

SCRUM PROFESSIONALS



SCRUM MASTER CERTIFIED PROFESSIONAL (SMCP)

STEP UP WITH AGILE EDUCATION

www.scrumprofessionals.org



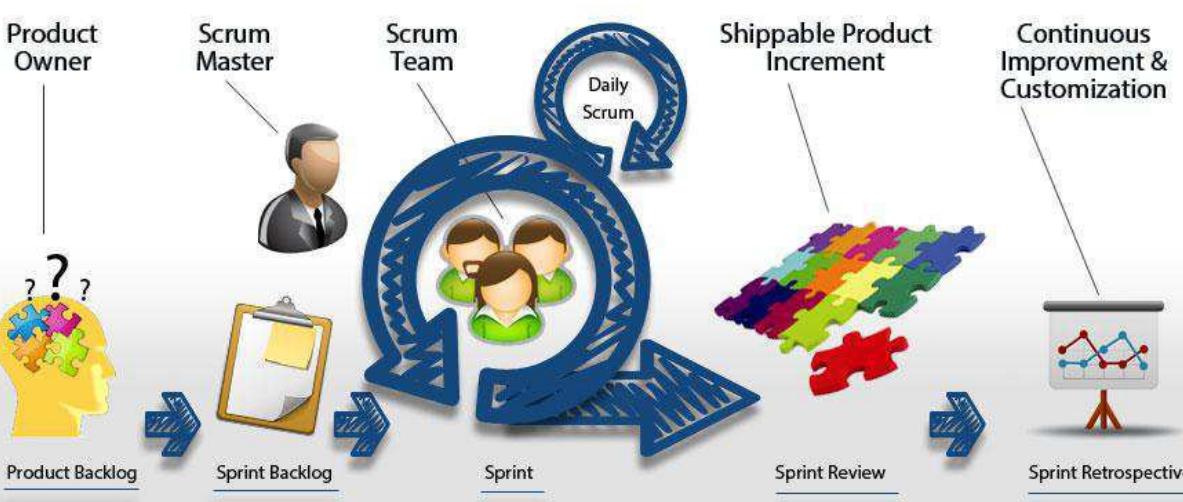
www.scrumprofessionals.org

Contents

Chapter 1: WHAT IS SCRUM?	3
Chapter 2: INTRODUCTION TO SCRUM - A REAL WORLD EXAMPLE	5
Chapter 3: WHAT MAKES WATERFALL SOFTWARE DEVELOPMENT MODEL FAIL IN MANY WAYS?	7
Chapter 4: WHAT MAKES THE SCRUM FRAMEWORK SUCCEED?	8
Chapter 5: SCRUM ROLES – THE SCRUM TEAM	9
Chapter 6: THE SCRUM MASTER	12
Chapter 7: SCRUM PRODUCT OWNER	14
Chapter 8: HOW DOES THE SCRUM FRAMEWORK WORK WITHOUT A PROJECT MANAGER?	15
Chapter 9: THE SCRUM PRODUCT BACKLOG	16
Chapter 10: SCRUM USER STORIES	18
Chapter 11: SCRUM EFFORT ESTIMATIONS – PLANNING POKER®	19
Chapter 12: WHAT IS A SPRINT?.....	20
Chapter 13: SCRUM BURNDOWN CHART	22
Chapter 14: SPRINT PLANNING MEETING	24
Chapter 15: THE SPRINT BACKLOG.....	25
Chapter 16: DEFINITION OF DONE (DoD).....	26
Chapter 17: SPRINT BURNDOWN REPORTS / CHARTS	27
Chapter 18: DAILY SCRUM MEETING / DAILY STAND-UP MEETING.....	28
Chapter 19: SPRINT REVIEW MEETING	29
Chapter 20: SPRINT RETROSPECTIVE MEETING	30
Chapter 21: DISTRIBUTED & LARGE SCRUM PROJECTS.....	31
Chapter 22: MULTI-TEAM COORDINATION & PLANNING	37
Chapter 23: SCRUM RELEASE PLANNING	40
Chapter 24: AGILE MANIFESTO	42

Chapter 1: WHAT IS SCRUM?

Scrum is a lightweight agile project management framework mainly used for software development. It describes an iterative and incremental approach for project work.



Overview of Scrum Framework

Scrum can be used in all kinds of software development: for developing complete software packages, for developing only some parts of bigger systems, for customer or internal projects.

The Scrum Framework implements the cornerstones defined by the agile manifesto:

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

The Scrum Framework itself is very simple. It defines only some general guidelines with only a few rules, roles, artifacts and events. Nevertheless each of these components is important, serves a specific purpose and is essential for a successful usage of the framework.

The main components of Scrum Framework are:

- The three roles: Scrum Master, Scrum Product Owner and the Scrum Team
- A prioritized Backlog containing the end user requirements
- Sprints
- Scrum Events: Sprint Planning Meeting (WHAT-Meeting, HOW-Meeting), Daily Scrum Meeting, Sprint Review Meeting, Sprint Retrospective Meeting

Important in all Scrum projects are self-organization and communication within the team. There is no longer a project manager in a classical sense. In the Scrum Framework the Scrum Master and the Scrum Product Owner share his responsibilities. However, in the end the team decides what and how much they can do in a given project iteration (Sprint).

Another central aspect within the Scrum Framework is continuous improvement: inspect & adapt. The Scrum Teams have to frequently inspect and assess their created artifacts and processes in order to adapt and optimize them. In the midterm this will optimize the results, increases predictably and therefore minimize overall project risk.

The Scrum Framework tries to deal with the fact that the requirements are likely to change quickly or are not completely known at the start of the project. The low-level requirements are only defined at the time when they are going to be really implemented. In Scrum, changes and optimizations of product, requirements and processes are an integral part of the whole engineering cycle.

Another cornerstone of the Scrum Framework is communication. The Scrum Product Owner works closely with the Scrum Team to identify and prioritize functionality. This functionality is written down in user stories and stored in a Scrum Product Backlog. The Product Backlog consists everything that needs to be done in order to successfully deliver a working software system.

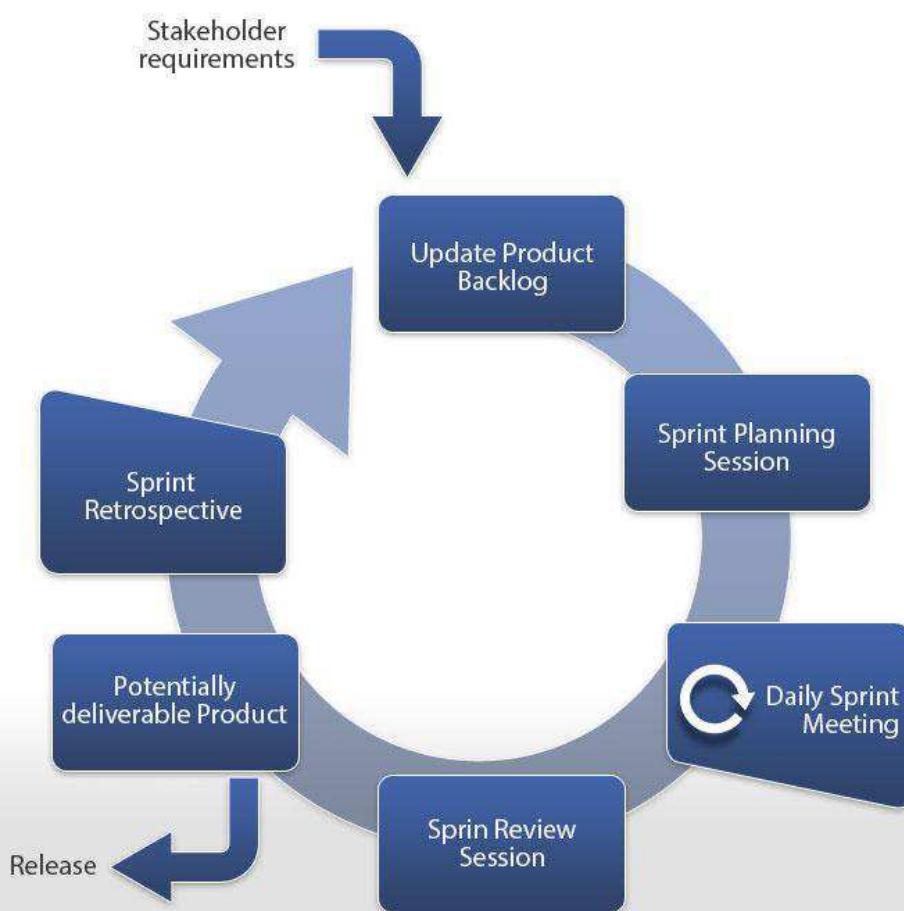
The Scrum Team is empowered to only select the user stories they are sure they can finish within the 2-4 weeks of Sprints. As the Scrum Team is allowed to commit their own goals they will be more motivated and work with best possible performance. The Scrum Master is another important role in the Scrum Framework as it works as a servant-master with the Scrum Team. His/her main tasks are to make the Scrum team understand how Scrum operates, to protect the Scrum Team from external interruptions and to remove impediments that hinder the Scrum Team to reach its maximum productivity.

The Scrum Framework in its simple form is best used for smaller, one-team projects. But with the introduction of additional roles like the "Chief Scrum Product Owner" it is also usable in bigger multi-teams and/or distributed-team projects.

Chapter 2: INTRODUCTION TO SCRUM - A REAL WORLD EXAMPLE

Before starting the first Sprint

Alex is assigned as the Scrum Product Owner of a new software development project. One of his first tasks is to start requirement engineering. He writes down the most important use-cases and discusses them with the architects, customer representatives and other stakeholders. After collecting the high-level use-cases and requirements, he writes them into the Scrum Product Backlog and initiates an estimation and prioritization session with the architects and some senior developers. As a result of this session all the items in the Scrum Product Backlog have an initial rough estimation and a prioritization. Now he starts to break-down the high-level requirements into smaller-grained user stories. With this list he then calls for the first Sprint Planning meeting.



Introduction to Scrum – A Real World Example across various Scrum Phases and Sprints

Sprint 1 - Day 0

During the Sprint Planning meeting Alex presents the Scrum Product Backlog items from the highest priority to the lowest. The team clarifies open questions and for each item the team discusses if they have enough capacity, the required know-how and if everything else needed is available. After this discussion they commit to complete the stories 1,2,3,6,7 and 8 until the end of this sprint. The items 4 and 5 cannot be realized in this sprint, as some technical infrastructure is not yet in place.

After the Sprint Planning meeting Frank - the Scrum Master of the team - calls the team to define the details of how the committed items are going to be implemented. The resulting tasks are written down on the cards at the prepared Sprint Task board. Now everyone of the Scrum Team selects a task to work on.

Sprint 1 - Day 1

In the morning the whole team gets together for their Daily Scrum Meeting. Everyone gives a short statement what has been achieved so far, updates the estimation of remaining hours on the cards of the Sprint Task board, tells what he or she is planning to do today and tells if there are any impediments that hinders him to continue his work. Today one of the team members tells that he has problems because he needs a new license for one of the software tools he is using. Frank checks if other team members have the same problem and says that he'll take care of that after the meeting. After 15 minutes everyone goes back to work.

After the meeting Frank updates the Sprint Burndown. Then he calls the software vendor of the tool, orders licenses and forwards them to the people that need them.

Sprint 1 - Day 2

In the morning again the whole team gets together for their Daily Scrum meeting. In the afternoon one of the Scrum team members is unsure about the details of one of the user stories. He calls Alex –Scrum Product Owner- and discusses the open points with him. After the team member finds out what to do, then he can continue with his implementation.

Sprint 1 - Day 28

This is the final day of the first Sprint and Frank –Scrum Master- has invited the team for the Sprint Review Meeting. The team has prepared a machine with the current software implementation. Alex –Scrum Product Owner- sits in front of the machine and checks if the implementation meets his expectations and if the features are documented as required. At the end of the Review Session he concludes:

- Stories 1,2,6 and 7 are finished as expected.
- Story 3 couldn't be finished in time and was not presented at all.
- Story 8 has some points that have to be re-factoring.

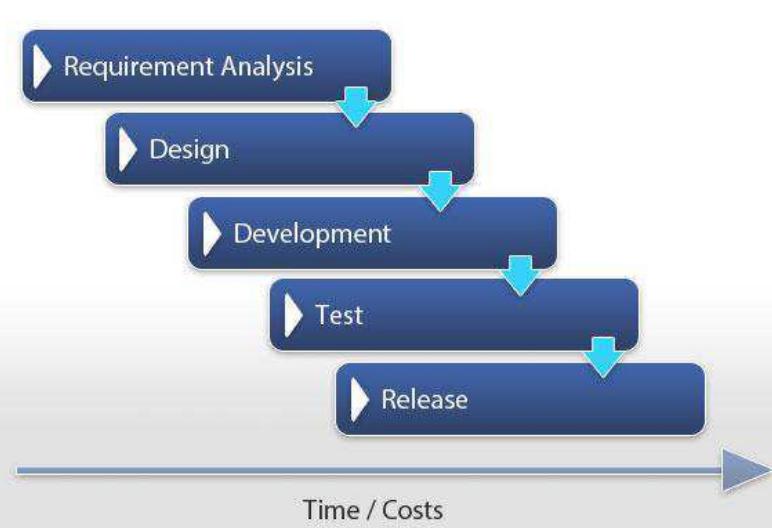
In the afternoon the team gets together for the Sprint Retrospective Meeting and discusses what went well during the sprint and what could be improved. One of the feedback is that the team has the feeling that they do not know enough about the overall system architecture. Frank takes the task to invite the system architect to give a more detailed introduction.

Sprint 2 - Day 1

Alex –Scrum Product Owner- adds new items to the Scrum Product Backlog based on his recent customer meetings. Moreover, he adds additional items for the re-factoring of story 8. Alex then invites the team for the Sprint Planning Meeting for Sprint 2. The team discusses and commits to stories with the guidance of Frank – Scrum Master- and the second Sprint begins.

Chapter 3: WHAT MAKES WATERFALL SOFTWARE DEVELOPMENT MODEL FAIL IN MANY WAYS?

Studies have shown that in over 80% of the investigated and failed software projects, the usage of the Waterfall methodology was one of the key factors of failure. But why?



Phases in the Classical Waterfall Software Development Model

As shown above, when deploying the waterfall methodology there is a strict sequential chain of the different project phases. A previous phase has to be completed before starting the next phase. Going back is in most cases difficult, costly, frustrating to the team and time consuming.

The project timeline is planned at the start. A releasable product is delivered only at the end of the project timeline. If one phase is delayed all other phases are also delayed.

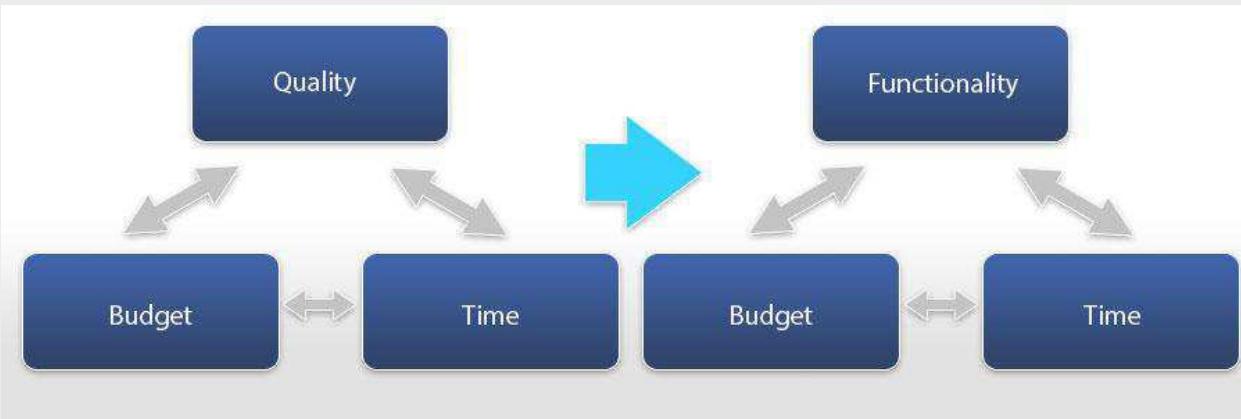
To avoid this users of the waterfall methodology normally try to anticipate all possibilities beforehand. This means that in one of the earliest phases of the project they try to define all requirements as fine grained and complete as possible. However, requirement definition in such an early phase of the project is often very difficult and therefore many requirements change (or should change) throughout the project. Studies have shown that in larger and complex projects about 60% of the initial requirements are changed throughout the project. Other requirements are implemented as defined but are not really needed by the customer and consume valuable time that could have been better used to implement functionality with a higher added value for the customer.

The separation into different project phases force users to estimate each phase separately. The problem is that most of these phases are normally not separate. They are working together and in parallel.

The waterfall approach for developing software can be used for implementing small and simple projects. But for bigger and more complex projects it can be said that this approach is highly risky, often more costly and generally less efficient than Scrum Project Management Framework.

Chapter 4: WHAT MAKES THE SCRUM FRAMEWORK SUCCEED?

The Scrum framework changes the classical triangle of project management. The compromise is no longer between Time, Budget and Quality. It is now becoming the triangle of Budget, Time and Functionality.



Triangle of Project Management

Quality is no longer an option. In Scrum the factors that define when a feature is complete (in terms of quality, required testing, documentation etc.) are defined by the Definition Of Done (DoD) right at the start of the project. No incomplete or untested feature will be released to the customer. Now the functionality to implement will be defined throughout the course of the project and implemented incrementally. This incremental development allows to remain flexible and to change in a controlled manner without the additional costs and risks of jeopardizing large chunks of previous work. At the end of each increment (Sprint) a result is available that can be shown and discussed with the customer to get and incorporate feedback as soon as possible.

As this flexibility does not only apply to software requirements, but also to the operational processes themselves, the Scrum Framework allows optimizing resources usage (time, budget) and minimizes waste.

Studies have shown that Scrum has following positive effects in practice:

- Increased productivity
- Better product quality
- Reduced or stable project costs after introducing agile methods
- Higher customer satisfaction
- Increased satisfaction and motivation of the employees

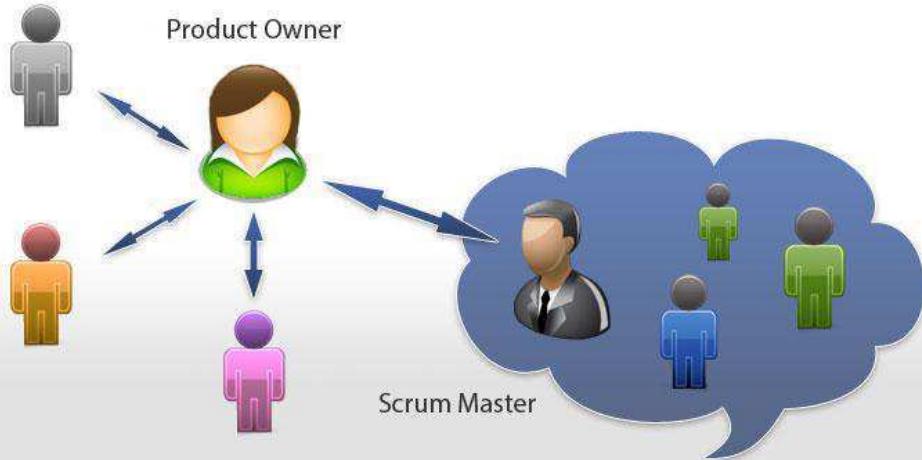
So even if introducing and using the Scrum Framework is sometimes non-trivial, the flexible and iterative approach of the Scrum Framework eases handling of complexity and better copes with the fact of ever-changing customer and business requirements. Thus, Scrum is in most cases a better alternative to the classical methodologies.

Chapter 5: SCRUM ROLES – THE SCRUM TEAM

Within the Scrum Framework three roles are defined:

- The Scrum Team
- Scrum Master
- Scrum Product Owner

Each of these roles has a defined set of responsibilities and only if they fulfill these responsibilities, closely interact and work together they can finish a project successfully.



Scrum Roles & Stakeholders

The Scrum Team

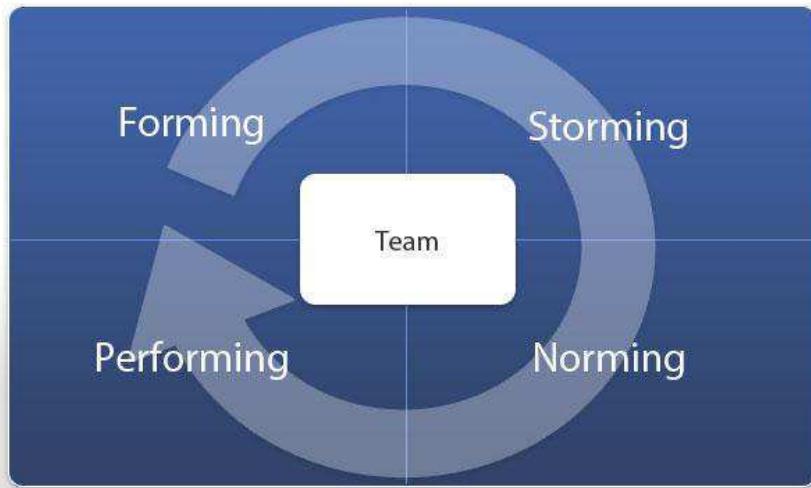
Within the Scrum Framework all work delivered to the customer is done by dedicated Scrum Teams. A Scrum Team is a collection of individuals working together to deliver the requested and committed product increments.

To work effectively it is important for a Scrum Team that everyone within the team

- follows a common goal
- adheres the same norms and rules
- shows respect to each other

When setting up a new Scrum Team one always has to keep in mind that no new team will deliver with the highest possible performance right from the beginning. After setting up the team it has to go through certain phases as described by the Tuckman-Model: Forming, Storming, Norming, Performing.

How long it takes until the Scrum Team is in the Performing Phase depends on the team, and yet it normally takes about 3 Sprints until the teams is mature enough to deliver their results in a predictable way.



Tuckman Model

Characteristics of a Scrum Team

Scrum Teams always have the following characteristics:

- Team members share the same norms and rules
- The Scrum team as a whole is accountable for the delivery
- The Scrum Team is empowered
- It is working as autonomous as it is possible
- The Scrum Team is self-organizing
- The skills within the Scrum team are balanced
- A Scrum Team is small and has no sub-teams
- The people within the Scrum Team work full time in the team
- People are collocated

Rules & Norms

Of course their environment defines some of the norms the teams have to follow, but some rules and norms are developed during the Norming phase. This set of common rules is quite important. Otherwise the team members would have to constantly waste valuable time to switch between different value systems and rule sets. Examples for such norms and rules are:

- time and location of the Daily Scrum Meeting
- the Definition Of Done (DoD) used to decide if work is finished or not
- coding guidelines
- tools to use

Accountability

The Scrum Team as a whole is responsible to deliver the committed delivery in time and with the defined quality. A good result or a failure is never attributed to a single team member but always the result of the Scrum Team.

Empowerment & Self organization

The Scrum Team has to be empowered to define

- what it will commit to deliver at the end of the sprint
- how the expected results have to be broken down into tasks
- who will perform the task and in which order they are performed

Only if the Scrum Team is empowered to decide these things it will work with the highest possible motivation and performance.

Balanced set of skill

Individuals within the Scrum Team will most certainly have specialized skills and focus. However to achieve best possible performance it would be optimal to have a balanced set of skills. Only then the Scrum Team will be able to deal with the ever-changing challenges and can act as autonomous as it is possible.

On one hand this means that a Scrum Team should be multidisciplinary (developers, tester, architects etc) right from the beginning. On the other hand this also means that each team member should learn a little bit of each other's specialization, e.g. a if required to finally reach the committed goal a developer should also perform or write tests.

As a consequence this also means that within the Scrum Framework it is not differentiated between e.g. "tester" and "architect", they all share the same title "Scrum Team Member" even if the primary skill is not to develop production code.

Size of the Scrum Team

Scrum Teams are small. The ideal size is 7 +/- 2 people.

If there are more people the communication overhead gets too large and the team should be split into multiple Scrum Teams. These Scrum Teams should be coordinated and communicate with each other but otherwise work independently.

Collocation

To minimize unnecessary communication overhead each Scrum Team should be collocated. If work has to be spread over multiple locations, independent Scrum Teams should be created.

Responsibilities of the Scrum Team

The Scrum Team and each of the team members has certain responsibilities which have to be fulfilled:

- They have to breakdown the requirements, create task, estimate and distribute them. In other words this means that they have to create the Sprint Backlog.
- They have to perform the short Daily Sprint Meeting.
- They have to ensure that at the end of the Sprint potentially shippable functionality is delivered.
- They have to update the status and the remaining efforts for their tasks to allow creation of a Sprint Burndown Diagram.

Chapter 6: THE SCRUM MASTER

Broad speaking it is the job of the Scrum Master to ensure that the Scrum Team adheres to the Scrum theory, practices and rules.

The Scrum Master is part of the Scrum Team and acts as a servant-leader for the Scrum Team. In the beginning this will be a full-time job so that the Scrum Master will not be able to directly contribute to the Sprint results. However after some Sprints the processes will settle so that the workload for the Scrum Master will drop and he could actively contribute to the Sprint Goal.

Since it is crucial that there is trust between the Scrum Master and the other team members it would be ideal if the Scrum Team selects the Scrum Master itself. However, in reality most often the Management selects the Scrum Master. To get the required trust the Scrum Master should have no line-management responsibility for one of the team members. Otherwise the necessary open communication and decision finding will be hampered.

Responsibilities of the Scrum Master

The Scrum Master has several important responsibilities:

- Guard the Scrum Team from external requests and disruptions
- Act as a change agent and adapt processes to maximize productivity of the team
- Coach the Scrum Team
- Remove impediments for the Scrum Team
- Ensure efficient communication between the Scrum Team and the Scrum Product Owner
- Facilitate the various Scrum Events

In order to effectively do this, a number of skills are helpful:

- Moderation
- Coaching
- Development know-how

Guarding the Scrum Team / Removing impediments

An important job of the Scrum Master is to guard the team members from "urgent requests". Line-Management or the Scrum Product Owner will often try to assign new, unplanned and not committed requests to the team or individual team members. However one of the key aspects of Scrum is that all deliverables and work-packages are known and committed by the Scrum Team before the Sprint and that the Scrum Team can work 100% on these deliverables. The job of the Scrum Master is to discuss such requests and to either postpone the request until the next sprint starts or to cancel the current sprint and start-over.

The developers within the Scrum Team should only concentrate on developing customer value by delivering potentially shippable functionality. The Scrum Master helps by removing impediments that block or hinder development. Examples could be organizing meetings, clarifying questions or performing supporting work.

Change agent

One of the cornerstones of the Scrum Framework is continuously improvement through inspect & adapt. The Scrum Master hosts and moderates the Scrum Retrospective Meeting and his job is then to facilitate the change of the identified shortcomings.

Facilitation of Scrum Events

The Scrum Framework defines several meetings that have to be organized and facilitated by the Scrum Master:

- Daily Scrum Meetings
- Sprint Planning Meetings
- Sprint Review Meetings
- Sprint Retrospective Meeting

Chapter 7: SCRUM PRODUCT OWNER

The Scrum Product Owner is a central role within the Scrum Framework. Most of the responsibilities of the classical product manager and the project manager are combined within this single role.

He represents the end customer and/or other stakeholders and is responsible for maximizing the value of the product by ensuring that the right work is done at the right time.

As a consequence this means of course that the Scrum Product Owner has to work very closely with the Scrum Team and coordinates their activities over the whole lifetime of the project. No one else is allowed to tell the development team to work from a different set of priorities.

The Scrum Product Owner has a number of responsibilities:

- Managing the Scrum Product Backlog
- Release Management
- Stakeholder Management
- Work closely with the Scrum Team

Of course the Scrum Product Owner can delegate certain activities (like physically maintaining the Scrum Product Backlog), but in the end he remains responsible.

Managing the Scrum Product Backlog

The Scrum Product Owner is the only person allowed to manage the contents of the Scrum Product Backlog. This means he has to:

- Create, maintain and clearly describe the Scrum Product Backlog items
- Prioritize the items to best achieve goals and mission
- Ensuring that the Scrum Team understands the items in the Scrum Product Backlog

Release Management

The Scrum Product Owner is responsible for reaching the project goals. He creates and maintains the release plan and decides about deliveries, functionalities and therefore about the costs of a project.

He manages the Scrum Team by creation and prioritization of appropriate Scrum Backlog items.

Stakeholder Management

External stakeholder should not communicate directly with the Scrum Team. Instead the Scrum Product Owner should collect and discuss required functionalities with the different Stakeholders (e.g. customer, marketing, service etc). These requirements are then combined and filtered before giving it to the team in the form of prioritized Scrum Product Backlog Items.

Work closely with the Scrum Team

For a successful project it is important that the Scrum Product Owner and the Scrum Team work together very closely. He is responsible that everyone in the Scrum Team understands what is required.

The Scrum Product Owner is also responsible for checking and accepting the Sprint results during the Sprint review session.

Chapter 8: HOW DOES THE SCRUM FRAMEWORK WORK WITHOUT A PROJECT MANAGER?

In a traditional project normally a product manager defines the requirements. Realization is then delegated to a project manager and he coordinates all activities needed to realize the project.

The Scrum Framework does not define a “project manager” in the classical sense although these activities are of course required. Within the Scrum Framework the responsibilities of a project manager are distributed over the roles of the Scrum Product Owner and the Scrum Master.

The definition of the role and responsibilities of the Scrum Master takes into account that decisions about functionality, release plan and costs can be done much easier and better if one person is responsible for the content and the planning. Otherwise there will always be a tension between the Scrum Product Owner (not responsible for the project) and the project manager (not responsible for the content). If these responsibilities are combined everything will be much easier.

Nevertheless the Scrum Master will take over some of the responsibilities of a traditional project manager. Tracking the tasks within a Sprint and resolving impediments are within his responsibility. Since he is part of the Scrum Team it is much easier and much more efficient to handle such topics directly in the team.

Chapter 9: THE SCRUM PRODUCT BACKLOG

In the simplest definition the Scrum Product Backlog is simply a list of all things that needs to be done within the project. It replaces the traditional requirements specification artifacts. These items can have a technical nature or can be user-centric e.g. in the form of user stories. The owner of the Scrum Product Backlog is the Scrum Product Owner. The Scrum Master, the Scrum Team and other Stakeholders contribute it to have a broad and complete To-Do list.

Working with a Scrum Product Backlog does not mean that the Scrum Team is not allowed to create and use other artifacts. Examples for additional artifacts could be a summary of the various user roles, workflow descriptions, user interface guidelines, storyboards, or user interface prototypes. However, these artifacts do not replace the Scrum Product Backlog but complement and detail its content.

The Scrum Product Owner uses the Scrum Product Backlog during the Sprint Planning Meeting to describe the top entries to the team. The Scrum Team then determines which items they can complete during the coming sprint.

Each Scrum Product Backlog has certain properties that differentiate it from a simple to-do list:

- an entry in the Scrum Product Backlog always add value for the customer
- the entries in the Scrum Product Backlog are prioritized and ordered accordingly
- the level of detail depends on the position of the entry within the Scrum Product Backlog
- all entries are estimated
- the Scrum Product Backlog is a living document
- there are no action-items or low-level tasks in the Scrum Product Backlog

ToDo List				
ID	STORY	ESTIMATION	PRIORITY	
7	As an unauthorized User I want to create account	3	1	
1	As an unauthorized User I want to login	1	2	
10	As an unauthorized User I want to logout	1	3	
9	Create script to purge database	1	5	
2	As an authorized User I want to see the List of items so that I can select one	2	5	
4	As an authorized User I want to add a new Item so that it appears in the list	5	6	
3	As an authorized User I want to delete the Selected item	2	7	
5	As an authorized User I want to edit the Selected item	5	8	
6	As an authorized User I want to set a reminder for a selected item so that I am reminded when item is due	8	9	
8	As an administrator I want to see the list of accounts on login	2	10	
TOTAL			30	

Example Scrum Product Backlog

Only entries that add value

Each entry in the Scrum Product Backlog must have some kind of customer value. Entries without any customer value are pure waste and should not be present anyway. The Scrum Product Backlog can include entries for the exploration of customer needs or various technical options, a description of both functional and nonfunctional requirements, the work necessary to launch the product, and other items as well, such as setting up the environment or remediating defects. Some tasks may not add direct value to the functionality.

Nevertheless they might add value by increasing quality or reducing incidents in the long term.

Living document

The Scrum Product Backlog is changed throughout the whole project. If needed, new requirements are added and existing requirements may be modified, defined in more detail or even deleted. Requirements are no longer frozen early on. Instead the final set of requirements within the Scrum Product Backlog is also developed iteratively, together with the resulting software. This is different to traditional requirements engineering but allows maximizing customer value and minimizes development effort.

Different level of details

The requirements in the Scrum Product Backlog have a different granularity. Only those requirements that shall be implemented during one of the next sprints are defined in greater detail and everything else is more coarse-grained. The simple reason for this is that it does not make sense to invest time and effort into the specification of these requirements, as most of these requirements will have changed anyway until implementation starts.

No low-level tasks

The Scrum Product Backlog shall not contain the detailed requirement information. Ideally the final requirements are defined together with the customer during the sprint. Breakdown and distribution of these requirements is the responsibility of the Scrum Team.

The Scrum Product Backlog is ordered

All entries are prioritized and the Scrum Product Backlog is ordered. The Scrum Product Owner with the help of the Scrum Team does the prioritization. Added Value, Costs and Risks are the most common factors for prioritization. With this prioritization the Scrum Product Owner decides what should be done next.

All entries are estimated

All the entries within the Scrum Product Backlog have to be estimated according to the agreed definition (e.g. story points). This estimation can then be used to prioritize entries in the Scrum Product Backlog and to plan releases.

Working with the Backlog

The backlog needs regular attention and care - it needs to be managed carefully. At the start of the project the Scrum Team and its Scrum Product Owner start by writing down everything they can think of easily. This is almost always more than enough for a first sprint.

After this initial setup, the Scrum Product Backlog has to be maintained in an ongoing process that comprises the following steps:

- As new items are discovered they are described and added to the list. Existing ones are changed or removed as appropriate.
- Ordering the Scrum Product Backlog. The most important items are moved to the top.
- Preparing the high-priority entries for the next Sprint Planning Meeting
- (Re-)Estimating the entries in the Scrum Product Backlog

The Scrum Product Owner is responsible for making sure that the Scrum Product Backlog is in good shape this is a collaborative process. When using the Scrum Framework about 10% of the Scrum Teams total time should be reserved for maintaining the Scrum Product Backlog (discussion, estimation etc.).

The collaborative maintenance of the Scrum Product Backlog helps to clarify the requirements and creates a buy-in from the Scrum Team.

Chapter 10: SCRUM USER STORIES

The entries in the Scrum Product Backlog are often written in the form of User Stories. A User Story tells a short story about someone using the product. It contains a name, a brief narrative, and acceptance criteria and conditions for the story to be complete. The advantage of user stories is that they focus on exactly what the user needs/wants without going into the details on how to achieve it.

There are different recommendations how to define User stories. A template could be as follows:

As an [actor], I [want|must] [action] so that [achievement]

Or in a shorter version:

As an [actor], I [want|must] [achievement]

Actor: The 'owner' of the user story. This is often a user but it is recommended to be more specific here. By using specific actors (e.g. "Administrator", "Logged in Customer", "Unauthenticated visitor") it's easier to understand and sets the user story into context.

Action: What the actor want to do. If it is a mandatory action it can be prefixed by "must". Otherwise "want" is used.

Achievement: What the actor wants to achieve by performing the action. This is the result from executing the action seen from the actor's point of view.

```
As a <customer> I want to <see the catalog of salable items>
      so that <I can order one of them>
As an <administrator> I want <be able to disable accounts>
As a <trainee> I must <answer all the questions>
```

Example User story

Chapter 11: SCRUM EFFORT ESTIMATIONS – PLANNING POKER®

All the entries within the Scrum Product Backlog have to be estimated to allow the Scrum Product Owner to prioritize the entries and to plan releases. This means that the Scrum Product Owner needs an honest assessment of how difficult the work will be. Nevertheless it is recommended that the Scrum Product Owner does not attend the estimation to avoid pressuring (intentionally or otherwise) the Scrum Team.

The Scrum Framework itself does not prescribe a single way for the Scrum Teams to estimate their work. However within the Scrum Framework the estimation is not normally done in terms of time - a more abstracted metric to quantify effort is used. Common estimating methods include numeric sizing (1 through 10), t-shirt sizes (XS, S, M, L, XL, XXL, XXXL) or the Fibonacci sequence (1, 2, 3, 5, 8, 13, 21, 34, etc.).

Important is that the team shares a common understanding of the scale it uses, so that every member of the team is comfortable with it.

Planning Poker® / Scrum Poker

One commonly used method during the estimation process is to play Planning Poker® (also called Scrum Poker). When using Planning Poker®, influence between the participants are minimized and therefore a more accurate estimation result is produced.

In order to play Planning Poker® the following is needed:

- The list of features to be estimated
- Decks of numbered cards.

A typical deck has cards showing the Fibonacci sequence including a zero: 0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89; other similar progressions are also possible. The reason for using the Fibonacci sequence is to reflect the uncertainty in estimating larger items. A high estimate usually means that the story is not well understood in detail or should be broken down into multiple smaller stories. Smaller stories can be estimated in greater detail. It would be a waste of time to discuss if it is 19, 20 or 25; the story is simply (too) big.

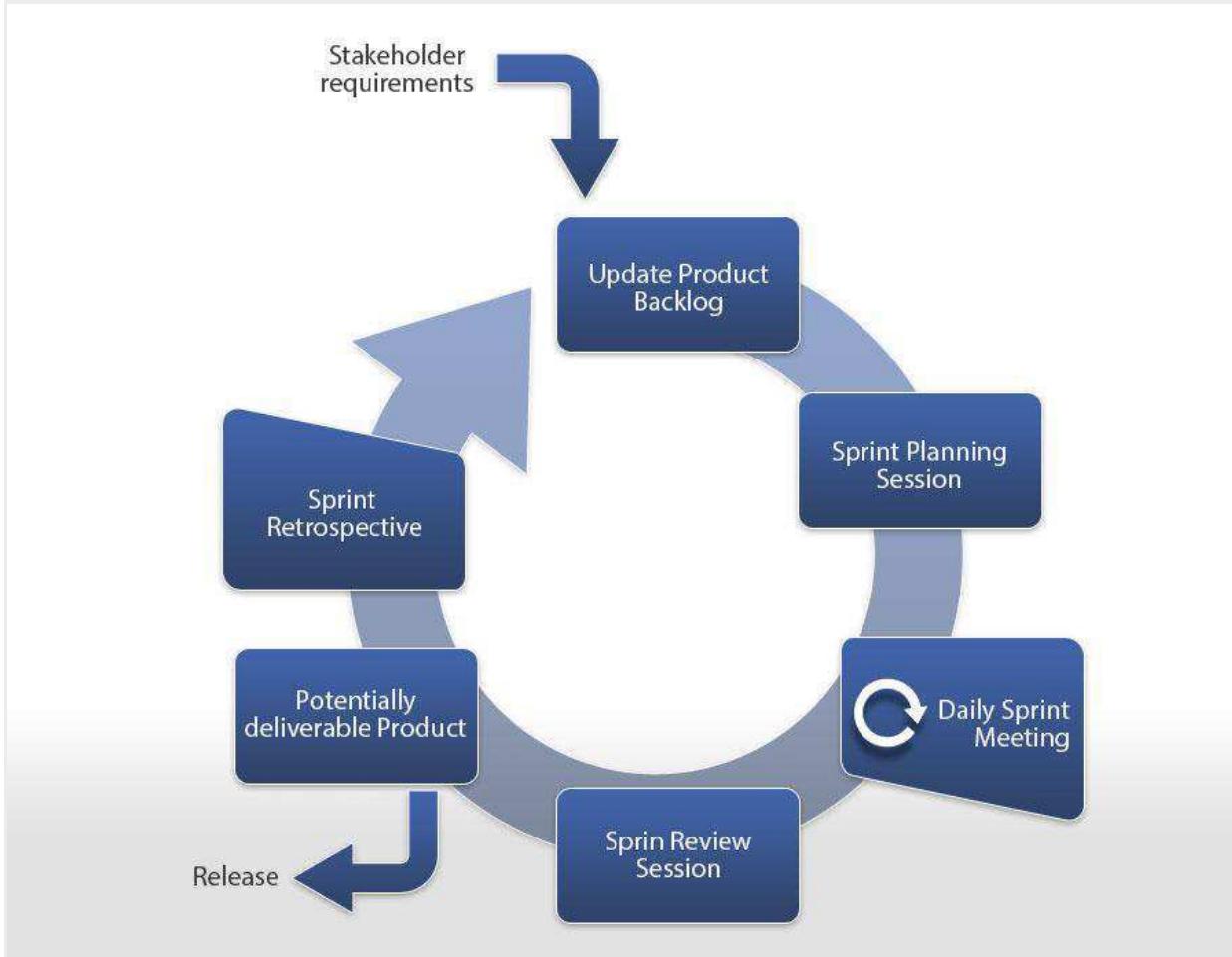
The game is then played in the following steps:

- The Scrum Product Owner presents the story to be estimated. The Scrum Team asks questions and the Scrum Product Owner explains in more detail. If many stories have to be estimated a time-constraint (e.g. only one minute for explanation) might be set as well. If the time-constraint is reached and the Scrum Team does not understand the story it is a sign that the story has to be re-written.
- Each member of the Scrum Team privately chooses the card representing the estimation.
- After everyone has chosen a card, all selections are revealed.
- People with high and low estimates are allowed to explain their estimate.
- Estimation starts again until consent is found.
- This game is repeated until all stories are estimated.

Chapter 12: WHAT IS A SPRINT?

In the Scrum Framework all activities needed for the implementation of entries from the Scrum Product Backlog are performed within Sprints (also called 'Iterations'). Sprints are always short: normally about 2-4 weeks.

Each Sprint follows a defined process as shown below:



The Sprint Process

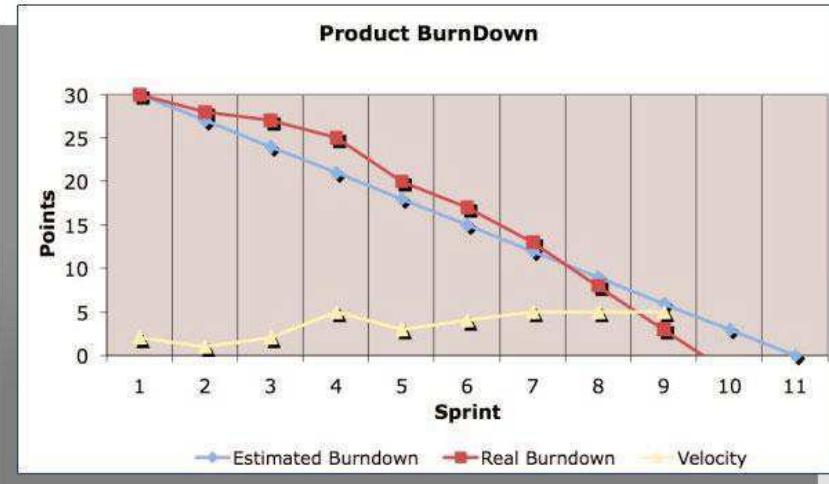
Each Sprint starts with two planning sessions to define the content of the Sprint: the WHAT-Meeting and the HOW-Meeting. The combination of these two meetings are also defined as Sprint Planning Meeting. In the WHAT-Meeting the Scrum Team commits to the User Stories from the Scrum Product Backlog and it uses a HOW-Meeting to break the committed User Stories into smaller and concrete tasks. Then implementation begins.

At the end of the Sprint a Sprint Review Meeting is conducted to allow the Scrum Product Owner to check if all of the committed items are complete and implemented correctly. Additionally a Sprint Retrospective Meeting is conducted to check and improve the project execution processes: What was good during the Sprint, what should continue as it is and what should be improved.

During the Sprint a short daily Standup-Meeting (Daily Scrum Meeting) is held to update the status of the items and to help self-organization of the team.

Chapter 13: SCRUM BURNDOWN CHART

The Scrum Burndown Chart is a visual measurement tool that shows the completed work per day against the projected rate of completion for the current project release. Its purpose is to enable that the project is on the track to deliver the expected solution within the desired schedule.

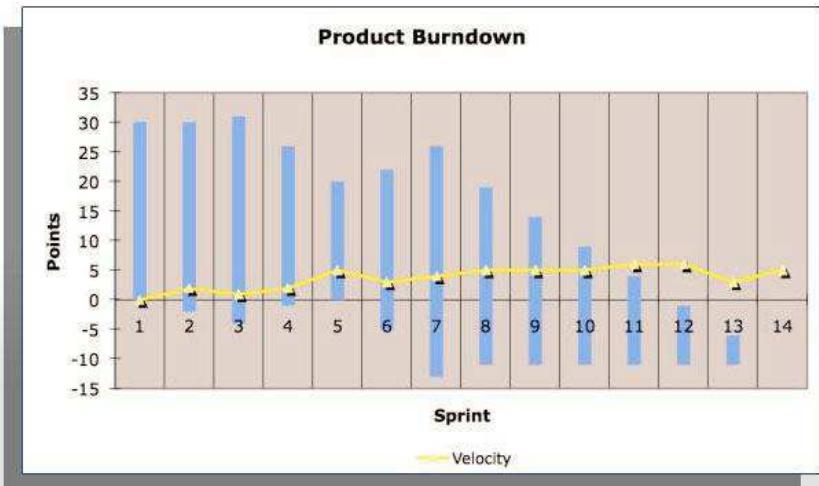


Simple Burndown Chart

The rate of progress of a Scrum Team is called "velocity". It expresses the amount of e.g. story points completed per iteration. An import rule for calculating the velocity is that only stories that are completed at the end of the iteration are counted. Counting partially finished work (e.g. coding only - test missing) is strictly forbidden.

After a few Sprints the velocity of a Scrum Team will most likely be predictable and would allow quite accurate estimation about the time needed until all entries in the Scrum Product Backlog will be completed. If the velocity of a Scrum Team is e.g. 30 story points and the total amount of remaining work is 155, we can predict that we need about 6 Sprint to complete all stories in the Backlog.

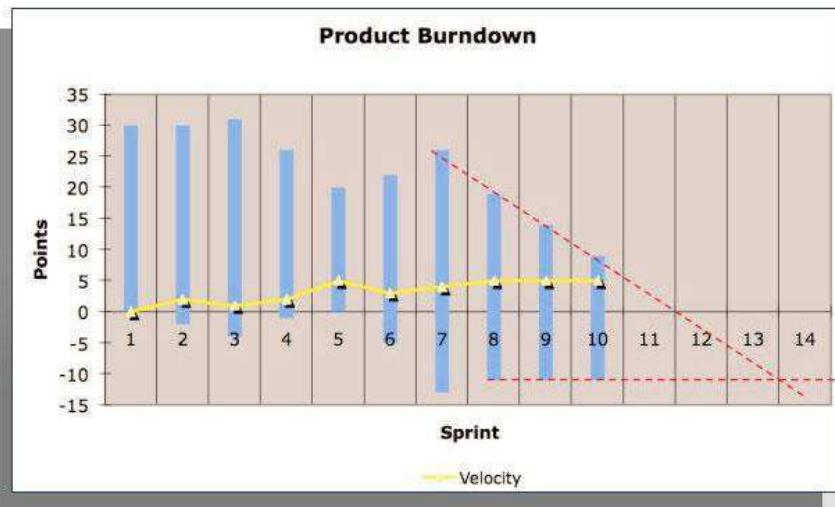
However in reality the entries in the Scrum Product Backlog will change over the duration of the project. New stories are added and other stories are changed or even deleted. In the simple Burndown Chart the velocity of the Scrum Team and the change in the scope cannot be distinguished. To reflect this, another form of diagram can be used.



Separating Velocity and Scope Changes

Here a bar chart instead of a line diagram is used. The size of each bar represents the total amount of work remaining at the start of each sprint. The velocity of the team is subtracted from the top while changes in the Scope change the bottom of the bar.

To get even more accurate we can also take the rate of changes in total work into account. However we have to be careful when using this model since the rate of change will be high in the beginning of the project but will drop at the end.



Extended Burndown Chart with Prediction

Chapter 14: SPRINT PLANNING MEETING

Each Sprint and each Sprint Planning Meeting starts with a WHAT-Meeting. Goal of this session is to define a realistic Sprint Backlog containing all items that could be fully implemented until the end of the Sprint.

Preparation

For a successful WHAT-Meeting some preparation is necessary:

- The Scrum Product Owner defines the Sprint Goal.
- Based on this goal the relevant entries in the Scrum Product Backlog are chosen by the Scrum Product Owner.
- These entries are updated and broken into smaller stories so that they can be completed within one Sprint.
- The entries are estimated & prioritized.
- The team defines their capacity for the upcoming Sprint.

Sprint Goal

The Scrum Product Owner defines the Sprint Goal. It is a short description of what the sprint will attempt to achieve and what should be realistic and comprehensible for everyone.

Team Capacity

The total capacity of the Scrum Team might change from Sprint to Sprint. In order to come to realistic commitments it is necessary to know the total capacity of the team for the upcoming Sprint considering e.g. vacations, public holidays, efforts for Scrum Meetings and time needed for other activities during the Sprint.

The Meeting Session

During the session the Scrum Product Owner presents the Sprint Goal and discusses it with the team. After that the Scrum Team iterates through the relevant items in the Scrum Product Backlog and the team commits to the entries which they think can be fully completed within the Sprint. The decision should be based on available capacity and knowledge about the entries.

At the end of the session the list of all committed entries from the Scrum Product Backlog provides the base for the HOW-Meeting and the Scrum Backlog.

The HOW-Meeting

The goal of the How-Meeting is to fill the Sprint Backlog by identifying the concrete tasks needed for complete implementation of the Scrum Product Backlog entries. Tasks normally include design-, implementation-, test- and documentation-activities.

The HOW-Meeting can be done in a separate session after the WHAT-Meeting, during the WHAT-Meeting when committing the entries or both.

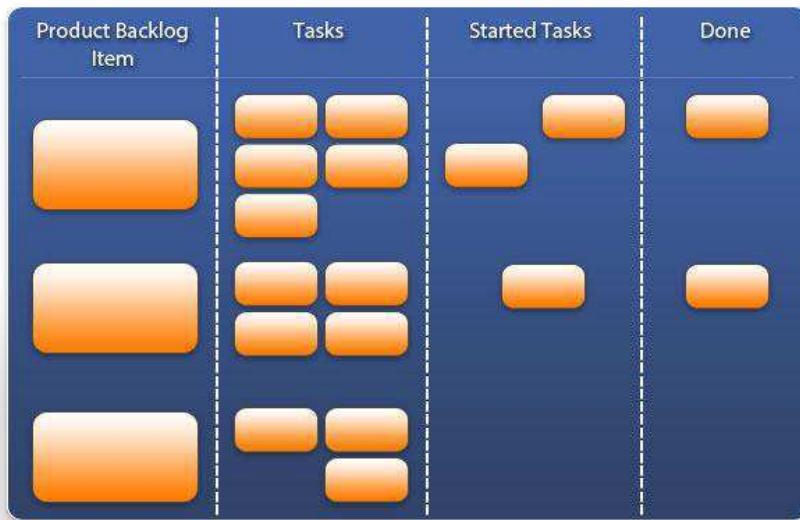
After identifying the necessary activities they are estimated by the team. Base for this estimation should be person-hours. The team should how long it will need to do everything that is required to finish this activity.

Chapter 15: THE SPRINT BACKLOG

Within the Sprint Backlog all activities required to complete the committed entries from the Scrum Product Backlog are stored. All entries have to be estimated on a person-hour base in order to track progress and remaining efforts.

The Sprint Backlog is a living artifact and is updated on a daily base. If a team member starts to work on an activity his name is recorded within the sprint backlog. New activities can be added to the Sprint Backlog during the Sprint. At the end of the day all remaining efforts are updated and this defines how much work is left until the Sprint Goal is reached. The Definition Of Done (see below) is used to decide if an item is done or not.

The Sprint Backlog can be kept electronically within e.g. an Excel-Sheet or with cards on a task board. The latter has some advantages (e.g. transparency and easy access) but add additional complexity if the Scrum Team is distributed over multiple sites. The figure below shows an example how such a task board could be organized. The structure should be adapted to reflect the needs of the project.



Example Sprint Task Board

Chapter 16: DEFINITION OF DONE (DoD)

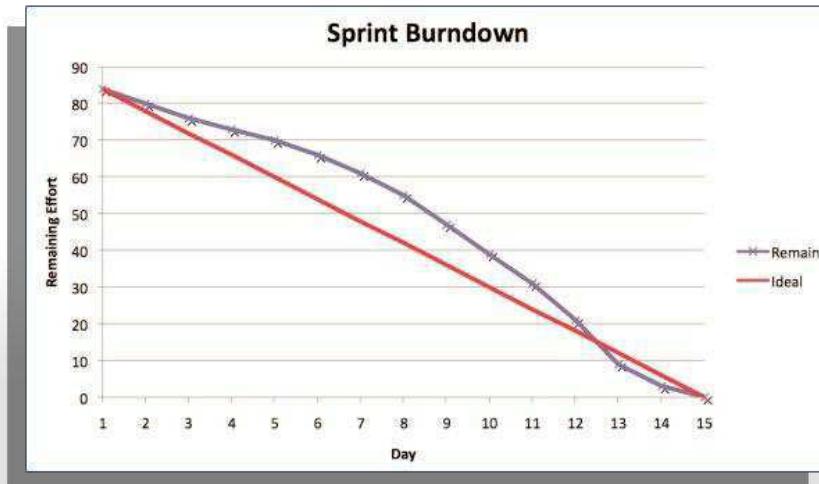
In order to be able to decide when an activity from the Sprint Backlog is completed, the Definition of Done (DoD) is used. It is a comprehensive checklist of necessary activities that ensure that only truly done features are delivered, not only in terms of functionality but in terms of quality as well. The DoD may vary from one Scrum Team to another, but must be consistent within one team.

There might be different DoD at various levels:

- DoD for a Scrum Product Backlog item (e.g. writing code, tests and all necessary documentation)
- DoD for a sprint (e.g. install demo system for review)
- DoD for a release (e.g. writing release notes)

Chapter 17: SPRINT BURNDOWN REPORTS / CHARTS

The Sprint Burndown Report shows the progress within the Sprint toward reaching the Sprint Goal. It provides transparency about the current performance (burndown rate) and allows easy estimation if the Sprint Goal can be reached in time or if the team has to find additional measures to speed-up completion of the remaining activities.



Sprint Burndown Report/Chart

The initial Sprint Backlog defines the start-point for the remaining efforts. The remaining effort of all activities are collected on a daily base and added to the graph. In the beginning the performance is often not as good as predicted by the ideal burndown rate due to wrong estimations or impediments that have to be removed in order to get on full speed.

Chapter 18: DAILY SCRUM MEETING / DAILY STAND-UP MEETING

The daily Scrum meeting is a short everyday meeting, ideally during start of the working day. Each team member who works towards the completion of a given sprint needs to participate. During this meeting, each team member should briefly provide the answers of the following three questions:

- What has he/she accomplished since the last daily Scrum meeting?
- What is he/she going to accomplish until the next Scrum meeting?
- What are the impediments that prevent he/she from accomplishing his/her tasks?

All team members should attend and they should stand during the meeting. The daily Scrum meeting should ideally not last more than 15 minutes. On the other no issues or concerns raised during the meeting are allowed to be ignored due to the lack of time. Issues or concerns ought to be recorded by the Scrum Master and needs to be specifically handled after the meeting.

Chapter 19: SPRINT REVIEW MEETING

At the end of each sprint a Sprint Review meeting is held. During this meeting the Scrum Team shows which Scrum Product Backlog items they completed (according to the Definition of Done) during the sprint. This might take place in the form of a demo of the new features.

It is important to notice that Backlog items that are not completed shall not be demonstrated. Otherwise this might suggest that these items are finished as well. Instead incomplete items/remaining activities shall be taken back into the Scrum Product Backlog, re-estimated and completed in one of the following sprints.

The Sprint Review meeting should be kept very informal. No PowerPoint slides should be used and time for preparation and performing the meeting should be limited. During the meeting the Scrum Product Owner inspects the implemented backlog entries and accepts the solution or adds new stories to the Scrum Product Backlog to adapt the functionality.

Participants in the sprint review typically include the Scrum Product Owner, the Scrum Team and the Scrum Master. Additionally management, customers, and developers from other projects might participate as well.

Chapter 20: SPRINT RETROSPECTIVE MEETING

After the Sprint Review meeting took place the Scrum Team and the Scrum Master get together for the Sprint Retrospective. In this meeting all team members reflect on the past sprint and check three things: what went well during the sprint, what didn't, and what improvements could be made in the next sprint. The meeting should be time-boxed (e.g. 3 hours).

The Sprint Retrospective is an integral part of the “inspect and adapt” process. Without this meeting the team will never be able to improve their overall output and cannot focus on the overall team performance. Therefore actionable suggestions to improve performance should be available at the end of the meeting.

Chapter 21: DISTRIBUTED & LARGE SCRUM PROJECTS

The Scrum Framework - as described so far - works best for a single Scrum Team in one location. However, in reality a single Scrum Team often cannot realize projects or resources are spread over multiple locations. As a consequence the number of teams has to be increased and/or the teams will be distributed. The reasons for this can be technical (e.g. experts are not available locally), size-related (project too big) or business-related (e.g. usage of resources in low-cost countries or speed-up by usage of different time-zones).

As communication is an integral part of the Scrum Framework, special care had to be taken to overcome the challenges when working within a distributed environment. Therefore all team members should have access to the appropriate communication tools (e.g. video conferencing and web cams) to breaking down the more tangible communication barriers.

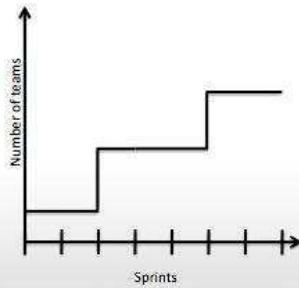
Project Organization - Multiple Teams

The simplest way of extending the Scrum Framework when working in a large-scale project is to increase the number of teams in the same location.



Multiple Teams in a Single Location

If multiple teams have to be used to implement the requirements it is important to make sure that the number of teams does not grow too fast. Best is to start with a single team and after the first sprints have been completed adding a small number of other teams. If required after these teams are productive other teams could be added.



Increasing the Number of Teams

For creating new teams there are two possibilities:

- Splitting an existing team into new teams and add new members
- Adding completely new teams

Splitting an existing team has the advantage that the required know-how is already available in the team and the team can get productive faster. The drawback is that already working teams are torn apart.

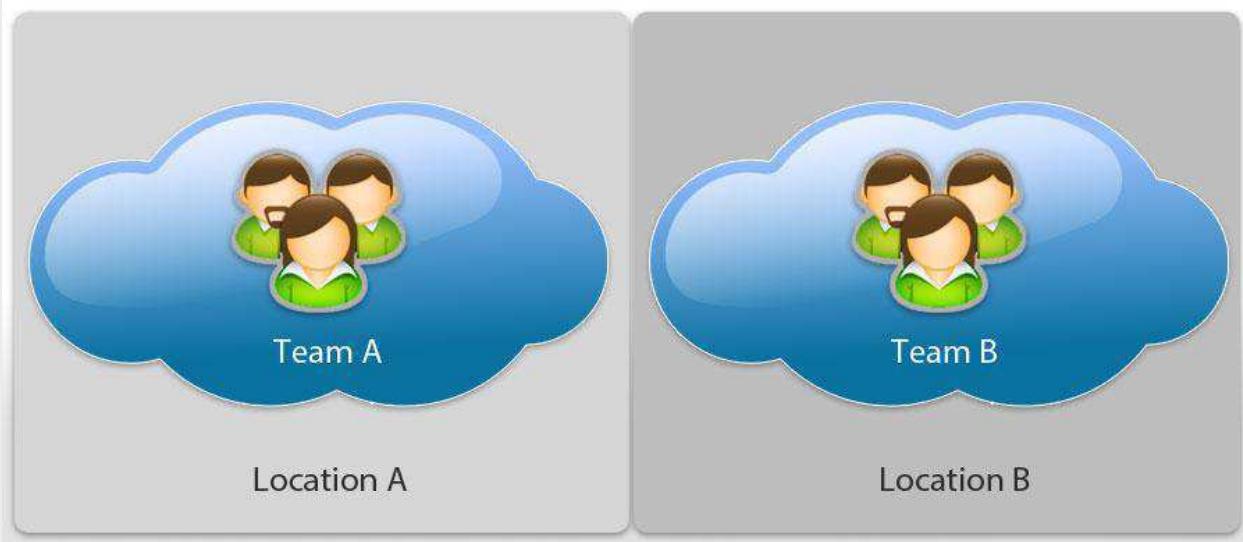
When adding completely new teams, these existing teams can continue with their work without much disruption. However, it will take longer to build up the necessary system-know-how in the new Scrum Team.

Independent from the decision how to add new teams the following rules should be followed:

- Start with a small number of teams
- Always wait until a foundation is build and the teams have stabilized
- Increase the number of teams in small steps

Project Organization – Distributed Teams

Even more complicated it will get if these new teams are distributed over multiple locations. Now also more often communication obstacles will occur and special care has to be taken to introduce and involve all team members adequately.

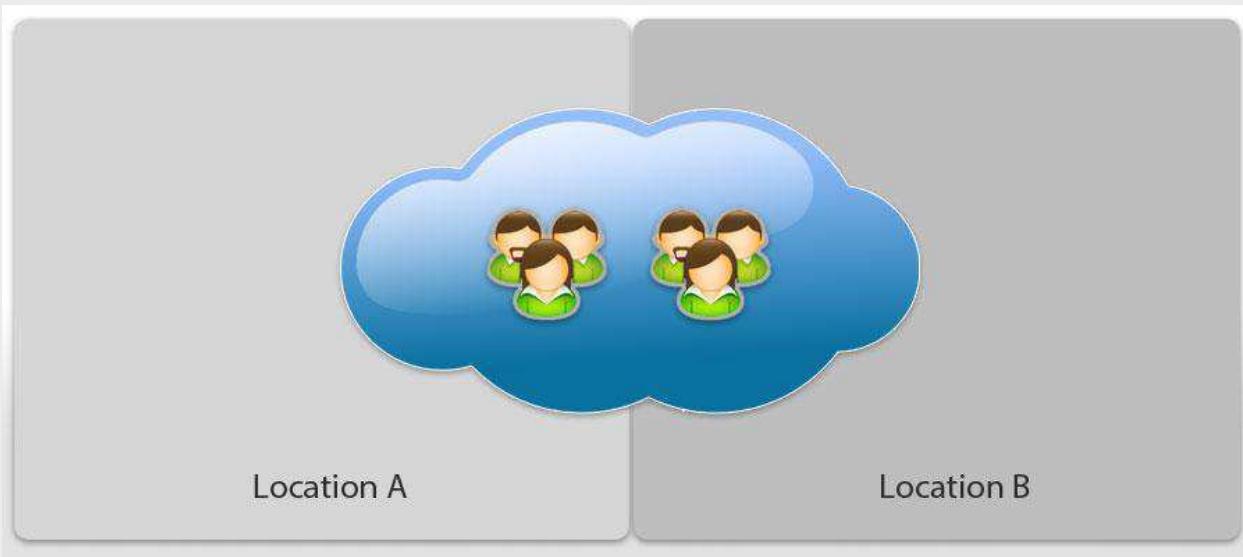


Multiple Teams in Multiple Locations

To make sure that new team members are introduced adequately and build-up the required knowledge as fast as possible, new team members could e.g. temporarily added to an existing team, preferably even in another location. With this approach the know-how is transferred and personal relationships with people in other teams and locations build.

Virtual Teams

Another possibility for distribution is that the team itself is spread over multiple locations. Such a team is called a "virtual team".



Virtual Teams

The main challenge here is to ensure good communication between the team members since some people might not be able to physically participate in meetings or have no access to "communication helpers" like the Sprint Board.

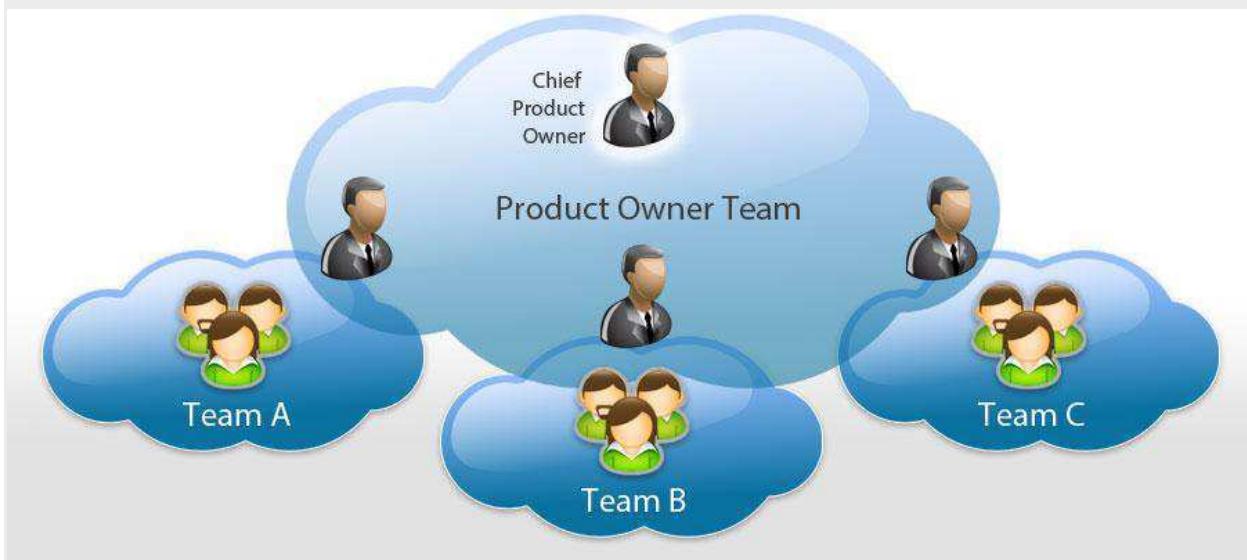
One possibility would be to use collaboration and/or communication tools. Co-located people could then e.g. be added to meetings via video conferencing or the meetings could even be performed completely in a 'virtual room' provided by most collaboration platforms.

Scrum Product Owner Team

Proper communication between the Scrum Product Owner and the team is crucial for successful implementation of the project. To ensure that the Scrum Product Owner is always available to the team, it is often necessary to have multiple Scrum Product Owners working together. Ideally there is one dedicated Scrum Product Owner per team.

The Scrum Product Owners should then build a dedicated Scrum Product Owner Team to effectively work together. One of the Scrum Product Owners should be assigned the role of the 'Chief Scrum Product Owner' who is responsible to ensure that the product is developed in a coordinated fashion.

Since this team is responsible for the complete requirement engineering it might also be beneficial to add other roles and stakeholder like architects or customer representatives.



Scrum Product Owner Team

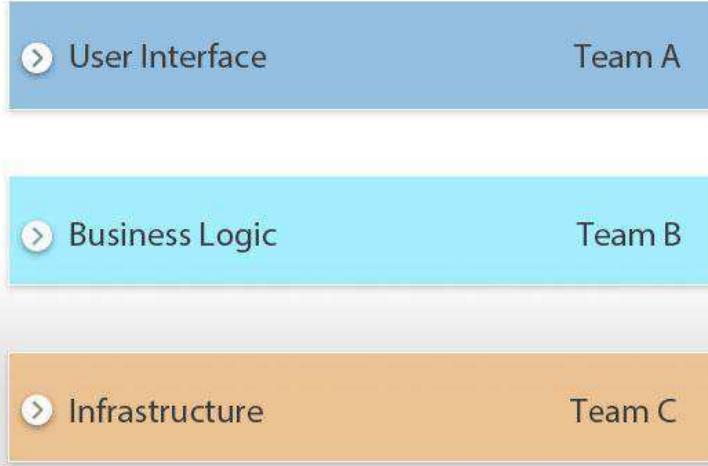
All Scrum Product Owners should work within a single Scrum Product Backlog containing all stories relevant for the project.

Component or Feature Teams

When distributing work we can slice the teams in different manners: as Component or Feature teams.

Component Teams

When using Component teams each team is only responsible for the implementation of dedicated components in the system. To finish a user story it is in most cases necessary to split the stories into smaller pieces that could be implemented within a single component. The resulting dependencies between the teams make integration on a regular base necessary. In many cases a single user story cannot be finished within a single sprint as implementation in one team depends on the results of other stories in other team that are not yet available. This is called "Pipelining" and should be avoided as far as possible.

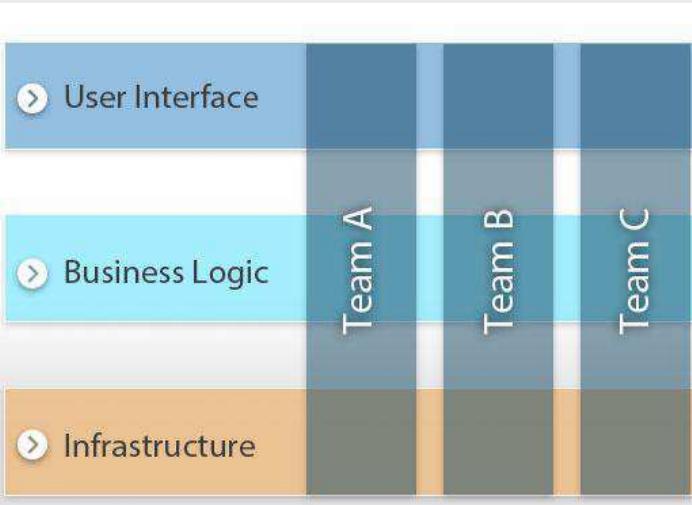


Component Teams

Advantage of using component teams is that it is easier to ensure proper architecture of the system. On the other hand people specialize only on small parts of the system and knowledge about the system as a whole might get lost. Without this knowledge local optimization might take place since the team might sometimes make decisions that are optimized for the single component but better solutions from a system perspective could have been made.

Feature Teams

Feature teams are fully responsible for implementation of user stories as contained in the Scrum Backlog. The team is not longer sliced along system components but implement everything what is necessary to finish the story.



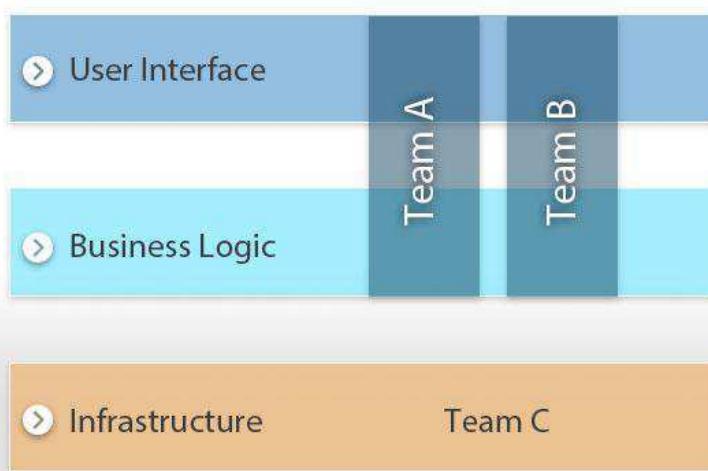
Feature Teams

Feature teams have to be interdisciplinary and ideally can act completely autonomous. The advantage is that system-knowledge is spread and integration is easier. However it is more difficult to ensure consistency of the

system architecture and it might be difficult or takes time to ensure that enough knowledge is available in all teams.

Component and Feature Teams

In reality many larger projects use both: dedicated Component teams and Feature teams.



Component and Feature Teams

Team C is a Component team and provides necessary infrastructure services to the other teams that are used as Feature teams. Team C does not directly implement user stories but get the requirements from the user stories committed by the Feature teams. This allows minimizing the number of required people with expert knowledge (e.g. databases know-how).

The Scrum Master in a Distributions Environment

In a distributions environment the role of the Scrum Master is even more important as such setups usually have more impediments that require the Scrum Masters attention and effort.

One important rule is that the Scrum Master has to be located where the team is otherwise it will be difficult to remove the obstacles in daily work. There should always be a primary Scrum Master, but in virtual teams it might also be an option that on the remote site one person acts as a local Scrum Master.

Chapter 22: MULTI-TEAM COORDINATION & PLANNING

Scrum of Scrum

To coordinate the different Scrum Teams "Scrum of Scrum"-Meetings can be used. It is similar to the Daily Scrum but the focus here is on team-level.

The meeting takes place every day and should be limited to e.g. 15 minutes. Each team sends out one member to participate and answer the following questions:

- What did the team finish?
- What does the team plan to finish today?
- Are there any impediments?

The answers should concentrate on the things that impact any other team.

The Chief Scrum Product Owner should moderate the meeting. Participation should not only be the Scrum Master of the teams, one approach could also be to shift participation on a daily basis within the team.

Common Sprint Reviews

Instead of multiple small Sprint Review meetings a common Sprint Review with all teams could also be used. This will show everyone what has been done within the Sprint and what the status of the project is.

Common Sprint Retrospectives

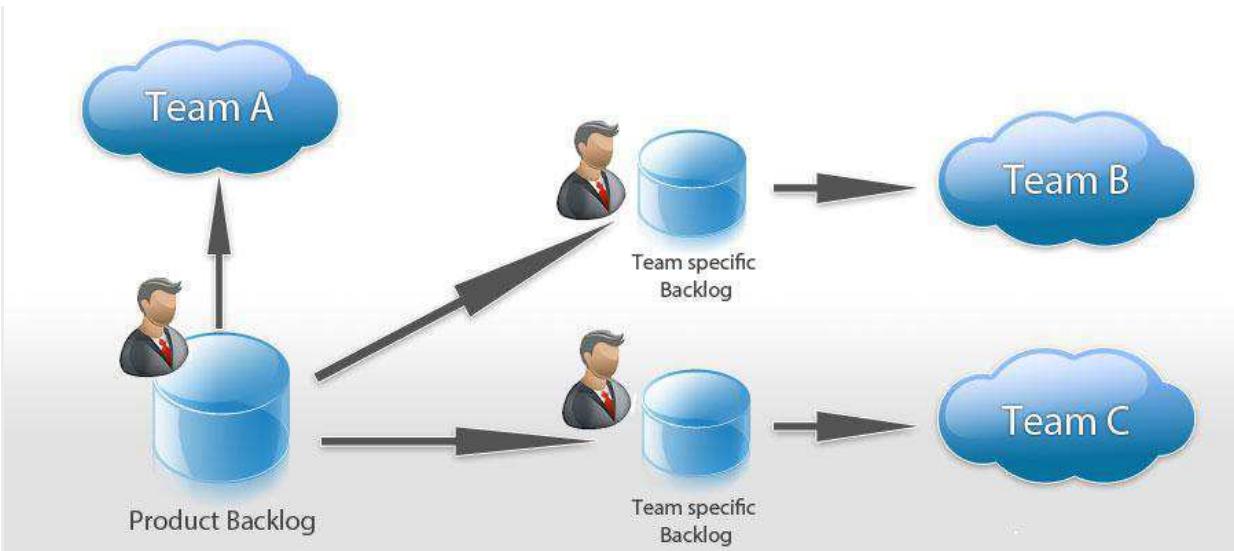
For the Sprint Retrospective there are two possibilities. The first possibility is that each team has their own Sprint Retrospective meeting followed by a Common Sprint Retrospective where all results are discussed that concern multiple teams.

Another possibility is that topics are collected, selected and then worked on in smaller groups with members of all teams. This approach will take more time but the advantage is that members from multiple teams are closely working together.

Multi-Team Planning - Scrum Product Backlog

Even when working with multiple teams it is important to keep only one common Scrum Product Backlog for all teams. The Scrum Product Backlog shall be maintained by the Chief Scrum Product Owner but is filled by all Scrum Product Owners.

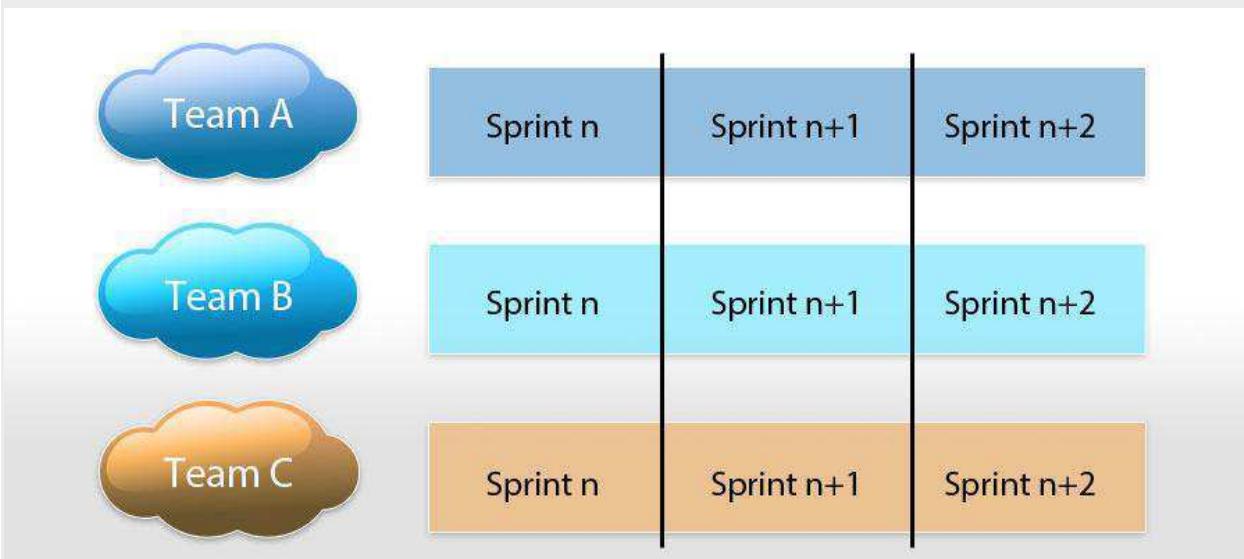
If necessary the items in the Scrum Product Backlog can be broken down into more team-specific stories and maintained in a team-specific Scrum Product Backlog, e.g. only the relevant parts for an infrastructure team. In this case references between these Scrum Product Backlogs shall exist.



Team-specific Backlogs

Sprint Scheduling

In a distributed Scrum environment it is possible to choose how to synchronize the different teams. One possibility is to use synchronous sprints. In this case all teams start and end their sprints on the same day.

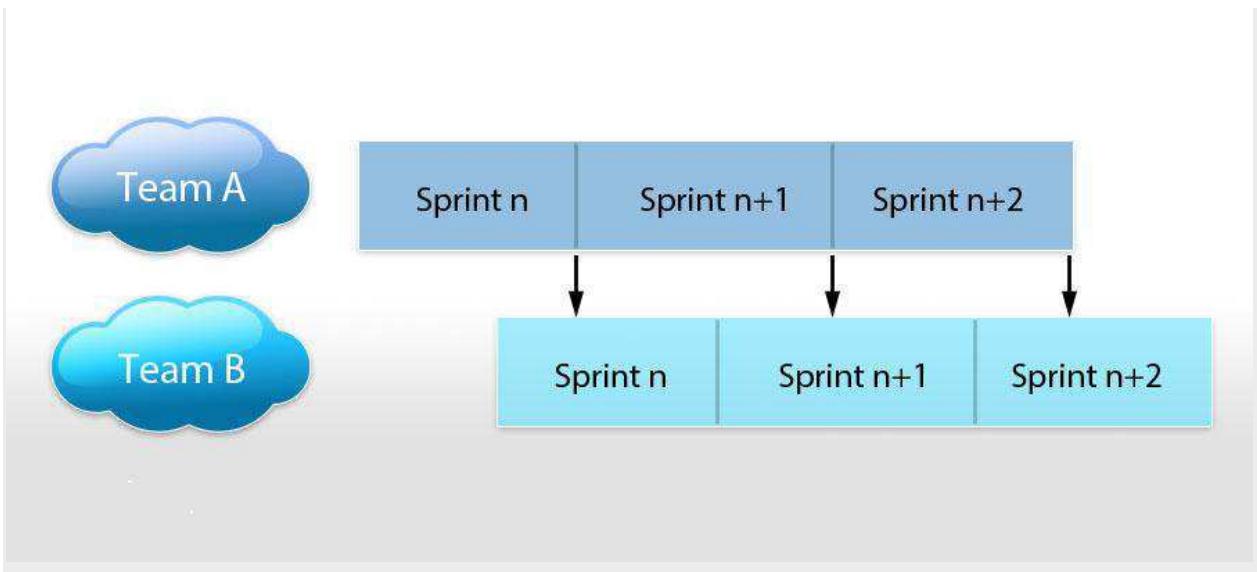


Synchronous Sprints

Synchronous Sprints are the easiest approach since it makes communication and coordination simpler.

Another possibility is to use asynchronous sprints. Here the sprints do not start on the same day. Using asynchronous sprints has the advantage that not all meetings have to be performed on the same day and makes it possible for e.g. the Scrum Product Owners to participate in more of these meetings.

When one team provides services to other teams asynchronous Sprints might also be a good option because here the results of the service-providing team is already available during the development time of the other teams and could be integrated.



Asynchronous Sprints

Effort Estimations

It is important that all items in the Scrum Product Backlog are estimated using the same base for estimation. If story points are used all teams have to agree on the same metric and a common scale to use. If Component Teams are used it is important that members of all teams participate in the estimation to ensure that all efforts are covered.

Chapter 23: SCRUM RELEASE PLANNING

A very high-level plan for multiple Sprints (e.g. three to twelve iteration) is created during the Release planning. It is a guideline that reflects expectations about which features will be implemented and when they are completed. It also serves as a base to monitor progress within the project. Releases can be intermediate deliveries done during the project or the final delivery at the end.

To create a Release Plan the following things have to be available:

- A prioritized and estimated Scrum Product Backlog
- The (estimated) velocity of the Scrum Team
- Conditions of satisfaction (goals for the schedule, scope, resources)

Depending on the type of project (feature- or date-driven) the release plan can be created in different ways:

If the project is feature-driven, the sum of all features within in a release can be divided by the expected velocity. This will then result in the number of sprints needed to complete the requested functionality.

Release Goals:

- (1) First Release after user able to login/logout
- (2) Second Release after user can manage items
- (3) Final Release when all stories are done

First Release

Stories 7, 1, 10 -> Estimation = 5 Points
Estimation / Velocity = 5/3 -> 2 Sprints

Second Release

Stories 9, 2, 4, 3, 5 -> Estimation = 15 Points
Estimation / Velocity = 15/3 -> 5 Sprints

Final Release

All Stories -> Estimation = 30 Points
Estimation / Velocity = 30 / 3 -> 10 Sprints

ToDo List

ID	STORY	ESTIMATION	PRIORITY
7	As an unauthorized User I want to create account	3	1
1	As an unauthorized User I want to login	1	2
10	As an unauthorized User I want to logout	1	3
9	Create script to purge database	1	5
2	As an authorized User I want to see the List of items so that I can select one	2	5
4	As an authorized User I want to add a new Item so that it appears in the list	5	6
3	As an authorized User I want to delete the Selected item	2	7
5	As an authorized User I want to edit the Selected item	5	8
6	As an authorized User I want to set a reminder for a selected item so that I am reminded when item is due	8	9
8	As an administrator I want to see the list of accounts on login	2	10
TOTAL		30	
Velocity: 3 Points / Sprint			

Releases after Sprint 2, 7 and 10

Release Plan for a feature-driven project

If the project is date-driven we can simply multiply the velocity by the number of Sprints and we'll get the total work that can be completed within the given timeline.

ToDo List				
ID	STORY	ESTIMATION	PRIORITY	
7	As an unauthorized User I want to create account	3	1	
1	As an unauthorized User I want to login	1	2	
10	As an unauthorized User I want to logout	1	3	
9	Create script to purge database	1	5	
2	As an authorized User I want to see the List of items so that I can select one	2	5	
4	As an authorized User I want to add a new Item so that it appears in the list	5	6	Release
3	As an authorized User I want to delete the Selected item	2	7	Release
5	As an authorized User I want to edit the Selected item	5	8	Release
6	As an authorized User I want to set a reminder for a selected item so that I am reminded when item is due	8	9	Release
8	As an administrator I want to see the list of accounts on login	2	10	Release
TOTAL		30		
Velocity: 3 Points / Sprint		Sprint Length 2 weeks		

Release Plan for a date-driven project

Like the Scrum Product Backlog the Release plan is not a static plan. It will change during the whole project when new knowledge is available and e.g. entries in the Scrum Product Backlog are changed and re-estimated. Therefore the Release Plan should be revisited and updated in regular intervals, e.g. after each Sprint.

Chapter 24: AGILE MANIFESTO

In February 2001, 17 software developers met at the Snowbird, Utah resort, to discuss lightweight development methods. They published the *Manifesto for Agile Software Development* to define the approach now known as agile software development. Some of the manifesto's authors formed the Agile Alliance, a nonprofit organization that promotes software development according to the manifesto's principles.

The Agile Manifesto reads, in its entirety, as follows:

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.

The meanings of the manifesto items on the left within the agile software development context are described below:

- Individuals and interactions – in agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.
- Working software – working software will be more useful and welcome than just presenting documents to clients in meetings.
- Customer collaboration – requirements cannot be fully collected at the beginning of the software development cycle, therefore continuous customer or stakeholder involvement is very important.
- Responding to change – agile development is focused on quick responses to change and continuous development.

The Agile Manifesto is based on twelve principles:

1. Customer satisfaction by rapid delivery of useful software
2. Welcome changing requirements, even late in development
3. Working software is delivered frequently (weeks rather than months)
4. Working software is the principal measure of progress
5. Sustainable development, able to maintain a constant pace
6. Close, daily cooperation between business people and developers
7. Face-to-face conversation is the best form of communication (co-location)
8. Projects are built around motivated individuals, who should be trusted
9. Continuous attention to technical excellence and good design
10. Simplicity—the art of maximizing the amount of work not done—is essential
11. Self-organizing teams
12. Regular adaptation to changing circumstances

The well-known background picture of the Agile Manifesto website was taken by Ward Cunningham, who wanted to capture the moment during the weekend meeting at Snowbird.

In 2005, a group headed by Alistair Cockburn and Jim Highsmith wrote an addendum of project management principles, the Declaration of Interdependence, to guide software project management according to agile development methods.

In 2009, a movement spearheaded by Robert C Martin wrote an extension of software development principles, the Software Craftsmanship Manifesto, to guide agile software development according to professional conduct and mastery