting

This is where the work that will form the Sprint Backlog is selected and planned. The meeting is a collaborative one using the expertise of all parties. Indeed, the Development Team may invite others to attend to provide necessary domain or expert technical advice. It addresses two topics: what can be done and how will the chosen work be done.

In the first part of the meeting, the team decides what will be delivered. This is not the kind of stubborn negotiation that many waterfall managers know only too well, where the Product Owner insists the team must complete a certain amount of functionality and the Development Team refuse to take it on. Instead the team forecasts what kind of functionality could be created from a given number of items in the Product Backlog.

They consider the last increment (which is their foundation for this Sprint), take into account their past performance and thus agree a likely capacity for what they can achieve this Sprint. From this they craft the Sprint Goal.

The second part of the meeting is devoted to how the work will be done. This is not a technical description of patterns of code, but rather a system of work and process. The planning becomes detailed here, separating the initial pieces of work into units of one day or less.

Time limit: 8 hours in total

Daily Scrum

Every day, the Development Team assembles for a stand-up meeting – usually in front of a work in progress task board (also called the Scrum board) – in order to make sure they are working together, assess how they are doing and plan the day's work.

To minimise disruption, it is important that the meeting is kept strictly to time and happens at the same time and place every day. It is not a status update and no-one other than the Development Team is permitted to participate.

During the meeting each Development Team member explains:

- What I did yesterday
- What I will do today
- What obstacles are in my way.

The meeting avoids duplication of effort, or the risk that any item will be left undone. It also forces the team to articulate anything that is missing and act on it, as well as encouraging cross-functional working or swarming on any particularly intractable problem.

Figure 12. A daily Scrum in action

Recently, the Scrum Guide has rephrased these questions to offer a greater emphasis on team. The questions are now - officially - What did I do yesterday that helped the Development Team meet the Sprint Goal; what will I do today to help the Development Team meet the Sprint Goal, and do I see any impediment that prevents me or the Development Team from meeting the Sprint Goal?

If you feel that these questions have lost the sense of jargon-free, everyday language, then we agree. Reminding everyone that they should be focused on the team and the Sprint Goal is helpful, insisting on an unnaturally formal phrasing is probably not. Unsuprisingly, most companies stick to the old version.

Time limit: 15 minutes

Sprint Review

At the end of the Sprint the full Scrum Team comes together to inspect the increment produced. The meeting is informal and intended to elicit high quality feedback. It is neither a pitch in which the team attempt to dazzle a critical Product Owner, nor a party at which everyone congratulates themselves on what a great job they did.

The Development Team needs to demonstrate the work that's been produced and the Product Owner needs to identify what has been 'done' or 'not-done', give feedback (along with that of any other stakeholders) and consider the remaining Product Backlog. The feedback and discussion that follows will form much of the input into the next Sprint Planning Meeting. There can be significant changes of direction or new ideas generated from these meetings which may have a large impact on the rest of the Product Backlog.

The meeting should emphasise what valuable work has been delivered, and how to optimise this for the next Sprint. The key point is to emphasise that making decisions about value is collaborative - not something for the Product Owner to simply pronounce on alone, even though the Product Owner remains responsible for the final decision.

Time limit: 4 hours

Sprint Retrospective

The Scrum Team meets to discuss how the process and team functioned and what improvements can be made. While the discussion may include issues with tools and definitions, it should not be a technical discussion regarding the work itself (which has already been dealt with in the Sprint Review). For example, a common question that is discussed in the Sprint Retrospective is how to improve the quality of the software – what checks and tools might help or how to unblock developers more easily when they encounter an obstacle. The meeting ends with a plan for improvements. Naturally how these improvements worked or didn't work will be discussed at the next Retrospective.

Time limit: 3 hours

Duration of events with shorter Sprints

The length of each meeting reduces accordingly for shorter sprints. The table below shows examples of the length of each meeting for 1-month and 2-week Sprints:

Sprint length	1 month	2 weeks
Sprint Planning Meeting	8 hours	4 hours
Sprint Review	4 hours	2 hours
Sprint Retrospective	3 hours	1.5 hours
Daily Scrum	15 mins	15 mins

Figure 13. Length of each meeting related to Sprint length

Essentially, in a 2-week Sprint you will do less work than in a 1-month Sprint and thus there should be less to discuss.

Activity 3 – The 59 minute Scrum

You will need an hour to run this activity and a team of 4 to 6 people.

Now that you have read about the key elements of Scrum it's time to try it out in action. This activity is a short simulation of the Scrum lifecycle. It provides an opportunity to experience building a product starting with a Sprint Planning Meeting, running a 2-day Sprint, Daily Scrum and doing the Sprint Review.

Goal: Deliver a tourist brochure for Martians visiting Earth.

Preparation:

- Gather stationery that might be useful for building a brochure: pens, cardboard, coloured cards or paper, old magazines, scissors, glue, etc.
- If people are not familiar with the Scrum framework take a few minutes to explain to everyone in the team how Scrum works, what the different roles, artefacts and activities are.
- With your team decide who is going to be the Product Owner and who will play the role of the Scrum Master. Others will represent the Development Team.
- Print out the "Martians Visiting Earth – Tourist Brochure Backlog" which can be found in the Additional Resources area of the VFQ website.
- Ask the Product Owner to order the Product Backlog based on his preference. This might be very random – that's fine.
- Get a timer and make sure to stick to the times indicated when running this activity.

Play:

1. Sprint Planning Meeting (10 minutes)

From the Product Backlog the Development Team should select the items they think can be achieved during a Sprint considering the order provided by the Product Owner. The whole team writes a Sprint Goal together.

The Development Team decides how the forecasted work will be delivered. They create a Sprint Backlog – a plan containing tasks that must be completed to achieve the agreed increment. You may want to estimate your tasks to ensure that they are achievable in the 14 minutes available.

2. Sprint – Day 1 (7 minutes)

The Development Team works on the Product Backlog guided by the Sprint Backlog and adjust where necessary. Stop working when the time is up.

3. Daily Scrum (5 minutes)

Stop and reflect of the first part of your Sprint. Gather the Development Team together; each member answers the three questions to coordinate the work (What was accomplished on the first day? What will be done in the second day? What obstacles are in the way?). Adapt your plan for the second day of the Sprint to meet the Sprint Goal.

4. Sprint – Day 2 (7 minutes)

The Development Team continues to work on the Product Backlog.

5. Sprint Review (10 minutes)

Your Sprint is now over and it's time to inspect the increment that you have delivered. Look at your initial Sprint Backlog, see what tasks have been completed and what, if anything, didn't make it. The Development Team should then demonstrate to the Product Owner the brochure that has been created and seek feedback.

6. Sprint Retrospective (10 minutes)

Now reflect on your Sprint, consider what worked well and what didn't work. Talk about what would you have done differently if you were to do another Sprint.

Activity review:

What was the outcome of your simulation? Was it easy to follow the Scrum process? Did you achieve the product increment planned? What feedback did you get from your Product Owner? What were the key problems you ran into?

2.4. Activities

Refining the Product Backlog

You may have heard of this activity as 'grooming the backlog' which brought to mind a large and unkempt Afghan hound with a penchant for rolling around under bushes. Sadly, the phrase has now changed to the less evocative, although perhaps more relevant, 'refining the backlog'.

Figure 14. No more grooming the Afghan hound.

From removing items which no longer seem to be valuable to adding new ideas sparked from the last Sprint Review or re-ordering the items based on new information, the process of refinement can require difficult decisions. The whole team is required to collaborate on doing this, but only the Product Owner can update the backlog itself.

Items which have been refined should be well-understood and sufficiently granular to be pulled into the next Sprint. Such items are known as 'ready'.

There is no formal meeting described where this takes place. The team needs to come to its own decision on whether it needs a formal meeting held regularly or whether updating the Product Backlog is something that can be done continuously. As a general guide we advise spending no more than one day every two weeks on such activity.

3 INTRODUCING SCRUM TO YOUR ORGANISATION

We have mentioned that we have seen Scrum implementations fail. Scrum does not tell you how to improve engineering and it does not solve problems within organisations or teams. Instead, its focus on self-organisation and a strict series of rules (time-boxing, etc.) only works with commitment from many different stakeholders. The more heavily 'waterfall' your organisation's approach has been, the more difficult it can be to successfully make the necessary changes. Nonetheless, the results that Scrum has achieved in many different companies are real and highly attractive. As one of the best-known of the Agile methods, it is also likely to have a certain amount of buy-in based simply on familiarity with the idea.

Scrum can be introduced at an organisation-wide level where the organisation is convinced that existing processes are not working. There are several examples where necessity has forced through a Scrum adoption. When Iron Mountain agreed a deal with Microsoft with a contractual go-live date for 6-months hence, the team knew that they would be incapable of delivering the project using traditional waterfall processes. They had no choice but to swap certain failure for uncertain success. Despite the difficulties of disparate teams and various issues that needed to be worked on throughout the project (upgrading the engineering environment, providing training, etc.), the approach succeeded, delivering on the date and generating further opportunities for the company. Such a plunge can be a difficult and risky way to manage major change – Schwaber and Sutherland describe watching organisations 'in upheaval, working in controlled chaos for several years'.

Scrum can also be adopted organically, team by team, either because teams implement the framework themselves (entertainingly called 'Stealth Scrum' by Schwaber and Sutherland), or because the organisation has trialled a 'pilot' scheme. This is obviously lower risk, but for it to work the pilot team needs to be relatively autonomous. Heavy dependencies on other non-agile teams will make the work much harder, while the team also needs to be co-located, free from interruptions from other work and given a clear objective with appropriate support. A spin-off or trial product makes the perfect independent test case.

3.1. Creating fertile ground

Key steps to encourage success in Scrum implementation:

- **Co-locate your team** – while there are examples of Scrum working well across different countries for at least the first sprint, make your life easier by helping the team to gel by co-locating them.
- **Provide start-up training and/or coaching for the whole team** – transparency requires that the whole team understand the process and share a vocabulary. Don't rely on just one or two people to act as 'coaches' for everyone else. IT puts unfair pressure on them and is the opposite to 'transparent'.
- **Look for 'typical' developers who are happy to work on the project full-time** – rather than creaming off the best developers or dooming the project to failure by selecting your most inexperienced people, you need a typical representation of your organisation. A volunteer will probably commit to the team better than someone who has been reluctantly drafted in. You also, of course, need to ensure you have the right mix of skills for the project. Don't underestimate the power of self-organisation where team members figure out which skills they are missing and recruit appropriately.
- **Create a realistic, clear goal about what you want to achieve** – especially in the initial implementation, crafting your goal is essential to dealing with some of most likely problems.
- **Ensure an appropriate level of buy-in from management** – if senior management are deeply suspicious of the process and not committed to its success, the project is handicapped from the beginning. By drawing a clear demarcation of the pilot's limitations and goal you may be able to reassure senior managers regarding the maximum investment and risk.
- **Remember the three pillars of Scrum** – transparency, inspection, adaptation. Apply them to your own set-up of the project as much as to the project itself.

3.2. Dealing with the stones, weeds and tares of infertile ground

These are some of the most likely problems that you will encounter or that will prejudice the success of Scrum:

- **Time commitment** – if your Development Team have specific skills that are in demand from other teams, it may be difficult to ring-fence their time for the Scrum project. Strong discipline is necessary to protect the team from interruptions.
- **Over-ambitious expectations** – business leaders may have unrealistic expectations from just one iteration. It must be emphasised that experimentation, exploration and learning have an important role.
- **Lack of delivery** – the iteration may have produced very little functionality. There may have been heavy up-front work required on architecture or technology, or the Development Team may not be up to the role and need training or replacing. Whatever the problem, the solution is to be more transparent, not less. Ensure that you are measuring results and that you share these results with management and other teams.
- **Misdirected results** – the Product Owner may not have explained his requirements sufficiently well, or the team may have misunderstood them. In many ways this is positive because confusion has been discovered after a small investment of time, but the team needs to make the appropriate changes in the next iteration.
- **Be prepared to adapt** – Your initial implementation may need adaptation according to your specific context and requirements. Be prepared to make these changes (although you probably want to give each change a chance to bed in). Read our Adapting Agile session for more details.
- **The process was difficult** – team members who have not worked together or in this way before may be struggling with the process itself. Occasionally the team composition may need to be changed, but normally help from a coach or discussion about the process itself ought to be sufficient.

Activity 4 – Impediments

This activity should take 20 minutes of planning and should be followed up by implementing any change required.

In the last two sections we have suggested some conditions favourable to a Scrum implementation and some problems/impediments you may encounter trying to implement it.

Impediments are a hindrance or obstruction that get in the way of doing something. Scrum likes to expose impediments and quickly deal with them. When you are first implementing a faster delivery process you will find many impediments that may stop Scrum from operating as intended.

We can try to identify the key obstacles to a successful Scrum implementation for your organisation in advance to minimise the resistance to change.

On your own or with a few colleagues, identify the top 3 impediments that you might expect if you decide to adopt the Scrum framework. Evaluate them, identifying why you believe each one would cause issues.

Pick one that you think is the most important or problematic. How could you tackle it? How could you expose the issue to others? How could you remove the impediment quickly?

Prepare a plan of action, focusing on what can be achieved in small steps of no more than 30 days. With the help of your colleagues try to solve the problem you have identified.

Commentary:

This may feel like a very small step, but the problem you have identified will probably be an obstacle to any form of faster delivery, not just Scrum. What you learn in solving this issue, and the observations you make of the wider team, will prove helpful when you decide to implement any changes to the way you work.

4 SUMMARISING SCRUM

The benefits of Scrum as propounded by its adherents are many. Without wishing to sound like missionaries, it's certainly true that Scrum's principles help contribute towards faster cycle time.

In essence, using time as the constraint means that scope is the element that changes. Because so much of what we think we need turns out to be unnecessary (the Standish Group issued a report suggesting that 45% of features in software were never used and this is fairly typical), reducing scope is often an extremely effective way to work. Fast feedback also reduces the risk of wasting lots of time developing features no-one wants. There are clear limits, however. The Product Owner is only a representative or proxy customer. Any single individual can be wrong, no matter how in touch with the market or business she may be. Nothing makes up for real feedback. Scrum delivers best on its claims when increments are being released to real customers.

Case study – BBC using Scrum

BBC's New Media division was characterised by uncertainty and emergent software process, lots of different teams who all had their own processes, and development cycles that were taking too long as teams failed to collaborate. In an attempt to pin things down and avoid change: 'the teams were spending more time managing the paper than the product', or in other words, they were creating documentation, not working software. As an internal department delivering to the BBC, the department was confused about who the real customer was. In 2003, they decided to use Scrum to more effectively deliver software.

The key element for the team was being able to create project backlogs written in non-technical language, which others within the BBC could understand and therefore appreciate and sign on to. When Scrum began proving itself through delivering working software in each iteration, other stakeholders were able to accept that they wouldn't get 100% of their requirements, but that early delivery meant that what they did get would meet their expectations. Several high-profile projects, including the local 'Postcoder' project, were delivered successfully through Scrum. The self-organising principle of Scrum assisted with previously intractable problems. They focused a dedicated team member on support tasks including system switching. They brought especially demanding customers into the team and asked their help in prioritisation, using non-techinical terms. Such changes meant that the team was no longer seen as a barrier.

The implementation began with a 'guerrilla style' implementation and then rolled out by adoption. It was not all plain sailing – it took several years, and success led to new problems. The successful Scrum teams began to shut out other teams who weren't prepared to align with their process. It meant that senior management had to get involved to tackle cultural issues more widely and insist on communication between software teams with other disciplines (design, editorial, etc.). As Andrew Scotland, Head of Development at the time admitted, Scrum got the teams working well but threw up all the problems of the business into sharp relief – communication, prioritisation, planning. Tackling these was a continual process – as iterative as Scrum itself – and included adapting and changing Scrum to suit different teams' individual needs, and combining the methodology with others, including traditional project management techniques and engineering processes like XP.

By keeping the planning horizon relatively near, Scrum makes some elements far more predictable. 'When will you deliver a new database?' is a difficult question to answer. 'When will you change the address field?' is much easier because the work is much smaller. If you still want to build the entire new system, however, while planning en route will be better, the overall plan will remain unpredictable in the long term. Nor will a time-boxed approach remove all risk – the market could change completely, technology can fail, the company can go bankrupt...

To implement Scrum you need to be able to split work into manageable units. There is very little within Scrum to help you know how to do this, and it is not easy because many items have large numbers of valuable dependencies. While it's true that any task can be broken down in the sense of 'first do X, then do Y and finish off with Z', none of this helps you if what you need is the end result. We admit it is difficult, but the difficulty can be over-emphasised. When a team is really determined to find ways of delivering smaller increments, you will find it figures out creative solutions or simply releases what feels like daringly pared-down functionality, which turns out to be acceptable. More acceptable, at any rate, than the alternative, which is to deliver full functionality very late (and even then, often of a low quality).

Activity 5 – Implementing Scrum

This extended activity should take 7 weeks and requires a real working team.

In Activity 3, you ran a simulation which showed all the Scrum components at work, but to really understand how Scrum works there's no better alternative than actually using it on a real project.

This activity will help you experiment with Scrum in a team of your choosing and allow you to get a feel for the benefits you can derive and also the issues you may face in its adoption, in a safe environment.

Preparation (1 week):

You will need to select a team that is prepared to take part in the experiment. There are many factors to be considered when choosing, most of which depend on your specific context. The best approach is to discuss it with others - start by identifying projects that are in flight, or about to start, look at the type of work they will be undertaking, consider the risks involved and finally talk to the teams - are they prepared to try a different way of working?

Run the Sprints (3 x 2-week Sprints):

Once you have agreed which team you will trial Scrum in, introduce the Scrum framework to them (even if some of the people have already been exposed to Scrum) to ensure everyone has the same understanding. Instead of the textbook 30 day Sprints, in this activity we'll make our Sprints 2 weeks long. In order to generate meaningful data for the experiment, you'll need to run it for at least 3 Sprints. Ensure that you keep all of the data that is created throughout the experiment - for example, keep records of the number of stories that the team takes on in each Sprint and how many get completed, the data and insights generated within each of the Retrospectives and anything else you think is worthy of note.

Final Retrospective (1 hour):

At the end of your experiment, review the data that you have gathered with the team that participated. What have you learnt by running the experiment? Facilitate an open discussion with your colleagues.

Commentary:

We've said that Scrum implementation sometimes fails. Actually, the most common reason for failure is because it doesn't get implemented at all.

Avoid this by taking steps straightaway!

Set yourself up to succeed:

- Build on the interest you have generated with colleagues with whom you ran the various activities throughout this session.
- Take a leaf out of Google's way of working – ask teams or individuals interested in a pilot to get in touch with you.
- Discuss the ideas with senior management – their support will prove crucial if and when you encounter resistance to change.
- Be clear that this is a pilot – asking people to invest 7 weeks in trying a new way of working tends to be easier than imposing radical change.
- Don't get hung up on names. If people are resistant to Scrum (perhaps because they've tried it before) then there's no need to use the vocabulary. What matters is breaking down requirements, deciding on priority and producing an increment to demonstrate within a short period of time. Presented like this, you may find it easier to gain buy-in.

5 CONCLUSION

Those who watch rugby will know that scrums have a disconcerting habit of collapsing as the teams drive forwards. But when one team is scrummaging well, the eight men, tightly bound together, utterly reliant on one another for direction, stability and force, and weighing on average a total of 827 kilos (1,824 pounds) is pretty well unstoppable. As a software development framework, Scrum relies above all on the power of the team for success. Maybe it's not such an inappropriate name after all.

Learning outcomes

Now that you have completed this session, you will be able to:

Have an appreciation of the key principles that lie behind Scrum

- A focus on empirical data
- A short feedback loop
- Increments delivered through iterations
- A culture of transparency, inspection and adaptation

Understand the key advantages and benefits the framework is intended to enable

- A sustainable pace at which teams can deliver projects with greater predictability
- Early delivery of most valuable items
- Greater flexibility in adapting to changing requirements
- Improved risk management through lower investment of time to deliver working software
- Higher quality as teams take responsibility and integrate with earlier increments

Confidently describe the practices and roles within Scrum

- Product Owner: representing the customer
- Development team: taking shared responsibility for goals
- Scrum Master: facilitator and advocate for Scrum values and practices, sometimes referred to as 'servant-leader'
- Artefacts: Product Backlog, Increment and Sprint Backlog
- Events: Sprint, Sprint Planning Meeting, Daily Scrum, Sprint Review, Sprint Retrospective
- Activities: Refining the Product Backlog

Prepare your organisation for a successful implementation of Scrum

- Co-locate the team
- Provide start-up training/coaching
- Begin with typical and willing developers
- Create realistic goals
- Ensure buy-in from managers

Identify common points of failure or difficulty

- Insufficient time commitment
- Over-ambitious expectations
- Lack of immediate results
- Misdirected results
- Not being prepared to adapt
- Difficulty with the process

BIBLIOGRAPHY

Chandrasekariah, K. 2006. *The First Standup*. [photo] Available at: <http://www.flickr.com/photos/karthikc/333796551>. [Accessed 03 June 2012].

Clark, T., Vizdos, M., 2006. *Shock Treatment for your Product Owner*. [cartoon] Available at: <http://www.implementingscrum.com/2006/10/30/shock-treatment-for-your-product-owner/>. [Accessed 26 May 2012].

Derby, E., Larsen, D., 2006. *Agile Retrospectives: Making Good Teams Great*. Pragmatic Bookshelf.

Hirotaka, T., Nonaka, I., 1986. *The New New Product Development Game*. [online] Available at: <http://hbr.org/1986/01/the-new-new-product-development-game/ar/1>. [Accessed 8 October 2012].

Scrum.org. Scrum: *The Rules of the Game*. [online] Available at: <http://www.scrum.org/scrumguides/>. [Accessed 26 May 2012].

Schwaber, K., Martin, R. C., 2004. *Primavera White Paper*. [online] Available at <http://www.scribd.com/doc/38226/Primavera-White-Paper>. [Accessed 30 May 2012].

Schwaber, K., Sutherland, J., 2012. *Software in 30 Days*. John Wiley & Sons.

Scotland, A., 2006. *Scrum Boosts Effectiveness at the BBC*. [online] Available at <http://www.infoq.com/presentations/Scrum-bbc-newmedia>. [Accessed 30 May 2012].

Wikipedia. *The 5 Whys*. [online] Available at: <http://en.wikipedia.org/wiki/5_Whys>. [Accessed 8 October 2012].

Emergn Ltd.
40 Bank St, Level 18
Canary Wharf, London E14 5NR
UK/EMEA: +44 (0)808 189 2043

Emergn Ltd.
20 Harcourt Street
Dublin D02 H364, Ireland
IRE: +353 (0)1 554 7874

Emergn Inc.
One International Place
Suite 1400, Boston, MA 02110
North America Toll Free: +1 888 470 0576

The Chartered Institute for IT

www.emergn.com

www.valueflowquality.com