Software Development with Scrum

Andrea Polini

Software Project Management
MSc in Computer Science
University of Camerino

What is Scrum?

Scrum

is an Agile methodology – i.e. a collection of practices combined with ideas, advice, experience (BoK) – to software system development

- certainly the most adopted agile methodology

Scrum values

Every company has a culture (e.g. separation of duties, transparency). When the culture mathches agile values and principles the adoption of agile methodologies will be much more successful. Scrum has its own values:

- Courage
- Commitment
- Respect
- Focus
- Openess

Courage

Team members have the courage to stand up for the project . . . Scrum teams have the courage to live by values and principles that benefit the project. It takes courage to ward off the constant pushback from a company whose values clash with the Scrum and agile values.

Courage

Team members have the courage to stand up for the project . . . Scrum teams have the courage to live by values and principles that benefit the project. It takes courage to ward off the constant pushback from a company whose values clash with the Scrum and agile values.

Commitment

Each person is committed to the project's goals. . . team has the authority to make decisions in order to meet project' goals, and everyone can influence how the project is planned and executed

Commitment

Each person is committed to the project's goals. . . team has the authority to make decisions in order to meet project' goals, and everyone can influence how the project is planned and executed

Respect

Team members respect each other . . . then they trust each other to do a good job with the work they've taken on.
The SM should find ways to increase mutual respect in the team.

Respect

Team members respect each other . . . then they trust each other to do a good job with the work they've taken on.
The SM should find ways to increase mutual respect in the team.

Focused

Everyone is focused on the work . . . a TM working on a sprint should not be distracted by other activities for the duration of the sprint (full time assignment). Switching people among activities lead to waste of time and money

Not attending a formative activity does not put in danger the career of the TM

Focused

Everyone is focused on the work . . . a TM working on a sprint should not be distracted by other activities for the duration of the sprint (full time assignment). Switching people among activities lead to waste of time and money

Not attending a formative activity does not put in danger the career of the TM

Openess

The teams value openness. . . when you're working on a Scrum team, everyone else on the team should always be aware of what you're working on and how it moves the project toward its current goals.

In many company you can observe a culture discouraging transparency. Rigid hierarchy that depends on opaqueness

Openess

The teams value openness. . . when you’re working on a Scrum team, everyone else on the team should always be aware of what you’re working on and how it moves the project toward its current goals.

In many company you can observe a culture discouraging transparency. Rigid hierarchy that depends on opaqueness

Which are the most important characteristics?

Scrum embodies the following practices:

- Three main roles in Scrum: Product Owner – PO, Scrum Master – SM, Team Member – TM
- The product owner creates and mantains a product backlog
- The team runs timeboxed month-long sprints. The requirements for the current sprint are called the sprint backlog
- The team meets for a daily standup meeting in which everyone talks through the work they did the day before, the work they plan to do today, and any obstacles in their way
- the Scrum master keeps the project rolling by working with the team to get past “roadblocks”. He/she drives the sprint review and the retrospective

Better-than-not-doing-it

It is possible to adopt all of these practices still missing to become an Agile team

Which are the most important characteristics?

Scrum embodies the following practices:

- Three main roles in Scrum: Product Owner – PO, Scrum Master – SM, Team Member – TM
- The product owner creates and mantains a product backlog
- The team runs timeboxed month-long sprints. The requirements for the current sprint are called the sprint backlog
- The team meets for a daily standup meeting in which everyone talks through the work they did the day before, the work they plan to do today, and any obstacles in their way
- the Scrum master keeps the project rolling by working with the team to get past “roadblocks”. He/she drives the sprint review and the retrospective

Better-than-not-doing-it

It is possible to adopt all of these practices still missing to become an Agile team

Which are the most important characteristics?

Scrum embodies the following practices:

- Three main roles in Scrum: Product Owner – PO, Scrum Master – SM, Team Member – TM
- The product owner creates and mantains a product backlog
- The team runs timeboxed month-long sprints. The requirements for the current sprint are called the sprint backlog
- The team meets for a daily standup meeting in which everyone talks through the work they did the day before, the work they plan to do today, and any obstacles in their way
- the Scrum master keeps the project rolling by working with the team to get past “roadblocks”. He/she drives the sprint review and the retrospective

Better-than-not-doing-it

It is possible to adopt all of these practices still missing to become an Agile team

Which are the most important characteristics?

Scrum embodies the following practices:

- Three main roles in Scrum: Product Owner – PO, Scrum Master – SM, Team Member – TM
- The product owner creates and mantains a product backlog
- The team runs timeboxed month-long sprints. The requirements for the current sprint are called the sprint backlog
- The team meets for a daily standup meeting in which everyone talks through the work they did the day before, the work they plan to do today, and any obstacles in their way
- the Scrum master keeps the project rolling by working with the team to get past “roadblocks”. He/she drives the sprint review and the retrospective

Better-than-not-doing-it

It is possible to adopt all of these practices still missing to become an Agile team

Which are the most important characteristics?

Scrum embodies the following practices:

- Three main roles in Scrum: Product Owner – PO, Scrum Master – SM, Team Member – TM
- The product owner creates and mantains a product backlog
- The team runs timeboxed month-long sprints. The requirements for the current sprint are called the sprint backlog
- The team meets for a daily standup meeting in which everyone talks through the work they did the day before, the work they plan to do today, and any obstacles in their way
- the Scrum master keeps the project rolling by working with the team to get past “roadblocks”. He/she drives the sprint review and the retrospective

Better-than-not-doing-it

It is possible to adopt all of these practices still missing to become an Agile team

Which are the most important characteristics?

Scrum embodies the following practices:

- Three main roles in Scrum: Product Owner – PO, Scrum Master – SM, Team Member – TM
- The product owner creates and mantains a product backlog
- The team runs timeboxed month-long sprints. The requirements for the current sprint are called the sprint backlog
- The team meets for a daily standup meeting in which everyone talks through the work they did the day before, the work they plan to do today, and any obstacles in their way
- the Scrum master keeps the project rolling by working with the team to get past "roadblocks". He/she drives the sprint review and the retrospective

Better-than-not-doing-it

It is possible to adopt all of these practices still missing to become an Agile team

Roles – Scrum Master

The SM is not the traditional PM in a command-and-control project. The SM does not own and mantain the plan. Guiding the team is his main responsibility. The SM:

- drives the team in the adoption and usage of scrum practices
- protects the team from “unfair” requests from the PO
- helps TMs to feel owenership for the project

Roles – Product Owner

- The PO is the one who made the commitment to the company
- The PO is generally acquainted with the business domain of the project
- should get the TMs able to understand the goals of the project, so that they can commit to the project itself
- s/he meets everyday with the TMs and takes day-to-day decisions to drive the project, possibly changing the backlog
- The PO manages the PB prioritizing the items in it
- brings the view of users and stakeholders in the project.

Commitment

Promise made by people to gets certain things done, usaully by a certain time

Product Owner responsibilities

A good product owner...

- Understand what the company needs most, and bring that knowledge back to the team
- Understand what software features the team can potentially deliver
- Figure out which features are more valuable to the company, and which are less valuable
- Work with the team to figure out which features are easier to build, and which are harder
- Use that knowledge of value, difficulty, uncertainty, complexity, etc. to help the team choose the right features to build in each sprint
- Bring that knowledge back to the rest of the company, so they can do what they need to do to prepare for the next release of the software

Commitment vs. involvment

A Pig and a Chicken are walking down the road

The Chicken says: Hey Pig, I was thinking we should open a restaurant!
Pig replies: Hm, maybe; what would we call it?
The Chicken responds: How about 'ham-n-eggs'?
The Pig thinks for a moment and says: No, thanks. I'd be committed, but you'd only be involved!

Commitment refers to each role and should be fostered by each other

Commitment vs. involvment

A Pig and a Chicken are walking down the road

The Chicken says: Hey Pig, I was thinking we should open a restaurant!
Pig replies: Hm, maybe; what would we call it?
The Chicken responds: How about 'ham-n-eggs'?
The Pig thinks for a moment and says: No, thanks. I'd be committed, but you'd only be involved!

Commitment refers to each role and should be fostered by each other

Sprints and Planning

Scrum is based on:

- Iterative and Incremental approach to software development
- Effective project planning
- Frequent delivery of running software (value to customer)
- User needs defined in User stories

Building the team

If you are asked to organize the team for the next project, and you would like to give a try to SCRUM, consider that:

- People do not come on board if they are not convinced. In teams accustomed to WF like processes people can be reluctant to accept the challenge. Known vs. unknown
- A possible framework
  - Establish a sense of urgengy – challenge real and reachable
  - form a powerful guiding coalition – You need support by the management
  - Create a vision – how the future will cahnge
  - Communicate the vision
  - Empower others to act on the vision
  - Plan for and create a short term wins
  - Consolidate improvements and produce still more changes
  - institutionalize new approaches
- Elevate the goals
- Try hard and be persuasive

Teams generally include within 5 and 7 plus 1-2 consultants if needed

Consultants vs. Team members

- Team members are generally multitalented and have variegated interests
- Consultants are generally much more specialised and they are considered gurus in their specific competence
- Team member are full time on the project
- Consultants are enrolled when needed and they can enter the project for a single sprint. When involved they are peer with respect to team members

Consultants vs. Team members

- Team members are generally multitalented and have variegated interests
- Consultants are generally much more specialised and they are considered gurus in their specific competence
- Team member are full time on the project
- Consultants are enrolled when needed and they can enter the project for a single sprint. When involved they are peer with respect to team members

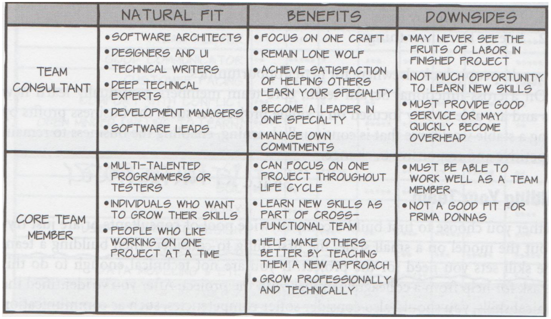
Consultants vs. Team members

- Team members are generally multitalented and have variegated interests
- Consultants are generally much more specialised and they are considered gurus in their specific competence
- Team member are full time on the project
- Consultants are enrolled when needed and they can enter the project for a single sprint. When involved they are peer with respect to team members

Consultants vs. Team members

- Team members are generally multitalented and have variegated interests
- Consultants are generally much more specialised and they are considered gurus in their specific competence
- Team member are full time on the project
- Consultants are enrolled when needed and they can enter the project for a single sprint. When involved they are peer with respect to team members

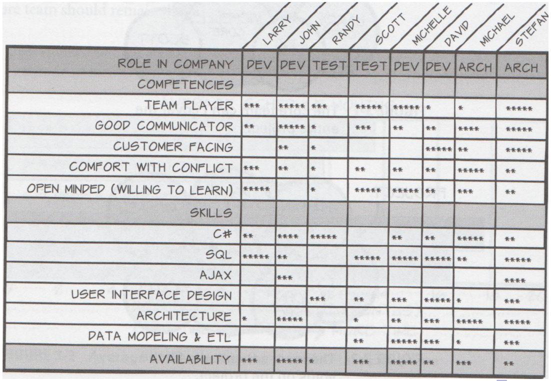
Consultants vs. Team members



	NATURAL FIT	BENEFITS	DOWNSIDES
TEAM CONSULTANT	• SOFTWARE ARCHITECTS • DESIGNERS AND UI • TECHNICAL WRITERS • DEEP TECHNICAL EXPERTS • DEVELOPMENT MANAGERS • SOFTWARE LEADS	• FOCUS ON ONE CRAFT • REMAIN LONE WOLF • ACHIEVE SATISFACTION OF HELPING OTHERS LEARN YOUR SPECIALITY • BECOME A LEADER IN ONE SPECIALTY • MANAGE OWN COMMITMENTS	• MAY NEVER SEE THE FRUITS OF LABOR IN FINISHED PROJECT • NOT MUCH OPPORTUNITY TO LEARN NEW SKILLS • MUST PROVIDE GOOD SERVICE OR MAY QUICKLY BECOME OVERHEAD
CORE TEAM MEMBER	• MULTI-TALENTED PROGRAMMERS OR TESTERS • INDIVIDUALS WHO WANT TO GROW THEIR SKILLS • PEOPLE WHO LIKE WORKING ON ONE PROJECT AT A TIME	• CAN FOCUS ON ONE PROJECT THROUGHOUT LIFE CYCLE • LEARN NEW SKILLS AS PART OF CROSS-FUNCTIONAL TEAM • HELP MAKE OTHERS BETTER BY TEACHING THEM A NEW APPROACH • GROW PROFESSIONALLY AND TECHNICALLY	• MUST BE ABLE TO WORK WELL AS A TEAM MEMBER • NOT A GOOD FIT FOR PRIMA DONNAS

Assessing needed Skills

To build the team list the competences and skills you need on the rows of a table. Add a column for each potential team candidate rating each quality with a mark within 1 and 5. Cover all the skill and competences you need with team members and consultants trying to get the maximum sum.



	LARRY	JOHN	RANDY	SCOTT	MICHELLE	DAVID	MICHAEL	STEFAN
ROLE IN COMPANY	DEV	DEV	TEST	TEST	DEV	DEV	ARCH	ARCH
COMPETENCIES								
TEAM PLAYER	***	*****	***	*****	*****	*	*	*****
GOOD COMMUNICATOR	**	*****	*	***	**	**	****	*****
CUSTOMER FACING		**	*			*****	**	*****
COMFORT WITH CONFLICT	***	**	*	**	**	**	*****	**
OPEN MINDED (WILLING TO LEARN)	*****		*	*****	*****	**	***	**
SKILLS								
C#	**	****	*****		**	**	*****	**
SQL	*****	**		*****	*****	*****	**	*****
AJAX		***						****
USER INTERFACE DESIGN			***	**	***	*****	*	***
ARCHITECTURE	*	*****		*	**	***	*****	*****
DATA MODELING & ETL				**	*****	***	*	***
AVAILABILITY	***	*****	*	***	*****	**	***	**

Scrum basic rules – Sprints

1. **Duration**: typically within 2 and 4 weeks, or a month.
2. **Characteristics**: timeboxed
3. **How to**: as soon as a TM recognizes he/she has overcommitted report to PO, that then will have to inform the users/stakeholder
4. Product backlog should be visible to everyone. Generally sprints cannot be stopped (in case the PO)

Scrum basic rules – Sprint closing/review

1. The software is presented to the stakeholders (only the running part of the system, no diagrams are admitted)
2. stakeholders are asked to provide their opinions and feedbacks. The PO possibly update the PB.

Scrum basic rules – After the sprint

1. Retrospective meeting is held. Each one answers to the following questions:
   - What went well?
   - What can improve in the future?
2. SM takes notes and possibly adds items to the PB for non funtional items

Sprint planning

First day of a Sprint:

1. **Attendance**: SM, PO, TMs
2. Meeting divided in two different parts, each one lasting 4 hours – Exploration and Operative plan – duration can be reduced according to the duration of the sprint
3. PO comes with an already prioritized backlog
4. 1st part of the meeting: PO and TMs works together to select items to be delivered during the next sprint. The sprint backlog is formed.
5. 2nd part of the meeting: TMs, with the help of the PO, figure out the individual tasks needed to implement the selected items.

Sprint planning

First day of a Sprint:

1. **Attendance**: SM, PO, TMs
2. Meeting divided in two different parts, each one lasting 4 hours – Exploration and Operative plan – duration can be reduced according to the duration of the sprint
3. PO comes with an already prioritized backlog
4. 1st part of the meeting: PO and TMs works together to select items to be delivered during the next sprint. The sprint backlog is formed.
5. 2nd part of the meeting: TMs, with the help of the PO, figure out the individual tasks needed to implement the selected items.

Sprint planning

First day of a Sprint:

1. **Attendance**: SM, PO, TMs
2. Meeting divided in two different parts, each one lasting 4 hours – Exploration and Operative plan – duration can be reduced according to the duration of the sprint
3. PO comes with an already prioritized backlog
4. 1st part of the meeting: PO and TMs works together to select items to be delivered during the next sprint. The sprint backlog is formed.
5. 2nd part of the meeting: TMs, with the help of the PO, figure out the individual tasks needed to implement the selected items.

Sprint planning

First day of a Sprint:

1. **Attendance**: SM, PO, TMs
2. Meeting divided in two different parts, each one lasting 4 hours – Exploration and Operative plan – duration can be reduced according to the duration of the sprint
3. PO comes with an already prioritized backlog
4. 1st part of the meeting: PO and TMs works together to select items to be delivered during the next sprint. The sprint backlog is formed.
5. 2nd part of the meeting: TMs, with the help of the PO, figure out the individual tasks needed to implement the selected items.

Sprint planning

First day of a Sprint:

1. **Attendance**: SM, PO, TMs
2. Meeting divided in two different parts, each one lasting 4 hours – Exploration and Operative plan – duration can be reduced according to the duration of the sprint
3. PO comes with an already prioritized backlog
4. 1st part of the meeting: PO and TMs works together to select items to be delivered during the next sprint. The sprint backlog is formed.
5. 2nd part of the meeting: TMs, with the help of the PO, figure out the individual tasks needed to implement the selected items.

Determining Sprint Length

General Guidelines and Relevant Factors

- No one-fits-all solutions (from 1 to 4 weeks)
- Consider customer availability and location
- Consider need for feedbacks and requirements clarity
- Consider team ability to decompose work
- Consider project duration
- Consider established engineering practice within the team

Determining Sprint Length

General Guidelines and Relevant Factors

- No one-fits-all solutions (from 1 to 4 weeks)
- Consider customer availability and location
- Consider need for feedbacks and requirements clarity
- Consider team ability to decompose work
- Consider project duration
- Consider established engineering practice within the team

Determining Sprint Length

General Guidelines and Relevant Factors

- No one-fits-all solutions (from 1 to 4 weeks)
- Consider customer availability and location
- Consider need for feedbacks and requirements clarity
- Consider team ability to decompose work
- Consider project duration
- Consider established engineering practice within the team

Determining Sprint Length

General Guidelines and Relevant Factors

- No one-fits-all solutions (from 1 to 4 weeks)
- Consider customer availability and location
- Consider need for feedbacks and requirements clarity
- Consider team ability to decompose work
- Consider project duration
- Consider established engineering practice within the team

Determining Sprint Length

General Guidelines and Relevant Factors

- No one-fits-all solutions (from 1 to 4 weeks)
- Consider customer availability and location
- Consider need for feedbacks and requirements clarity
- Consider team ability to decompose work
- Consider project duration
- Consider established engineering practice within the team

Determining Sprint Length

General Guidelines and Relevant Factors

- No one-fits-all solutions (from 1 to 4 weeks)
- Consider customer availability and location
- Consider need for feedbacks and requirements clarity
- Consider team ability to decompose work
- Consider project duration
- Consider established engineering practice within the team

User Stories

What is it?

A simple and quick description of a specific way that the user will use the software. Generally between one and four sentences long stays in a 3$\times$5 index card

- They permit to deliver value to the customer
- Reduce the risk of gold-plating

Can generally follow a template:

As a <*type of user*>, I want to <*specific action I'm taking*> so that <*what I want to happen as a result*>

User Stories

What is it?

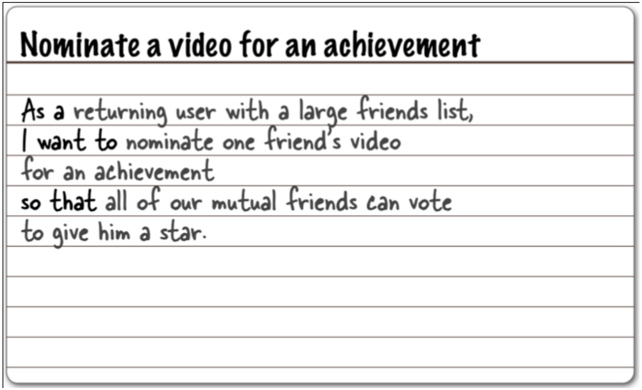
A simple and quick description of a specific way that the user will use the software. Generally between one and four sentences long stays in a 3$\times$5 index card

- They permit to deliver value to the customer
- Reduce the risk of gold-plating

Can generally follow a template:

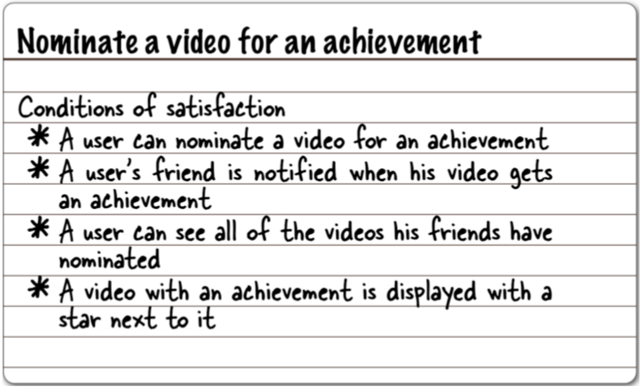
As a <*type of user*>, I want to <*specific action I'm taking*> so that <*what I want to happen as a result*>

User stories and satisfaction conditions


Nominate a video for an achievement

As a returning user with a large friends list,
I want to nominate one friend's video
for an achievement
so that all of our mutual friends can vote
to give him a star.

User stories and satisfaction conditions



Nominate a video for an achievement

Conditions of satisfaction

* A user can nominate a video for an achievement
* A user's friend is notified when his video gets an achievement
* A user can see all of the videos his friends have nominated
* A video with an achievement is displayed with a star next to it

Story points

Useful tool to assess the effort needed to elaborate a user story. The objective is to assign a value to each user story using a comparative analysis

1. value each story between 1 and 5 (or 10)
2. Or use Fibonacci numbers ($f_0 = 1, f_1 = 1, f_n = f_{n-1} + f_{n-2}$)

How?

- discussion among team members
- Planning poker

Story points

Useful tool to assess the effort needed to elaborate a user story.
The objective is to assign a value to each user story using a comparative analysis

1. value each story between 1 and 5 (or 10)
2. Or use Fibonacci numbers ($f_0 = 1, f_1 = 1, f_n = f_{n-1} + f_{n-2}$)

How?

- discussion among team members
- Planning poker

Story points – How to

Possible steps in planning session using story points:

1. Start with the most valuable user stories from the product backlog
2. Take a story in that list, ideally the smallest one, find a similarly sized story from a previous sprint, and assign it the same number of points.
3. Discuss with the team whether that estimate is accurate – discovering additional problems, work, or technical challenges increases the estimate; simplifying factors, reusing existing code, or scaling back what needs to be built decreases the estimate.
4. Keep going through the stories until you've accumulated enough points to fill the sprint.

The first time you apply the strategy use the highest number to mark the most complex (5) and the smallest one to mark the simplest one (1).

Story points qualities

- They are simple
- They are not magic
- The team is in control of them
- They get your team talking about estimates
- Developers are not scared of them – they do not immediately relate to time
- They help the team discover exactly what a story means
- They help everyone on the team become genuinely committed

Story points qualities

- They are simple
- They are not magic
- The team is in control of them
- They get your team talking about estimates
- Developers are not scared of them – they do not immediately relate to time
- They help the team discover exactly what a story means
- They help everyone on the team become genuinely committed

Story points qualities

- They are simple
- They are not magic
- The team is in control of them
- They get your team talking about estimates
- Developers are not scared of them – they do not immediately relate to time
- They help the team discover exactly what a story means
- They help everyone on the team become genuinely committed

Story points qualities

- They are simple
- They are not magic
- The team is in control of them
- They get your team talking about estimates
- Developers are not scared of them – they do not immediately relate to time
- They help the team discover exactly what a story means
- They help everyone on the team become genuinely committed

Story points qualities

- They are simple
- They are not magic
- The team is in control of them
- They get your team talking about estimates
- Developers are not scared of them – they do not immediately relate to time
- They help the team discover exactly what a story means
- They help everyone on the team become genuinely committed

Story points qualities

- They are simple
- They are not magic
- The team is in control of them
- They get your team talking about estimates
- Developers are not scared of them – they do not immediately relate to time
- They help the team discover exactly what a story means
- They help everyone on the team become genuinely committed

Story points qualities

- They are simple
- They are not magic
- The team is in control of them
- They get your team talking about estimates
- Developers are not scared of them – they do not immediately relate to time
- They help the team discover exactly what a story means
- They help everyone on the team become genuinely committed

Team velocity and sprint planning

Velocity

Measure of the ability of the team to satisfy story points within a single sprint.

Sprint planning:

- Select the most valuable stories for which the sum is smaller than the velocity (stay below)

Sprint planning – second meeting

One of the most common ways for Scrum teams to plan out the actual work for the team is to add cards for individual development tasks. These tasks can be anything that the team actually does:

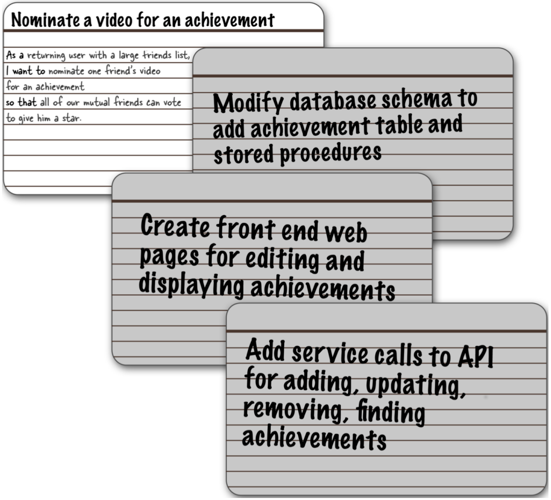
- write code
- create design and architecture
- build tests
- install operating systems
- design and build databases
- deploy software to production servers
- run usability tests
- all of those other things that teams actually do every day to build and release software.

Second meeting – How to?

The meeting can follow the following principles:

- SM takes the first user story and start the discussion
- everyone participates and proposes simple needed tasks (2 days most) to reach the objective
- Each task is written on a card. Set of tasks are grouped with the corresponding user story
- if meeting run out of time non discusses US get only one task card
- all stories and tasks are added to the “to do” column of the spring backlog

User stories and activities



Nominate a video for an achievement

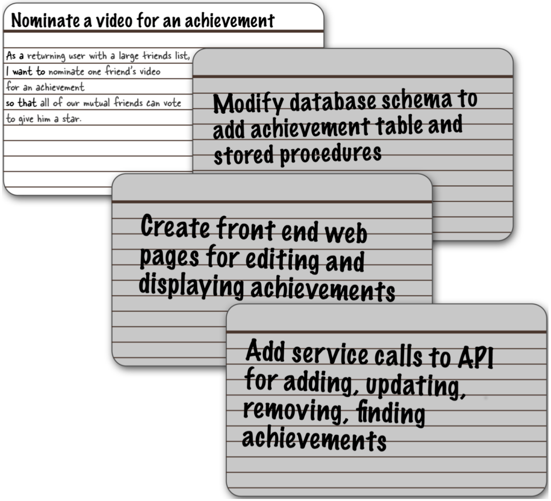
As a returning user with a large friends list,
I want to nominate one friend's video
for an achievement
so that all of our mutual friends can vote
to give him a star.

Modify database schema to add achievement table and stored procedures

Create front end web pages for editing and displaying achievements

Add service calls to API for adding, updating, removing, finding achievements

Story decomposition



Nominate a video for an achievement

As a returning user with a large friends list,
I want to nominate one friend's video
for an achievement
so that all of our mutual friends can vote
to give him a star.

Modify database schema to add achievement table and stored procedures

Create front end web pages for editing and displaying achievements

Add service calls to API for adding, updating, removing, finding achievements

Epics, themes, stories, tasks

Writing stories and tasks of the right size can mean the difference between succeeding with Scrum and failing miserably

Setting the stage

Three flavors stories – epics, themes, stories

- Redesign the user interface for the document editor
- Redesign the menu bar for the document editor
- Redesign the edit menu

Tasks should not need more than two days to complete, stories less than a sprint, themes more then a sprint, epics several sprints

Epics, themes, stories, tasks

Writing stories and tasks of the right size can mean the difference between succeeding with Scrum and failing miserably

Setting the stage

Three flavors stories – epics, themes, stories

- Redesign the user interface for the document editor
- Redesign the menu bar for the document editor
- Redesign the edit menu

Tasks should not need more than two days to complete, stories less than a sprint, themes more then a sprint, epics several sprints

Epics, themes, stories, tasks

Writing stories and tasks of the right size can mean the difference between succeeding with Scrum and failing miserably

Setting the stage

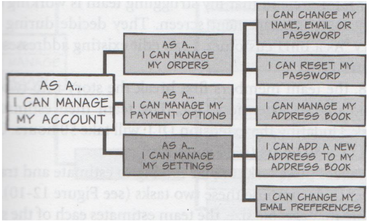
Three flavors stories – epics, themes, stories

- Redesign the user interface for the document editor
- Redesign the menu bar for the document editor
- Redesign the edit menu

Tasks should not need more than two days to complete, stories less than a sprint, themes more then a sprint, epics several sprints

Decomposing stories and tasks

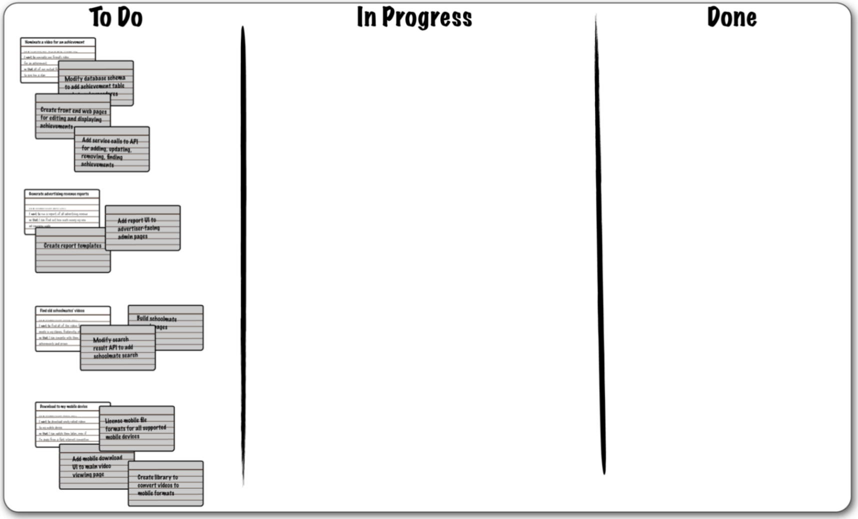
How do you know when the story has the right size?



Rule of thumbs for the size of User Stories

- A story is a the right size when it describes the smallest action that a user would tipically want to do, or it is the smallest piece of functionality with business value
- Can the team estimate the product backlog in story points?
- Is there clarity on the stories in the backlog?
- How precise are your stories, and what is the right precision? Is it testable?
- Do I understand this story well enough to do it myself?

Sprint Backlog - example


To Do
In Progress
Done
Nominate a video for an achievement
Modify database schema to add achievement table
Create front end web pages for editing and displaying achievements
Add service calls to API for adding, updating, removing, finding achievements
Generate advertising revenue reports
Add report UI to advertiser-facing admin pages
Create report templates
Find old schoolmates' videos
Build schoolmate
Modify search result API to add schoolmate search
Download to my mobile device
License mobile file formats for all supported mobile devices
Add mobile download UI to main video viewing page
Create library to convert videos to mobile formats

Still on velocity

How can you define your velocity for a new team at the first sprint?

Alternatives

- Take data from similar teams, working in similar contests
- Make a “random” guess
- Do not define velocity and postpone the decision

Still on velocity

How can you define your velocity for a new team at the first sprint?

Alternatives

- Take data from similar teams, working in similar contests
- Make a “random” guess
- Do not define velocity and postpone the decision

Running sprint

Let's consider a general sprint backlog, how do we plan the sprint?

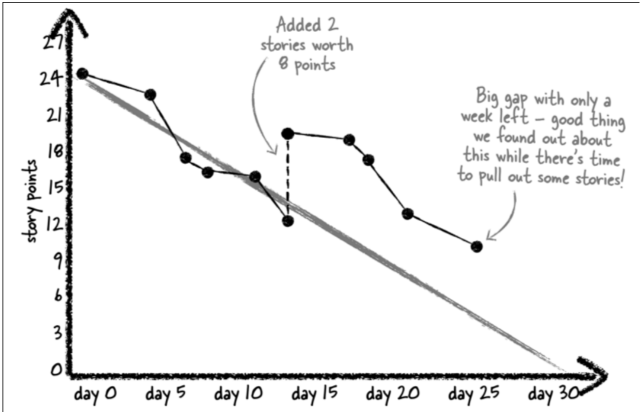
- Consider the user stories in the sprint backlog and detail them on needed activities, where each activity does not require more than one day to complete (2^{nd} planning meeting lead by the SM)
- group cards together with the user story
- proceed iteratively selecting one card per time and move them among the columns of the backlog
- if the the sprint ends and there are still cards non fully implemented, move them back to the product backlog

Sprint execution and burndown charts

When the sprint starts:

- Draw a burndown chart
- report points acquisition for “done done” stories
- in case you add stories to the sprint report it in the chart

Burndown chart sample



How do you know you are done?

“Done” dimensions

- . . . with a story
- . . . with a sprint
- . . . release to integration
- . . . release to production

How to

- Brainstorming session
- Categorization session (Development, Testing, Project Management, Other)
- Sorting and Consolidation
- Creation and publishing

Scrum basic rules – Daily scrum

1. **Attendance**: PO, SM, TMs
2. **Period**: every day
3. **Duration**: 15 minutes (timeboxed) – everyone should be on time!
4. **Characteristics**: stand-up meeting
5. **Objective**:
   - What have I done since the last daily scrum?
   - What will I do between now and the next daily scrum?
   - What obstacles and roadblocks are in my way?
6. Follow-up meeting among interested members to further elaborate on possible emerged issues

Scrum basic rules – Daily scrum

1. **Attendance**: PO, SM, TMs
2. **Period**: every day
3. **Duration**: 15 minutes (timeboxed) – everyone should be on time!
4. **Characteristics**: stand-up meeting
5. **Objective**:
   - What have I done since the last daily scrum?
   - What will I do between now and the next daily scrum?
   - What obstacles and roadblocks are in my way?
6. Follow-up meeting among interested members to further elaborate on possible emerged issues

Scrum basic rules – Daily scrum

1. **Attendance**: PO, SM, TMs
2. **Period**: every day
3. **Duration**: 15 minutes (timeboxed) – everyone should be on time!
4. **Characteristics**: stand-up meeting
5. **Objective**:
   - What have I done since the last daily scrum?
   - What will I do between now and the next daily scrum?
   - What obstacles and roadblocks are in my way?
6. Follow-up meeting among interested members to further elaborate on possible emerged issues

Scrum basic rules – Daily scrum

1. **Attendance**: PO, SM, TMs
2. **Period**: every day
3. **Duration**: 15 minutes (timeboxed) – everyone should be on time!
4. **Characteristics**: stand-up meeting
5. **Objective**:
    - What have I done since the last daily scrum?
    - What will I do between now and the next daily scrum?
    - What obstacles and roadblocks are in my way?
6. Follow-up meeting among interested members to further elaborate on possible emerged issues

Scrum basic rules – Daily scrum

1. **Attendance**: PO, SM, TMs
2. **Period**: every day
3. **Duration**: 15 minutes (timeboxed) – everyone should be on time!
4. **Characteristics**: stand-up meeting
5. **Objective**:
    - What have I done since the last daily scrum?
    - What will I do between now and the next daily scrum?
    - What obstacles and roadblocks are in my way?
6. Follow-up meeting among interested members to further elaborate on possible emerged issues

Daily Scrum

The Daily Scrum

It functions as an inspection of the work that the team is doing, so TMs can adapt that work to deliver the most value. And it gives the team the opportunity to make decisions at the last responsible moment, giving the flexibility to have the right person do the right work at the right time

Team mood

- interested - listen to colleagues statements
- collaborative - if you can help do it
- humble - people generally make mistakes

Daily Scrum

The Daily Scrum

It functions as an inspection of the work that the team is doing, so TMs can adapt that work to deliver the most value. And it gives the team the opportunity to make decisions at the last responsible moment, giving the flexibility to have the right person do the right work at the right time

Team mood

- interested - listen to colleagues statements
- collaborative - if you can help do it
- humble - people generally make mistakes

Daily Scrum – How to

How to hold an effective Daily Scrum

- Act like a “pig” – do not ignore the “to do” column
- Take detailed conversation off-line
- Take turns going first
- Don’t treat like a ritual
- Everyone participates
- Don’t treat it like a status meeting
- Inspect every task
- Change the plan if it needs to be changed

Daily Scrum – How to

How to hold an effective Daily Scrum

- Act like a “pig” – do not ignore the “to do” column
- Take detailed conversation off-line
- Take turns going first
- Don’t treat like a ritual
- Everyone participates
- Don’t treat it like a status meeting
- Inspect every task
- Change the plan if it needs to be changed

Daily Scrum – How to

How to hold an effective Daily Scrum

- Act like a “pig” – do not ignore the “to do” column
- Take detailed conversation off-line
- Take turns going first
- Don’t treat like a ritual
- Everyone participates
- Don’t treat it like a status meeting
- Inspect every task
- Change the plan if it needs to be changed

Daily Scrum – How to

How to hold an effective Daily Scrum

- Act like a “pig” – do not ignore the “to do” column
- Take detailed conversation off-line
- Take turns going first
- Don’t treat like a ritual
- Everyone participates
- Don’t treat it like a status meeting
- Inspect every task
- Change the plan if it needs to be changed

Daily Scrum – How to

How to hold an effective Daily Scrum

- Act like a “pig” – do not ignore the “to do” column
- Take detailed conversation off-line
- Take turns going first
- Don’t treat like a ritual
- Everyone participates
- Don’t treat it like a status meeting
- Inspect every task
- Change the plan if it needs to be changed

Daily Scrum – How to

How to hold an effective Daily Scrum

- Act like a “pig” – do not ignore the “to do” column
- Take detailed conversation off-line
- Take turns going first
- Don’t treat like a ritual
- Everyone participates
- Don’t treat it like a status meeting
- Inspect every task
- Change the plan if it needs to be changed

Daily Scrum – How to

How to hold an effective Daily Scrum

- Act like a “pig” – do not ignore the “to do” column
- Take detailed conversation off-line
- Take turns going first
- Don’t treat like a ritual
- Everyone participates
- Don’t treat it like a status meeting
- Inspect every task
- Change the plan if it needs to be changed

Daily Scrum – How to

How to hold an effective Daily Scrum

- Act like a “pig” – do not ignore the “to do” column
- Take detailed conversation off-line
- Take turns going first
- Don’t treat like a ritual
- Everyone participates
- Don’t treat it like a status meeting
- Inspect every task
- Change the plan if it needs to be changed

Release Planning

Does SCRUM suggest to not derive release plans?

- Sketch a preliminary release plan
  - Estimated, ordered, and prioritized product backlog
  - Team velocity (worst, best)
  - End-of-sprint dates
- Establish degree of confidence
- Revise the plan every sprint

Key to success

- Communicate up front and often
- Update the release plan after every sprint
- Try to do the highest-priority items first
- Refine estimates on bigger items
- Deliver working software

Sprint review

- run the very last day of the sprint
- Team is supposed to be able to prepare for no more than an hour
- No presentation (just for recap), and demo in an environment as close to production as possible
- Around one hour for each week composing the sprint, but probably less is better

Review preparation

Slides?

The can be useful to provide something in the hands of the stakeholdes, and to take it home. One slide for each item:

- The sprint goal
- Stories the team forecasted/committed to deliver
- Stories the team completed
- Storied the team failed to complete
- Key decisions made during the sprint, which may inldue technical, market drive, requirements, etc.
- Project metrics (e.g. code coverage)
- Demonstration of the completed work
- Priority review for the next sprint

Running the review

Suggestions:

- Make a short recap at the beginning, possibly with slides (no more than 15 minutes)
- When possible put the software in the hands of stakeholders
- Carefully observe what they do
- Permit to them to do whatever they like
- Remember that modifications to stories are good discoveries

Retrospectives

- Run retrospective as the very last activity of each sprint
- A reptrospective meeting should last less than two hours
- the whole team should attend (PO,SM,TMs)

Run the meeting

- first 15 minutes collect data
- prioritize reported issues
- discuss each issue looking for solutions

Retrospectives

- Run retrospective as the very last activity of each sprint
- A reptrospective meeting should last less than two hours
- the whole team should attend (PO,SM,TMs)

Run the meeting

- first 15 minutes collect data
- prioritize reported issues
- discuss each issue looking for solutions

FAQs

- How does a scrum team deal with dependencies between tasks?
- This all sound good in theory, but can it really work on a real project?
- The "last responsible moment" seems to be a bit risky. Isn't it a better idea to plan up front, even if that plan has to change?
- Don't programmers suck at planning – especially for projects, which are inherently unpredictable?
- Isn't unrealistic to promise that you will have working software to demonstrate at the end of each sprint? What if the team is working on something that cannot really be demonstrated?

FAQs

- How does a scrum team deal with dependencies between tasks?
- This all sound good in theory, but can it really work on a real project?
- The "last responsible moment" seems to be a bit risky. Isn't it a better idea to plan up front, even if that plan has to change?
- Don't programmers suck at planning – especially for projects, which are inherently unpredictable?
- Isn't unrealistic to promise that you will have working software to demonstrate at the end of each sprint? What if the team is working on something that cannot really be demonstrated?

FAQs

- How does a scrum team deal with dependencies between tasks?
- This all sound good in theory, but can it really work on a real project?
- The “last responsible moment” seems to be a bit risky. Isn’t it a better idea to plan up front, even if that plan has to change?
- Don’t programmers suck at planning – especially for projects, which are inherently unpredictable?
- Isn’t unrealistic to promise that you will have working software to demonstrate at the end of each sprint? What if the team is working on something that cannot really be demonstrated?

FAQs

- How does a scrum team deal with dependencies between tasks?
- This all sound good in theory, but can it really work on a real project?
- The "last responsible moment" seems to be a bit risky. Isn't it a better idea to plan up front, even if that plan has to change?
- Don't programmers suck at planning – especially for projects, which are inherently unpredictable?
- Isn't unrealistic to promise that you will have working software to demonstrate at the end of each sprint? What if the team is working on something that cannot really be demonstrated?

FAQs

- How does a scrum team deal with dependencies between tasks?
- This all sound good in theory, but can it really work on a real project?
- The “last responsible moment” seems to be a bit risky. Isn’t it a better idea to plan up front, even if that plan has to change?
- Don’t programmers suck at planning – especially for projects, which are inherently unpredictable?
- Isn’t unrealistic to promise that you will have working software to demonstrate at the end of each sprint? What if the team is working on something that cannot really be demonstrated?

FAQs

- When we have a bug in our production software, my team has to stop what they are doing and fix it, and I cannot wait until the end of the sprint to do it. Isn’t Scrum being unrealistic about support tasks?
- It does not seem realistic to have a Product Owner who has all that authority to make decisions, all of those connections with customers and the company, and also so much free time to spend with the team every day. Does not that mean that Scrum cannot possibly work?
- I can see how sprint planning works once the team comes up with estimates, but I am still not sure where those estimates come from. How do teams estimate tasks?
- How do you handle global teams?
- OK, I get that the Daily Scrum keeps people working on the right tasks. But even well-meaning developers can get caught up doing things that are not really the best use of their time. Cannot Scrum teams still get sidetracked?

FAQs

- When we have a bug in our production software, my team has to stop what they are doing and fix it, and I cannot wait until the end of the sprint to do it. Isn't Scrum being unrealistic about support tasks?
- It does not seem realistic to have a Product Owner who has all that authority to make decisions, all of those connections with customers and the company, and also so much free time to spend with the team every day. Does not that mean that Scrum cannot possibly work?
- I can see how sprint planning works once the team comes up with estimates, but I am still not sure where those estimates come from. How do teams estimate tasks?
- How do you handle global teams?
- OK, I get that the Daily Scrum keeps people working on the right tasks. But even well-meaning developers can get caught up doing things that are not really the best use of their time. Cannot Scrum teams still get sidetracked?

FAQs

- When we have a bug in our production software, my team has to stop what they are doing and fix it, and I cannot wait until the end of the sprint to do it. Isn't Scrum being unrealistic about support tasks?
- It does not seem realistic to have a Product Owner who has all that authority to make decisions, all of those connections with customers and the company, and also so much free time to spend with the team every day. Does not that mean that Scrum cannot possibly work?
- I can see how sprint planning works once the team comes up with estimates, but I am still not sure where those estimates come from. How do teams estimate tasks?
- How do you handle global teams?
- OK, I get that the Daily Scrum keeps people working on the right tasks. But even well-meaning developers can get caught up doing things that are not really the best use of their time. Cannot Scrum teams still get sidetracked?

FAQs

- When we have a bug in our production software, my team has to stop what they are doing and fix it, and I cannot wait until the end of the sprint to do it. Isn't Scrum being unrealistic about support tasks?
- It does not seem realistic to have a Product Owner who has all that authority to make decisions, all of those connections with customers and the company, and also so much free time to spend with the team every day. Does not that mean that Scrum cannot possibly work?
- I can see how sprint planning works once the team comes up with estimates, but I am still not sure where those estimates come from. How do teams estimate tasks?
- How do you handle global teams?
- OK, I get that the Daily Scrum keeps people working on the right tasks. But even well-meaning developers can get caught up doing things that are not really the best use of their time. Cannot Scrum teams still get sidetracked?

FAQs

- When we have a bug in our production software, my team has to stop what they are doing and fix it, and I cannot wait until the end of the sprint to do it. Isn't Scrum being unrealistic about support tasks?
- It does not seem realistic to have a Product Owner who has all that authority to make decisions, all of those connections with customers and the company, and also so much free time to spend with the team every day. Does not that mean that Scrum cannot possibly work?
- I can see how sprint planning works once the team comes up with estimates, but I am still not sure where those estimates come from. How do teams estimate tasks?
- How do you handle global teams?
- OK, I get that the Daily Scrum keeps people working on the right tasks. But even well-meaning developers can get caught up doing things that are not really the best use of their time. Cannot Scrum teams still get sidetracked?

Bibliography

Andrew Stellman and Jennifer Greene
Learning Agile Understanding Scrum, XP, Lean, and Kanban
O'Reilly 2015.

- Chapters 4 and 5