Introduction to Agile Software Development and SCRUM

Andrea Polini

Software Project Management
MSc in Computer Science
University of Camerino

Outline

1. Setting the context
2. Agile Methodologies: values and principles
3. SCRUM and some of its practices

Some software peculiarities?

A Software is an artefact that has some distinctive characteristics with respect to other human developed artefacts:

- Its production cost is dominated by people cost
- Making a copy is almost free
- Almost no storage cost
- In practice no transportation costs.
- Its a logical product (intangible) making the definition of its characteristics more difficult
- It is rather easy (or at least it is perceived to be so) to modify a software artefact. Anyway it is certainly more malleable than other engineered artefacts.

Software Processes

A software process establish how to proceed with the development of a software product:

- Activity perspective
- Artefact (data) perspective
- Workflow perspective
- Role Perspective

How software has been developed and is often still developed? (simplified view)

Workflow perspective of adopted processes are not so much distinguishable from processes adopted in other engineering disciplines:

1. Fully understand what the software should do
2. Produce a detailed design for the software
3. Produce a codebase covering all the required functionality
4. Check if everything works

What kind of process is that?

How software has been developed and is often still developed? (simplified view)

Workflow perspective of adopted processes are not so much distinguishable from processes adopted in other engineering disciplines:

1. Fully understand what the software should do
2. Produce a detailed design for the software
3. Produce a codebase covering all the required functionality
4. Check if everything works

What kind of process is that?

How software has been developed and is often still developed? (simplified view)

Workflow perspective of adopted processes are not so much distinguishable from processes adopted in other engineering disciplines:

1. Fully understand what the software should do
2. Produce a detailed design for the software
3. Produce a codebase covering all the required functionality
4. Check if everything works

What kind of process is that?

How software has been developed and is often still developed? (simplified view)

Workflow perspective of adopted processes are not so much distinguishable from processes adopted in other engineering disciplines:

1. Fully understand what the software should do
2. Produce a detailed design for the software
3. Produce a codebase covering all the required functionality
4. Check if everything works

What kind of process is that?

How software has been developed and is often still developed? (simplified view)

Workflow perspective of adopted processes are not so much distinguishable from processes adopted in other engineering disciplines:

1. Fully understand what the software should do
2. Produce a detailed design for the software
3. Produce a codebase covering all the required functionality
4. Check if everything works

What kind of process is that?

What is agile?

Observations

- No single recipe that results in perfect software every time. Agile teams have ideas and ground rules that help to guide the team to make the right choices and avoid problems, or deal with them when they emerge.
- A good developer almost always has opinions about the whole direction of the project
- Changes are unavoidable
- Software is an highly added value artefact, quality is strongly dependent from people

What is agile?

Set of methods and methodologies

- more effective work
- more efficient work
- make better decisions

Promises

- On time
- high quality
- highly maintainable
- user happy
- working normal hours

What is agile?

Different mindset, based on ideas, values, and principles

- focus on teams over individuals
  - sharing knowledge
  - taking responsibilities
  - taking decisions
  - feeling commitment

Note

Manager and Developers generally have different perspectives on the same project. This affect the introduction of novel approaches to development. e.g. introduction of a daily standup meeting

What is agile?

Different mindset, based on ideas, values, and principles

- focus on teams over individuals
  - sharing knowledge
  - taking responsibilities
  - taking decisions
  - feeling commitment

Note

Manager and Developers generally have different perspectives on the same project. This affect the introduction of novel approaches to development. e.g. introduction of a daily standup meeting

Agile values

A group of highly skilled and innovative people started agile as a revolution against the “waterfall mindset”.

Agile values

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

Can waterfall work?

- Good communication
- Good practices
- It's more important the creation of the plan than sticking to it

The biggest issue . . . Requirements up-front!

Can waterfall work?

- Good communication
- Good practices
- It's more important the creation of the plan than sticking to it

The biggest issue . . . Requirements up-front!

Better-than-not-doing-it

For a “waterfall” team one of the most complex issue in PM is the transition toward an Agile mindset

Going agile is not equal to becoming an agile team

- Not just tools, techniques, and practices (can just lead to a better-than-not-doing-it effect)

Agile tools, practices, techniques

- test driven development, automated build script, build server, SCRUM, iterations, task boards, velocity, burndown charts, user stories, product owner, release plan

In a fractured perspective everyone has a different view of the agile practice

Often team’s members just improve their individual capabilities in activities for which they were already good at

Better-than-not-doing-it

For a “waterfall” team one of the most complex issue in PM is the transition toward an Agile mindset

Going agile is not equal to becoming an agile team

- Not just tools, techniques, and practices (can just lead to a better-than-not-doing-it effect)

Agile tools, practices, techniques

- test driven development, automated build script, build server, SCRUM, iterations, task boards, velocity, burndown charts, user stories, product owner, release plan

  In a fractured perspective everyone has a different view of the agile practice

Often team’s members just improve their individual capabilities in activities for which they were already good at

Better-than-not-doing-it

For a “waterfall” team one of the most complex issue in PM is the transition toward an Agile mindset

Going agile is not equal to becoming an agile team

- Not just tools, techniques, and practices (can just lead to a better-than-not-doing-it effect)

Agile tools, practices, techniques

- test driven development, automated build script, build server, SCRUM, iterations, task boards, velocity, burndown charts, user stories, product owner, release plan

  In a fractured perspective everyone has a different view of the agile practice

Often team’s members just improve their individual capabilities in activities for which they were already good at

Individuals and interactions **over** processes and tools

A great tool can sometimes help you do the wrong thing faster

It is important to understand people in the team:

- how they work together
- how each person's work impact everyone else

the Value and the practices

Used practices in line with the value:

- Daily stand-up meeting
- retrospectives
- user stories
- ...

Working software **over** comprehensive documentation

Often complex software documents have no readers. Agile methodologies aim at providing working software that adds value to the organization

Obviously this does not mean that no documentation should be provided. Instead documentation should save more time and effort than it costs

the value and the practices

- comments
- javadocs
- test-driven development
- demoing at the end of each iteration
- ...

Customer collaboration **over** contract negotiation

The objective is to provide valuable software to the customer, so software that he/she really needs.

Issues

Up-front requirements reduce customer involvement and the possibility to revise the plan after the contract is signed

Agile methodologies foster inclusion of the customer in the development team and strict cooperation.

Give customers what they really need, and not just what they ask for

If I had asked people what they wanted, they would have said faster horses

Henry Ford

Responding to change **over** following a plan

A plan provides a comfortable path toward the development of a possible wrong software. Agile methodologies ask for taking into consideration any change that could emerge.

Task board

The use of a task board is a practice helping the team to take the right decision when a change emerge. Three sections each one contaning user stories (in general) in on of the possible three different states (To do, In progress, Done).

- electronic format vs. paper based

Principles over practices

> *Without concrete practices, principles are sterile; but without principles, practices have no life, no character, no heart. Great products arise from great teams–teams who are principled, who have character, who have heart, who have persistence, and who have courage.* *Jim Highsmith*

A team becomes agile if it shares the values and not just if they adopt the practices

Interesting questions

- When the agile manifesto talks about not having comprehensive documentation, does that mean we don't have to write anything down?
- I've definitively heard that agile means doing any planning, and instead jumping straight into programming. Isn't that more efficient?
- Can I have the developers on the team go agile, but leave the rest of the team alone?
- If I'm not using Scrum, XP, Lean, Kanban, does that mean my team isn't agile?

Interesting questions

- When the agile manifesto talks about not having comprehensive documentation, does that mean we don't have to write anything down?
- I've definitively heard that agile means doing any planning, and instead jumping straight into programming. Isn't that more efficient?
- Can I have the developers on the team go agile, but leave the rest of the team alone?
- If I'm not using Scrum, XP, Lean, Kanban, does that mean my team isn't agile?

Interesting questions

- When the agile manifesto talks about not having comprehensive documentation, does that mean we don't have to write anything down?
- I've definitively heard that agile means doing any planning, and instead jumping straight into programming. Isn't that more efficient?
- Can I have the developers on the team go agile, but leave the rest of the team alone?
- If I'm not using Scrum, XP, Lean, Kanban, does that mean my team isn't agile?

Interesting questions

- When the agile manifesto talks about not having comprehensive documentation, does that mean we don't have to write anything down?
- I've definitively heard that agile means doing any planning, and instead jumping straight into programming. Isn't that more efficient?
- Can I have the developers on the team go agile, but leave the rest of the team alone?
- If I'm not using Scrum, XP, Lean, Kanban, does that mean my team isn't agile?

Agile principles

Principle motivations

The Agile manifest signers identified ground rules and ideas that help the team to make the right choices and avoid problems. 12 principles were then defined.

Principles can be organized according to four sections:

- delivery
- communication
- execution
- improvement

Principles list – Delivering the project

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage
   - Nobody get's in trouble when there's a change
   - We are all in this together, including the customer
   - We don't sit on change until it's too late
   - Changes are not solutions to previous mistakes
   - Learn from the changes
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
   - Practice of timeboxed iterations

Principles list – Delivering the project

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage
   - Nobody get's in trouble when there's a change
   - We are all in this together, including the customer
   - We don't sit on change until it's too late
   - Changes are not solutions to previous mistakes
   - Learn from the changes
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
   - Practice of timeboxed iterations

Principles list – Delivering the project

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage
   - Nobody get's in trouble when there's a change
   - We are all in this together, including the customer
   - We don't sit on change until it's too late
   - Changes are not solutions to previous mistakes
   - Learn from the changes
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
   - Practice of timeboxed iterations

Principles list – Communicating and working together

Objective is somehow to suggest the definition of as much documentation as you need to run the project. And this depends from communications habits. Comprehensive documentation leads to unnecessary changes and wasted effort. Waterfall real practices on changes often do not reflect theory

4. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
5. Business people and developers must work together daily throughout the project.
6. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
   - rewards on bug metrics for individuals is not a good idea
   - reward BAs on the amount of documentation is not a good idea
   - CYA attitude

Principles list – Communicating and working together

Objective is somehow to suggest the definition of as much documentation as you need to run the project. And this depends from communications habits. Comprehensive documentation leads to unnecessary changes and wasted effort. Waterfall real practices on changes often do not reflect theory

4. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
5. Business people and developers must work together daily throughout the project.
6. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
   - rewards on bug metrics for individuals is not a good idea
   - reward BAs on the amount of documentation is not a good idea
   - CYA attitude

Principles list – Communicating and working together

Objective is somehow to suggest the definition of as much documentation as you need to run the project. And this depends from communications habits. Comprehensive documentation leads to unnecessary changes and wasted effort. Waterfall real practices on changes often do not reflect theory

4. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
5. Business people and developers must work together daily throughout the project.
6. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
   - rewards on bug metrics for individuals is not a good idea
   - reward BAs on the amount of documentation is not a good idea
   - CYA attitude

Principles list – Project execution

7 Working software is the primary measure of progress.

8 Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9 Continuous attention to technical excellence and good design enhances agility.

Principles list – Project execution

7 Working software is the primary measure of progress.

8 Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9 Continuous attention to technical excellence and good design enhances agility.

Principles list – Project execution

7 Working software is the primary measure of progress.

8 Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9 Continuous attention to technical excellence and good design enhances agility.

Principles list – Costantly improving the project and the team

10 Simplicity – the art of maximizing the amount of work not done – is essential.

11 The best architectures, requirements, and designs emerge from self-organizing teams.

- the work generally starts from user stories
- incremental design, instead of big design architecture covering all requirements

12 At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Principles list – Costantly improving the project and the team

10 Simplicity – the art of maximizing the amount of work not done – is essential.

11 The best architectures, requirements, and designs emerge from self-organizing teams.

- the work generally starts from user stories
- incremental design, instead of big design architecture covering all requirements

12 At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Principles list – Costantly improving the project and the team

10 Simplicity – the art of maximizing the amount of work not done – is essential.

11 The best architectures, requirements, and designs emerge from self-organizing teams.

- the work generally starts from user stories
- incremental design, instead of big design architecture covering all requirements

12 At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

FAQs

- I'm a project manager, and I'm still not clear on how I fit into an agile team: What's my role in all of this?
- If the whole team plans together, then does that mean nobody is in charge? How are decision made?

Bibliography

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

- Chapters 1, 2 and 3

SCRUM

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 most relevant 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

SCRUM most relevant 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

SCRUM most relevant 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

SCRUM most relevant 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

SCRUM most relevant 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

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

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

Product Owner responsibilities

A good product owner...

- 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

Sprints and Planning

Scrum is based on:

- Iterative and Incremental approach to software development
- Effective project planning: planning is performed in clearer conditions and for shorter periods
- Frequent delivery of running software (value to customer)
- User needs defined in User stories

How can I provide a project overall cost to the management?

Misconception: In SCRUM (Agile) assessments only refer to a single sprint

The process is simple:

- Create User Stories
- Prioritize stories
- Estimate stories
- Determine velocity
- Determine team cost
- Calculate project cost
- Commit?

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 3×5 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 3×5 index card

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

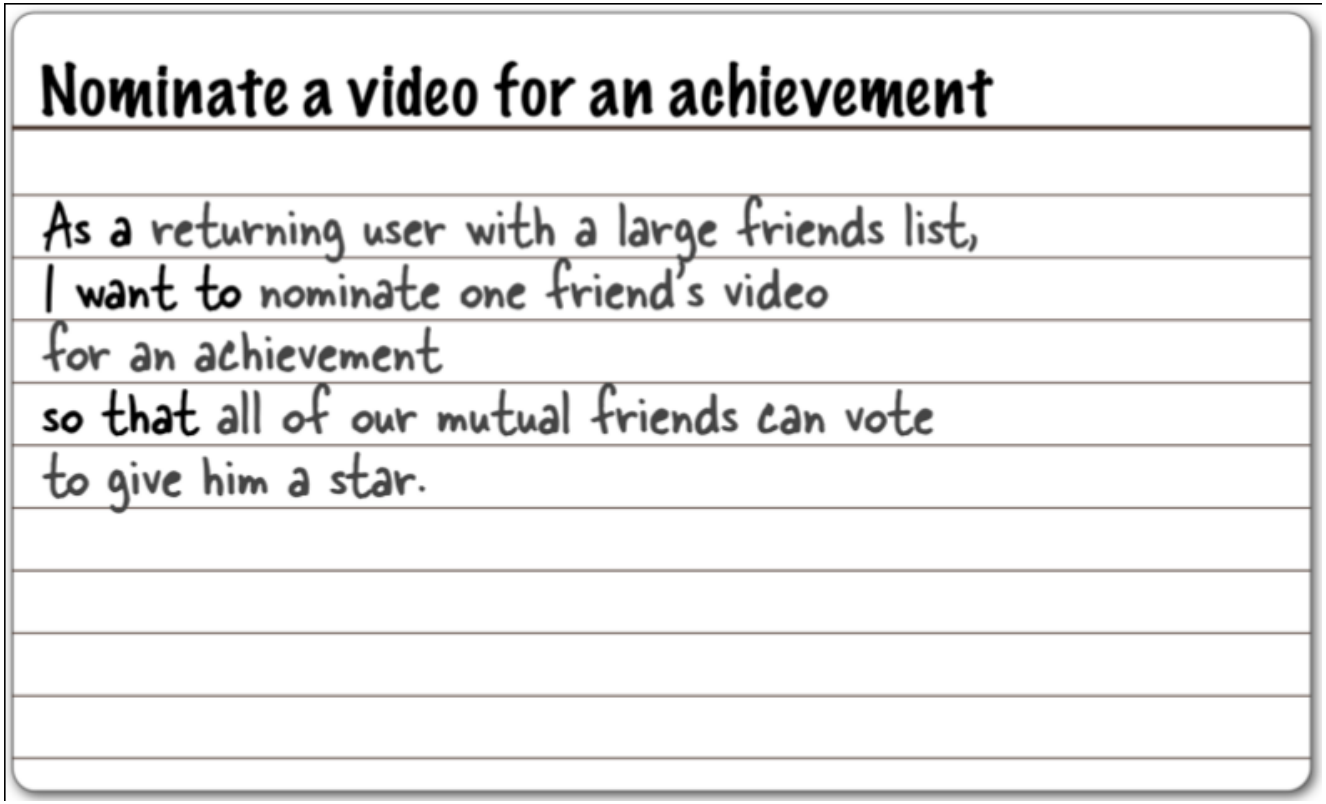
Can generally follow a template:

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

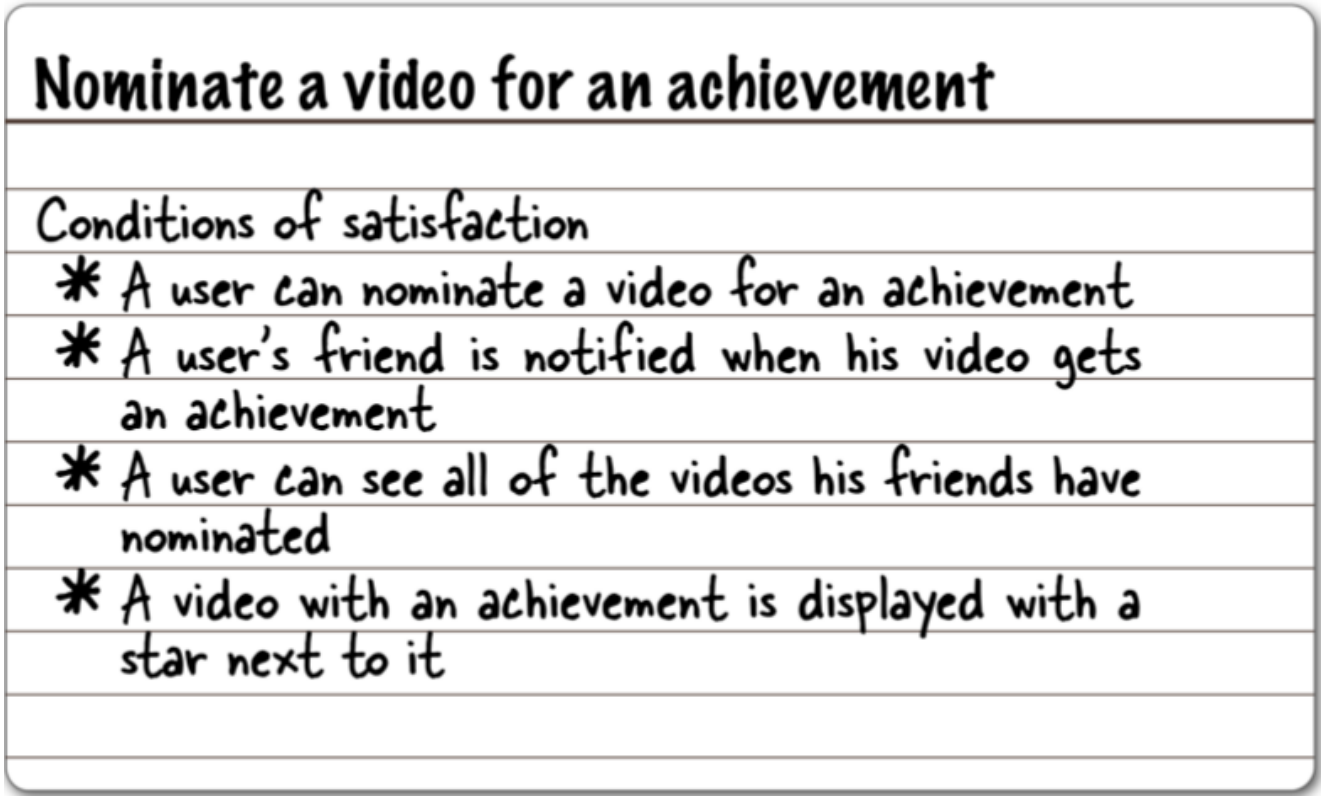
User Stories

User stories are often written on index cards or sticky notes, stored in a shoe box, and arranged on walls or tables to facilitate planning and discussion. As such, they strongly shift the focus from writing about features to discussing them. In fact, these discussions are more important than whatever text is written.

User stories and satisfaction conditions



User stories and satisfaction conditions



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:

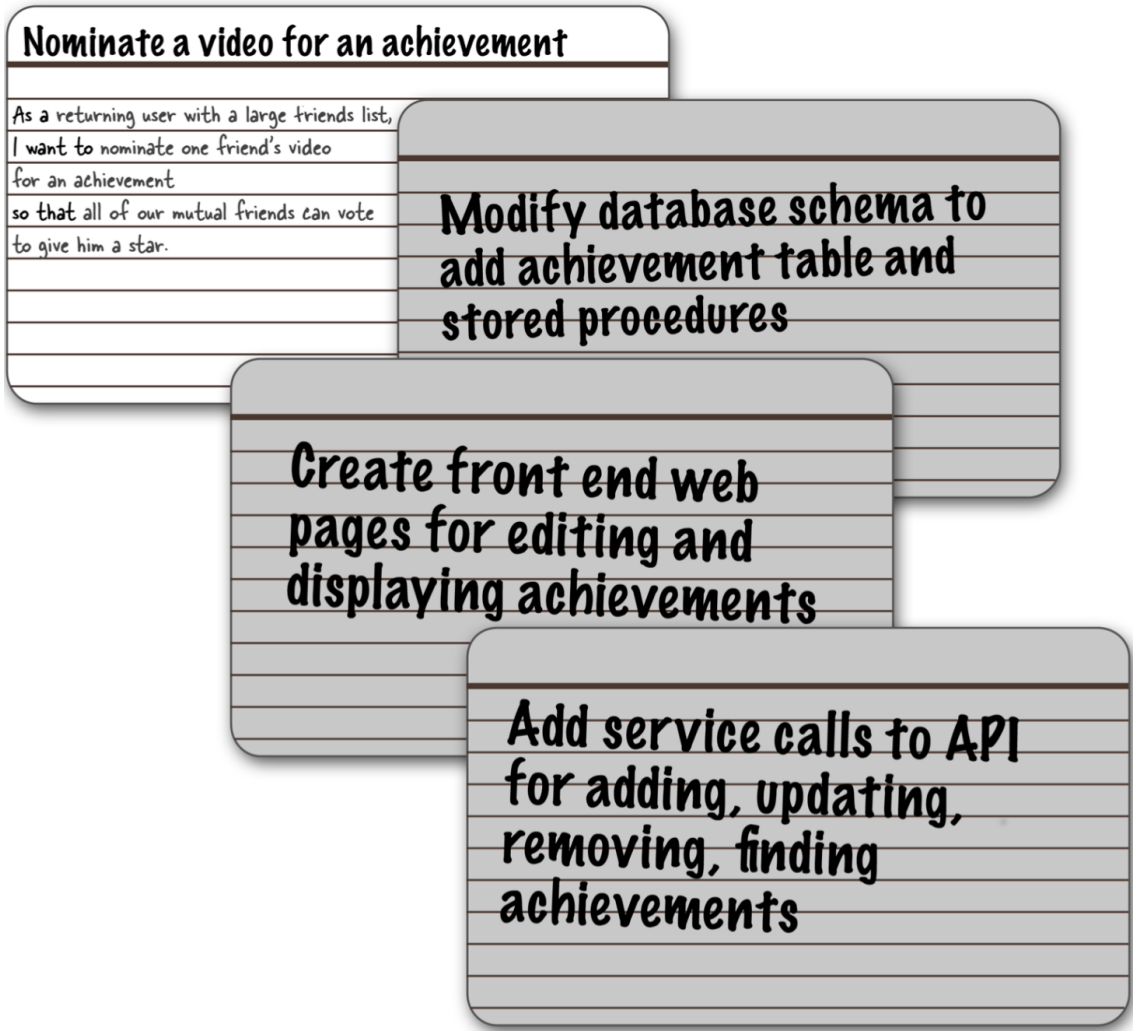
- write code
- create design and architecture
- build tests
- install operating systems
- design and build databases
- deploy software to production servers
- run usability tests
- all of those other things that teams actually do every day to build and release software.

Second meeting – How to?

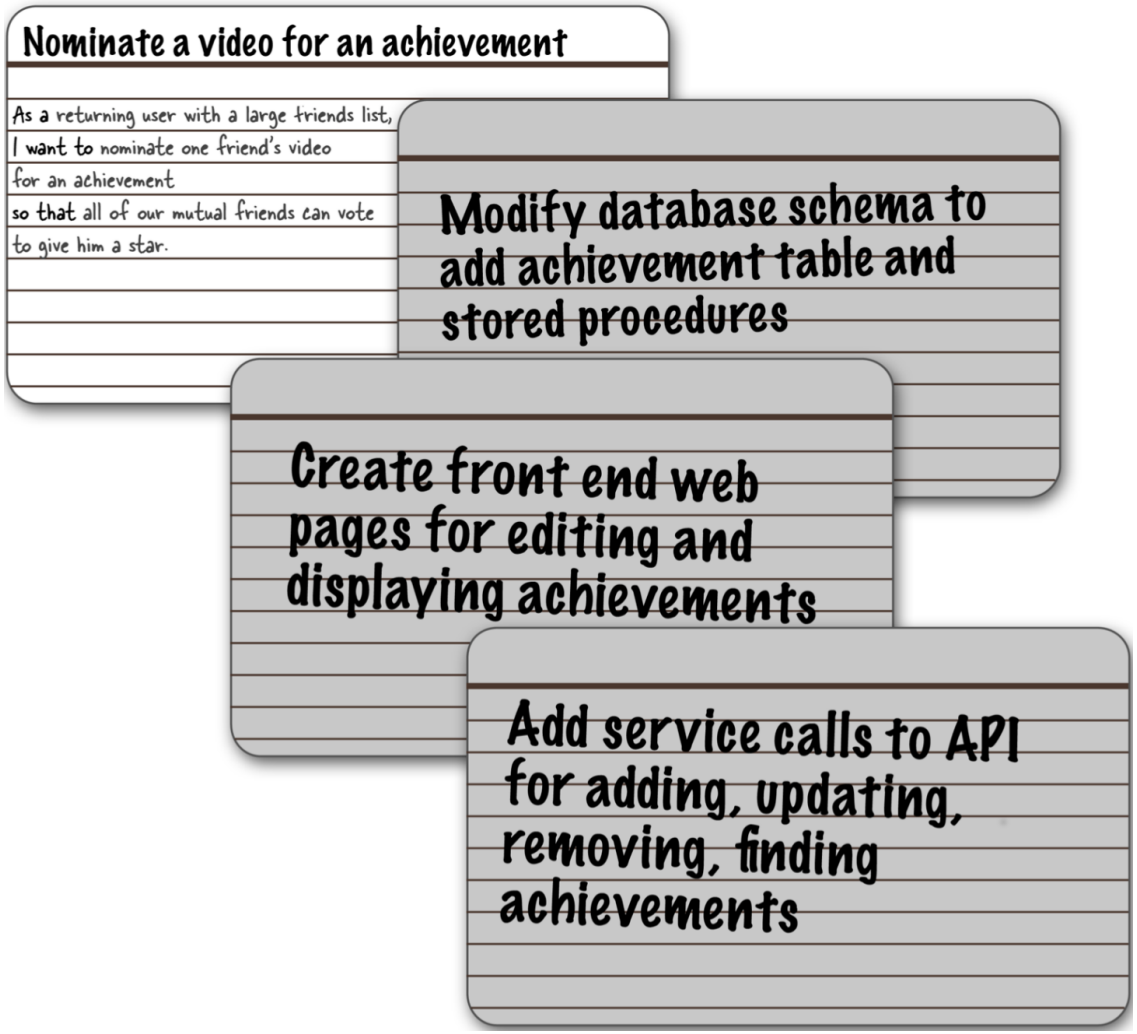
The meeting can adopt the following principles:

- 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 not discussed US get only one task card
- all stories and tasks are added to the “to do” column of the sprint backlog

User stories and activities



Story decomposition



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

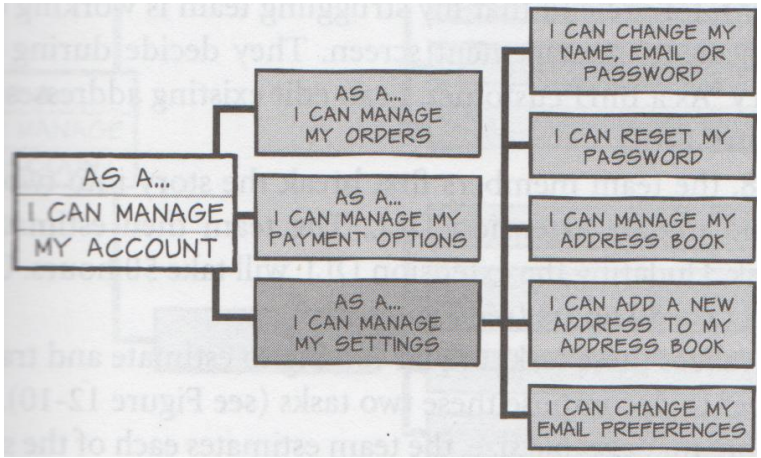
Three flavors stories – epics, themes, stories

- 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?

Role modeling steps

Before you discover stories you should discover roles.
The following steps should be performed to identify and select a useful set of user roles:

- brainstorm an initial set of user roles: use “standard” brainstorming strategies to identify roles
- organize the initial set: cluster identified roles and identify overlaps
- consolidate roles: discuss and decide on the relevant roles
- refine the roles

User and Roles - refining step

Identify relevant attributes for roles:

- The frequency with which the user will use the software.
- The user's level of expertise with the domain.
- The user's general level of proficiency with computers and software.
- The user's level of proficiency with the software being developed.
- The user's general goal for using the software. Some users are after convenience, others favor a rich experience, and so on.

After roles have been clearly defined it is possible to create personas

User and Roles - refining step

Identify relevant attributes for roles:

- The frequency with which the user will use the software.
- The user's level of expertise with the domain.
- The user's general level of proficiency with computers and software.
- The user's level of proficiency with the software being developed.
- The user's general goal for using the software. Some users are after convenience, others favor a rich experience, and so on.

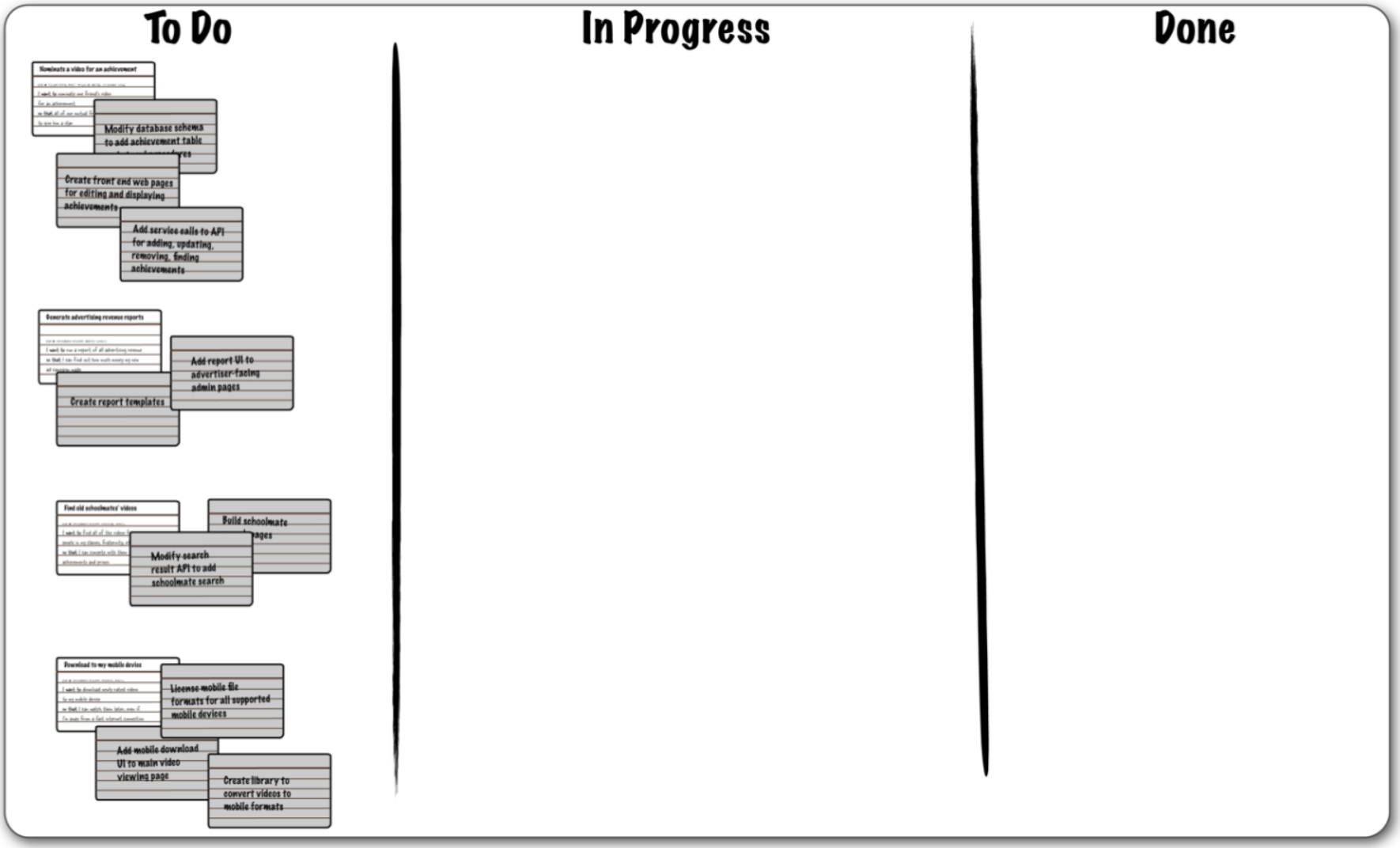
After roles have been clearly defined it is possible to create personas

Writing stories

To create good stories We INVEST time in creating stories in line with the following principles:

- Independent
- Negotiable
- Valuable to users or customers
- Estimatable
- Small
- Testable

Sprint Backlog - example



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?

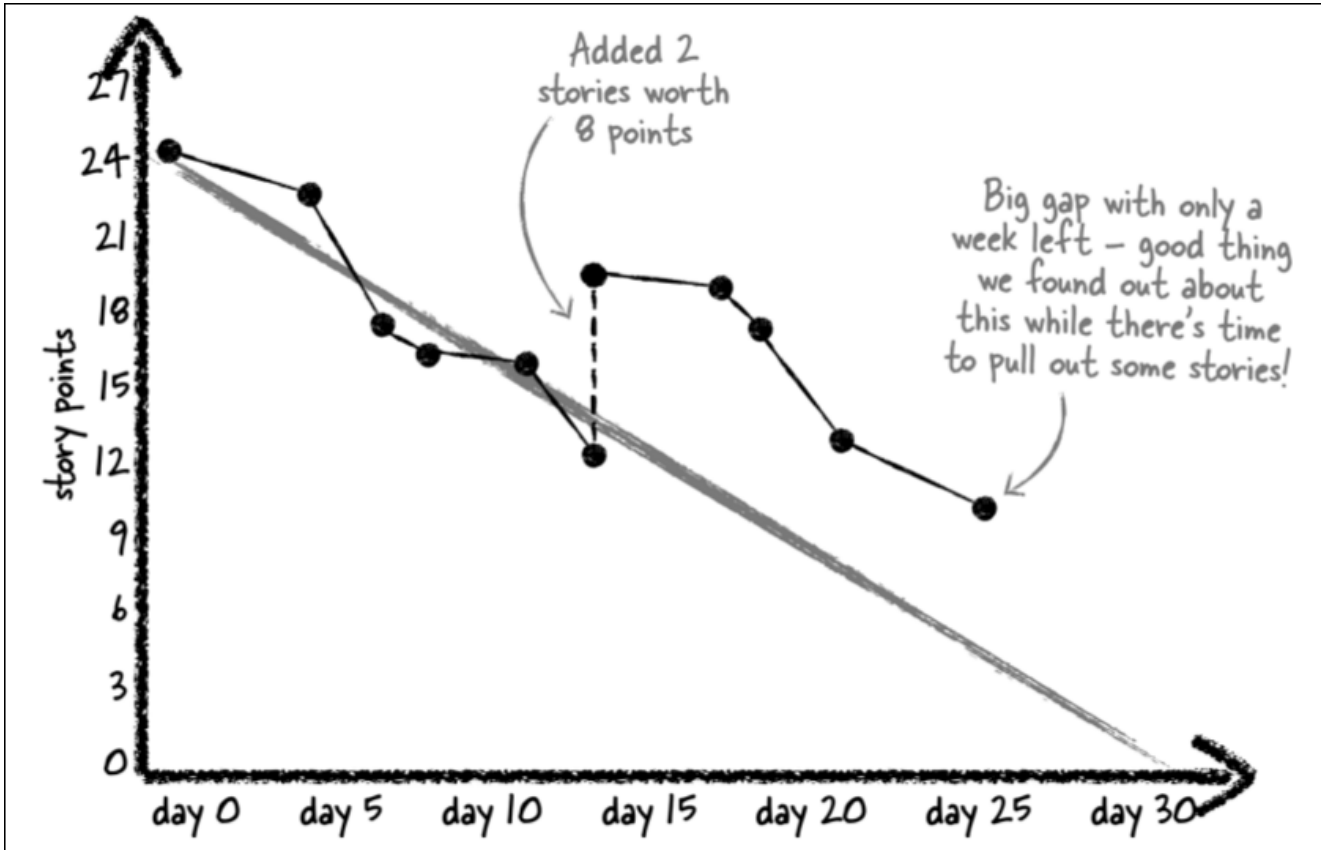
- Consider the user stories in the sprint backlog and detail them on needed activities, where each activity does not require more than one day to complete (2^{nd} planning meeting lead by the SM)
- group cards together with the user story
- proceed iteratively selecting one card per time and move them 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

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

Scrum 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.

Scrum 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.

Scrum 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

Scrum 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

Scrum values: 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.

Scrum values: 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.

Scrum values: 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

Scrum values: 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

Scrum values: 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

Scrum values: 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

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	***	*****	*	***	*****	**	***	**

FAQs

- How does a scrum team deal with dependencies between tasks?
- 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?
- 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?
- 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?
- 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?