



## Lesson Objectives



- Introduction to SCRUM
- Scrum Framework
  - Scrum Roles
    - Product Owner
    - Scrum Master
    - Team
  - Ceremonies
    - Sprint planning
    - Sprint review
    - Sprint retrospective
    - Daily scrum meeting
  - Artifacts
    - Product backlog
    - Sprint backlog
    - Burndown charts



### Lesson Objectives

- Definition of "Ready"
- Definition of "Done"
- Introduction to Extreme Programming
- Introduction to Lean Software Development
- Principles of Lean Software Development
- What is Kanban?



## 2.1: Agile Methods and Practices - SCRUM

### Introduction to SCRUM



- Agile way of project management
- A team based collaborative approach
- Iterative & incremental development
- Always focus to deliver "Business Value"

#### Wikipedia definition:

Scrum is an iterative and incremental agile software development framework for managing software projects and product or application development.

#### [www.scrumalliance.org](http://www.scrumalliance.org):

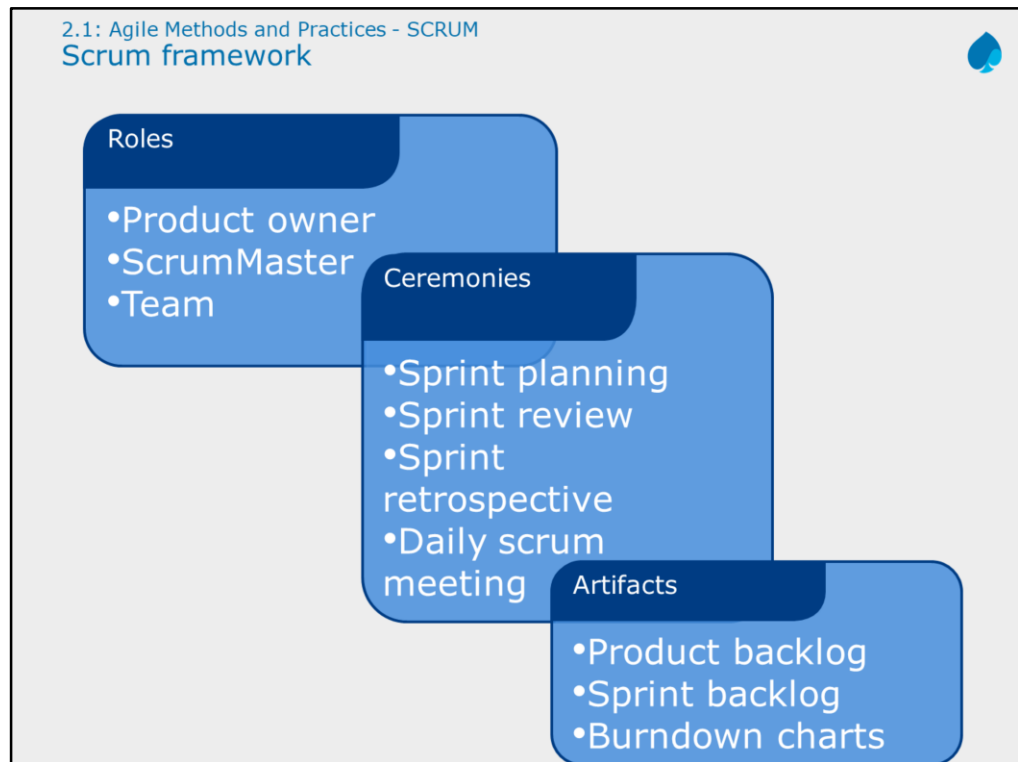
Scrum is an agile framework for completing complex projects. Scrum originally was formalized for software development projects, but works well for any complex, innovative scope of work. The possibilities are endless. The Scrum framework is deceptively simple.

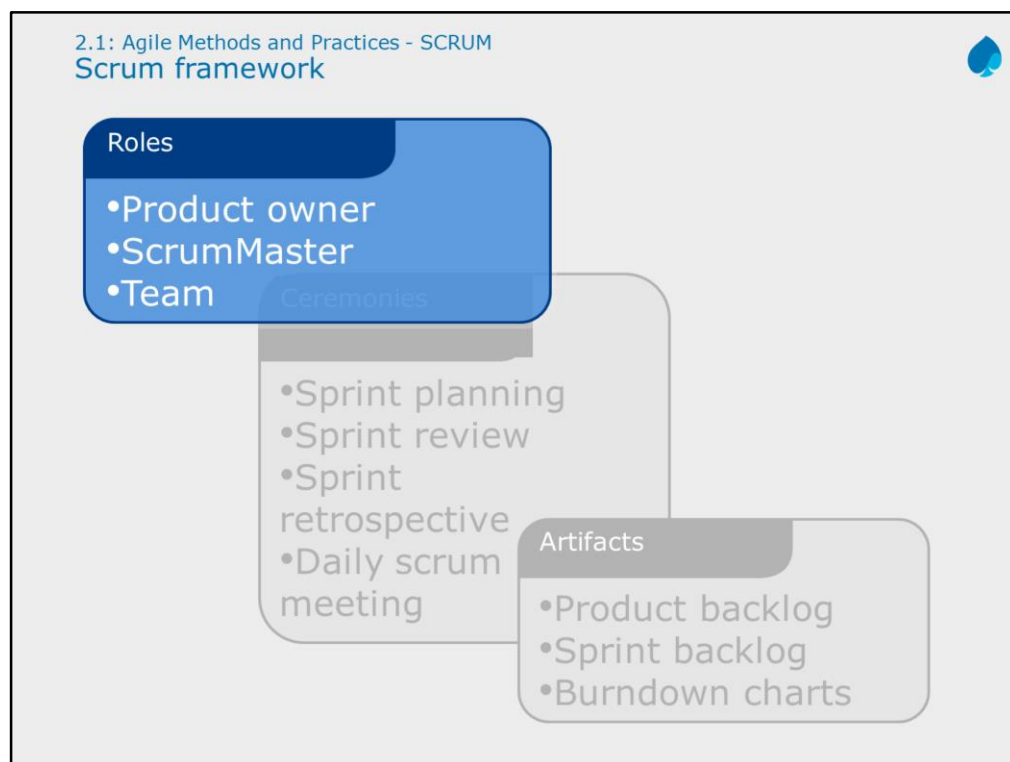
### Introduction to SCRUM

Scrum is one of the more popular agile methods in use today and originally developed at Easel in 1993 by Jeff Sutherland and Ken Schwaber. Since that time it has been used at many software companies, which have resulted in the method being extended and enhanced.

When Jeff Sutherland created the scrum process in 1993, he borrowed the term "scrum" from an analogy put forth in a 1986 study by Takeuchi and Nonaka, published in the Harvard Business Review. In that study, Takeuchi and Nonaka compare high-performing, cross-functional teams to the scrum formation used by Rugby teams.

Scrum is the leading agile development methodology, used by Fortune 500 companies around the world. The Scrum Alliance exists to transform the way we tackle complex projects, bringing the Scrum framework and agile principles beyond software development to the broader world of work.





## 2.1.1.1: SCRUM Framework

## Scrum Roles



## – Product Owner

- Possibly a Product Manager or Project Sponsor
- Decides features, release date, prioritization, \$\$\$



## – Scrum Master

- Typically a Project & Process Co-ordinator or Team Leader
- Responsible for enacting Scrum values and practices
- Remove impediments / politics, keeps everyone productive



## – Project Team

- 5-10 members; Teams are self-organizing
- Cross-functional: QA, Programmers, UI Designers, etc.
- Membership should change only between sprints



#### 2.1.1.1: SCRUM Roles

##### Product owner



- Define the features of the product
- Decide on release date and content
- Be responsible for the profitability of the product (ROI)
- Prioritize features according to market value
- Adjust features and priority every iteration, as needed
- Accept or reject work results
- Responsible for:
  - Product Vision
  - Stakeholder management
  - Scope Management
  - Cost Management
  - Monitoring Release progress



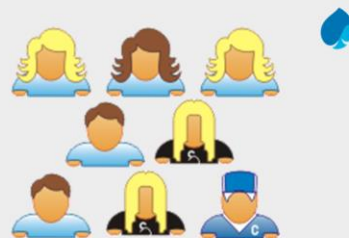
#### 2.1.1.1: SCRUM Roles The ScrumMaster



- Responsible for facilitation of all ceremonies
- Responsible for enacting Scrum values and practices
- Removes impediments
- Ensure that the team is fully functional and productive
- Enable close cooperation across all roles and functions
- Shield the team from external interferences

#### 2.1.1.1: SCRUM Roles The team

- Typically 5-9 people
- Cross-functional:
  - Programmers, testers, user experience designers, etc.
- **M**embers should be full-time
  - May be exceptions (e.g., database administrator)
- Teams are self-organizing
  - Ideally, no titles but rarely a possibility
- Membership should change only between sprints



## 2.1.1 SCRUM framework

## Agreement - Definition of "Ready" (DoR)



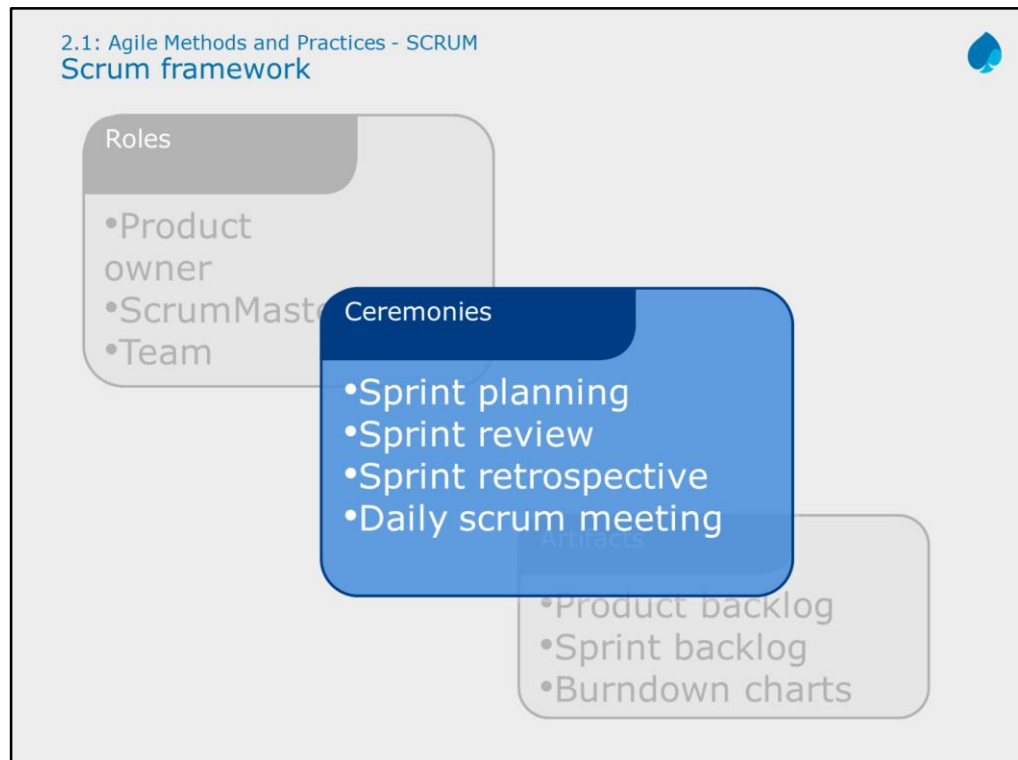
- For a Story to be "Ready", following criteria have to be met
  - The story should reasonably show INVEST characteristics
    - I** – Independent / Immediately actionable
    - N** – Negotiable
    - V** – Valuable to the customer, user or product
    - E** – Estimable
    - S** – Sized to fit
    - T** – Testable
  - The business implications of the story have been discussed, any impacts to finance, customer care have been addressed
  - The User Interaction Design is ready (At the very least wireframes covering all interactions of the story should be available)
  - Any design assets needed for the story have been prepared to a reasonable degree (PSDs for some if not all pages in the Story should be available)

## 2.1.1 SCRUM framework



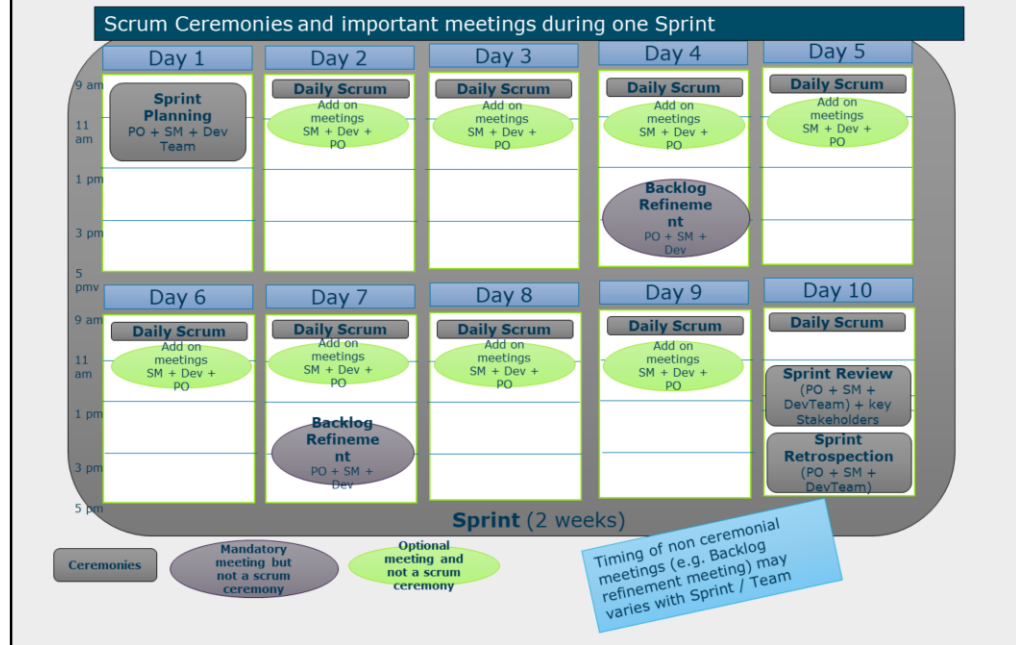
## Agreement - Definition of "Done" (DoD)

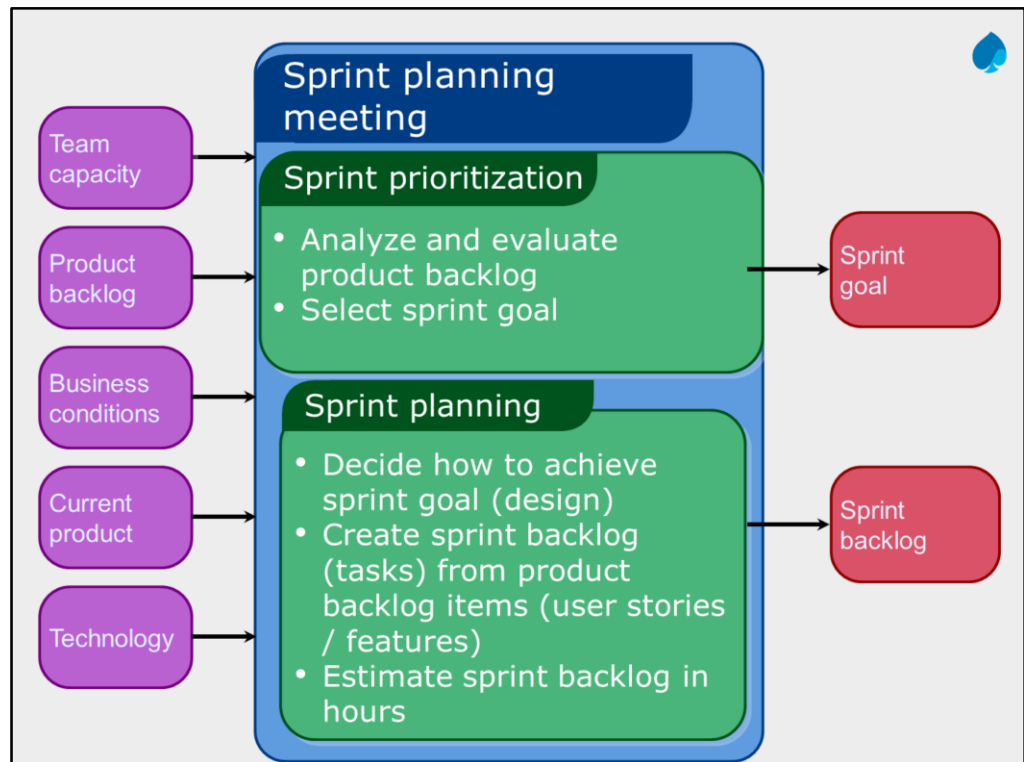
- Definition of Done must describe exactly what "done" means
  - Product Owner must pay careful attention when defining the DoD
  - The scrum team must challenge the DoD, if necessary
  - *"What's not in DoD, is not needed"*
  - Item is either "done" or "not done"
- Example:
  - Story: Picture upload
    - end user can upload his/her picture from profile settings page
    - picture is shown on the left upper corner of the profile page
    - picture is scaled to fit the profile picture box on the profile page
    - functional tests are passed
    - regression tests are passed
    - design documents are updated
    - user's guide is updated
- **Does not** define any details of the implementation!



## 2.1.1 SCRUM framework

## Scrum with 2 weeks Sprint Duration





## 2.1.1 SCRUM framework

**Sprint planning**

- Team selects items from the product backlog they can commit to completing
- Sprint backlog is created
  - Tasks are identified and each is estimated (1-16 hours)
  - Collaboratively, not done alone by the ScrumMaster
- High-level design is considered

As a vacation planner, I want to see photos of the hotels.

Code the middle tier (8 hours)  
Code the user interface (4)  
Write test fixtures (4)  
Code the foo class (6)  
Update performance tests (4)



### 2.1.1 SCRUM framework

#### The daily scrum

- Parameters
  - Daily
  - 15-minutes
  - Stand-up
- Not for problem solving
  - Whole world is invited
  - Only team members, ScrumMaster, product owner, can talk
- Helps avoid other unnecessary meetings



2.1.1.1 SCRUM framework  
The daily scrum



## Everyone answers 3 questions

1  
What did you do yesterday?

2  
What will you do today?

3  
Is anything in your way?

- These are *not* status for the ScrumMaster
  - They are commitments in front of peers

### 2.1.1.1 SCRUM framework The sprint review



- Team presents what it accomplished during the sprint
- Typically takes the form of a demo of new features or underlying architecture
- Informal
  - 2-hour prep time rule
  - No slides
- Whole team participates
- Invite the world



### 2.1.1 SCRUM framework

#### Sprint retrospective



- Periodically take a look at what is and is not working
- Typically 15–30 minutes
- Done after every sprint
- Whole team participates
  - ScrumMaster
  - Product owner
  - Team
  - Possibly customers and others

### 2.1.1 SCRUM framework Sprint retrospective



## Start / Stop / Continue

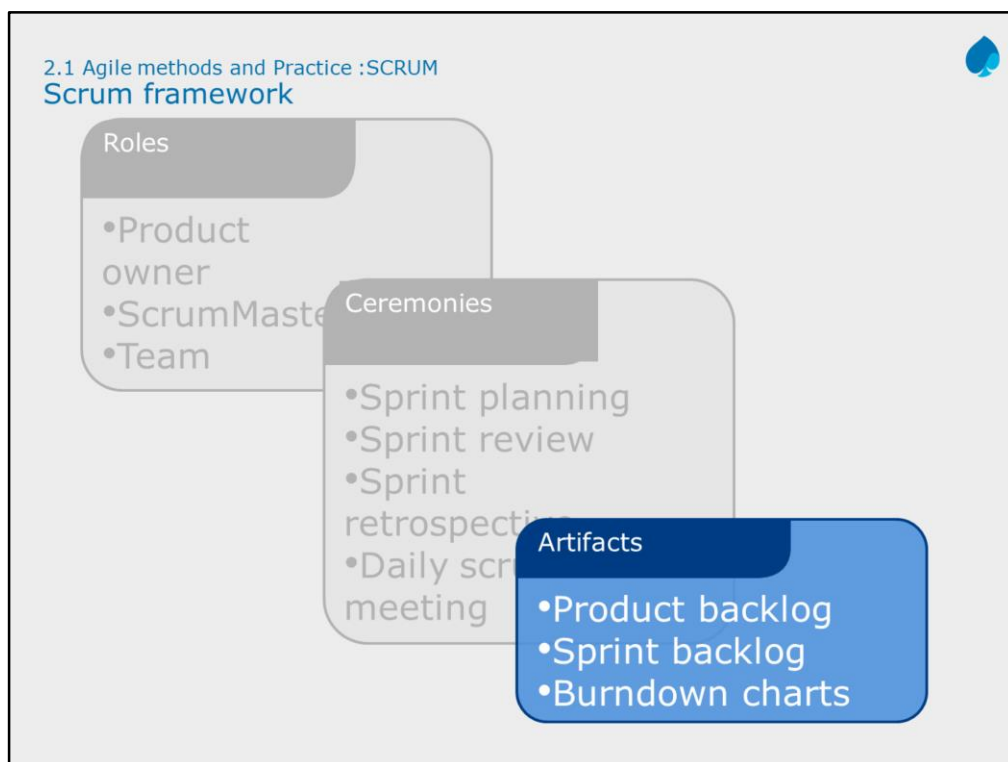
Whole team gathers and discusses what they'd like to:

Start doing

Stop doing


This is just one  
of many ways to  
do a sprint  
retrospective.

Continue doing



2.1.1.1 SCRUM Framework

## Product backlog



• The requirements

• A list of all desired work on the project

Ideally expressed such that each item has value to the users or customers of the product

• Prioritized by the product owner

• Reprioritized at the start of each sprint

This is the product backlog

COPYRIGHT © 2005, MOUNTAIN GOAT SOFTWARE

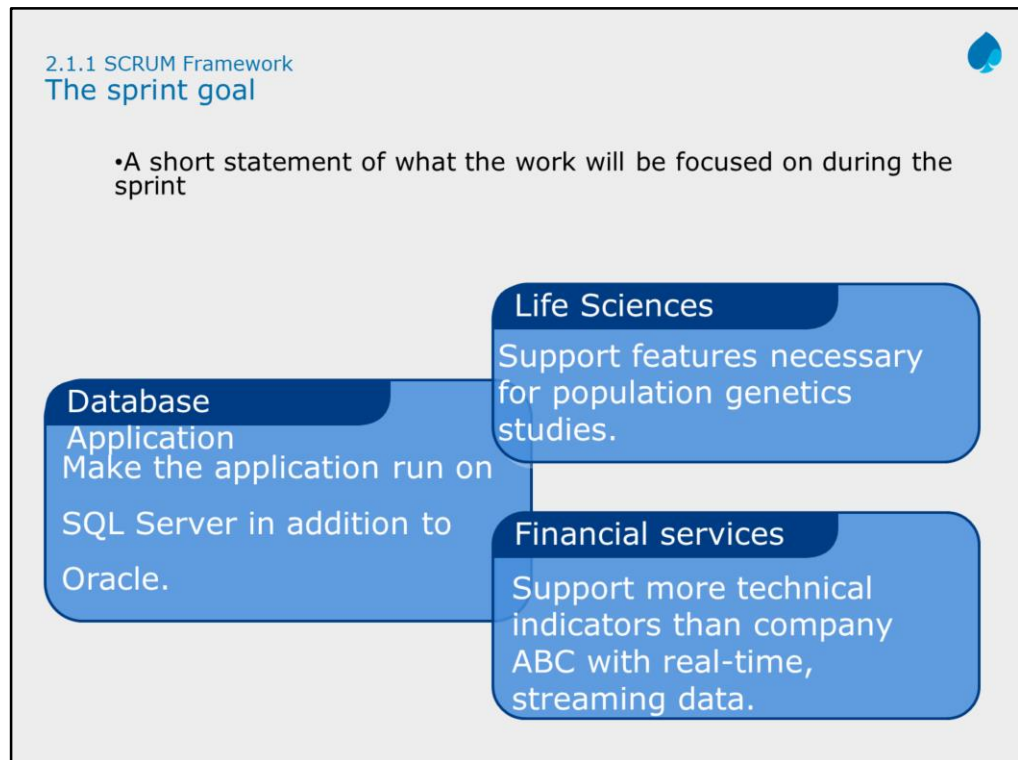
## 2.1.1 SCRUM Framework

## Sample Product Backlog



Backlog item	Estimate
Allow a guest to make a reservation	3 (story points)
As a guest, I want to cancel a reservation.	5
As a guest, I want to change the dates of a reservation.	3
As a hotel employee, I can run Revenue reports	8
Improve exception handling	8
...	30
...	50





### 2.1.1 SCRUM Framework Sprint Backlog



- Individuals sign up for work of their own choosing
  - Work is never assigned
- Estimated work remaining is updated daily
- Any team member can add, delete change sprint backlog
- Work for the sprint emerges
- If work is unclear, define a sprint backlog item with a larger amount of time and break it down later
- Update work remaining as more becomes known

### 2.1.1 SCRUM Framework

#### Sample Sprint Backlog



#### Sprint 1

01/11/2015

		Sprint Day						
		1	2	3	4	5	6	7
		Mo	Tu	We	Th	Fr	Sa	Su
19 days work in this sprint								
Hours remaining		152	152	152	152	152	152	152
Backlog Item	Backlog Item	Owner	Estimate					
1 Minor	Remove user kludge in .dpr file	BC	8	8	8	8	8	8
2 Minor	Remove cMap/cMenu/cMenuSize from disciplines.pas	BC	8	8	8	8	8	8
3 Minor	Create "Legacy" discipline node with old civils and E&I content	BC	8	8	8	8	8	8
4 Major	Augment each tbl operation to support network operation	BC	80	80	80	80	80	80
5 Major	Extend Engineering Design estimate items to include summaries	BC	16	16	16	16	16	16
6 Super	Supervision/Guidance	CAM	32	32	32	32	32	32

#### Sprint 1

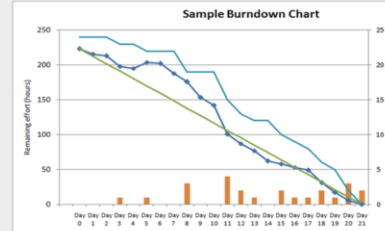
01/11/2015

		Sprint Day						
		1	2	3	4	5	6	7
		Mo	Tu	We	Th	Fr	Sa	Su
19 days work in this sprint								
Hours remaining		152	150	140	130	118	118	118
Backlog Item	Backlog Item	Owner	Estimate					
1 Minor	Remove user kludge in .dpr file	BC	8	8	4	2	0	
2 Minor	Remove cMap/cMenu/cMenuSize from disciplines.pas	BC	8	8	4	0		
3 Minor	Create "Legacy" discipline node with old civils and E&I content	BC	8	8	8	6	0	
4 Major	Augment each tbl operation to support network operation	BC	80	80	80	80	78	78
5 Major	Extend Engineering Design estimate items to include summaries	BC	16	16	16	16	16	16
6 Super	Supervision/Guidance	CAM	32	32	30	28	26	24

### 2.1.1 SCRUM Framework Sprint Burndown

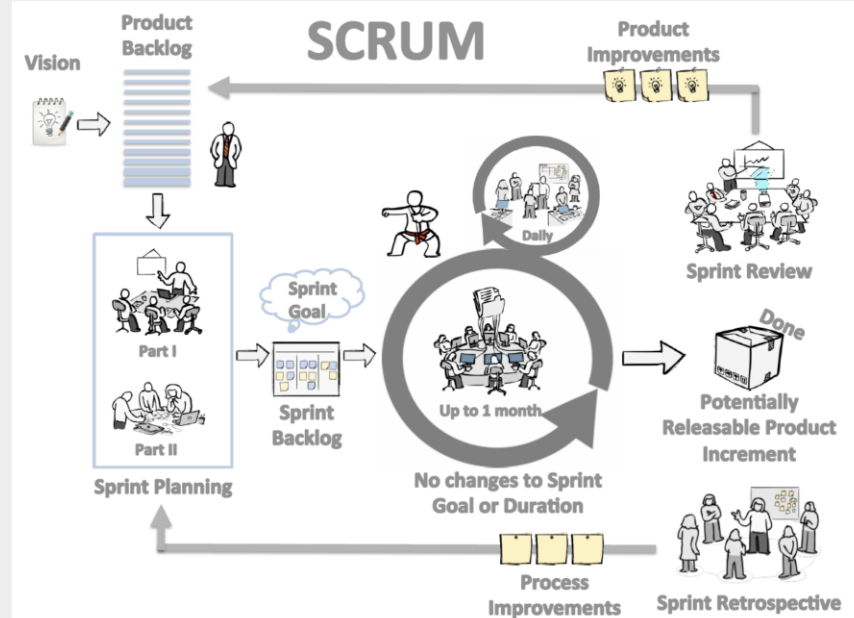


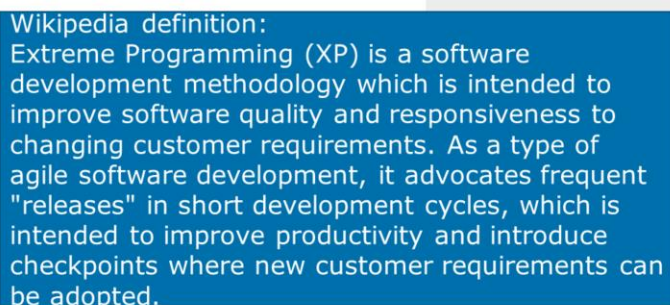
- A display of what work has been completed and what is left to complete
  - one for each developer or work item
  - updated every day
  - (make best guess about hours/points completed each day)
- Gives indication to:
  - No work being performed
  - Not fast enough work
  - Too fast work
- *variation*: Release burndown chart
  - shows overall progress
  - updated at end of each sprint



## 2.2: Agile Methods and Practices - SCRUM

## Scrum at a Glance





## 2.3: Agile Methods and Practices – Extreme Programming (XP)



## The Rules of Extreme Programming

Planning	Managing	Coding	Designing	Testing
<ul style="list-style-type: none"> <li>• User stories are written</li> <li>• Release planning creates the release schedule</li> <li>• Make frequent small releases</li> <li>• The project is divided into iterations</li> <li>• Iteration planning starts each iteration</li> </ul>	<ul style="list-style-type: none"> <li>• Give the team a dedicated open work space</li> <li>• Set a sustainable pace</li> <li>• A stand up meeting starts each day</li> <li>• The Project Velocity is measured</li> <li>• Move people around</li> <li>• Fix XP when it breaks</li> </ul>	<ul style="list-style-type: none"> <li>• The customer is always available</li> <li>• Code must be written to agreed standards</li> <li>• Code the unit test first</li> <li>• All production code is pair programmed</li> <li>• Only one pair integrates code at a time</li> <li>• Set up a dedicated integration computer</li> </ul>	<ul style="list-style-type: none"> <li>• Simplicity</li> <li>• Choose a system metaphor</li> <li>• Use CRC cards for design sessions</li> <li>• Create spike solutions to reduce risk</li> <li>• No functionality is added early</li> <li>• Refactor whenever and wherever possible</li> </ul>	<ul style="list-style-type: none"> <li>• All code must have unit tests</li> <li>• All code must pass all unit tests before it can be released</li> <li>• When a bug is found tests are created.</li> <li>• Acceptance tests are run often and the score is published</li> </ul>

### 2.3: Agile Methods and Practices – Lean Software Development

#### Introduction to Lean Software Development



- Lean Software Development is the application of Lean Thinking to the software development process
- Lean Software Development is more strategically focused than other Agile methodology
- The goals are to develop software in one-third the time, with one-third the budget, and with one-third the defect rate
- "Lean Software Development" is not a management or development methodology in itself, but it offers principles that are applicable in any environment to improve software development"

#### What is Lean Software Development?

The concepts of a lean-enterprise originate from Toyota in Japan after the Second World War. Lean software development is a management philosophy that focuses on throughput. Lean software development doesn't focus on particular components of the value-stream like code-construction or QA, but on whether the components of the chain are working as efficiently as possible so as to generate as much value as possible to the customer.

Lean software development practices were created as a result of recognizing that enterprises were reacting to complex software development and resource challenges by adding:

- **More measurements**
- **More controls**
- **More checks and balances**
- **More process rigor**



### 2.3: Agile Methods and Practices – Lean Software Development

#### Principles of Lean Software Development



- **Eliminate waste:** Do only what adds value for a customer, and do it without delay
- **Amplify learning:** Use frequent iterations and regular releases to provide feedback
- **Decide as late as possible:** Make decisions at the last responsible moment
- **Deliver as fast as possible:** The measure of the maturity of an organization is the speed at which it can repeatedly and reliably respond to customer need
- **Empower the team:** Assemble an expert workforce, provide technical leadership and delegate the responsibility to the workers
- **Build integrity in:** Have the disciplines in place to assure that a system will delight customers both upon initial delivery and over the long term
- **See the whole:** Use measurements and incentives focused on achieving the overall goal

#### Principles of Lean Software Development

**Eliminate waste** - In software development, waste is anything that does not improve the quality of code, reduces the amount of time and effort it takes to produce code, or does not deliver business value to the customer. In other words, any activity that does not “pay for itself” in reduced effort elsewhere in the system.

**Amplify Learning** - For programmers to develop a system that delivers business value, they will have to learn about many things. Some are technical, such as the advantages and disadvantages to various approaches to do remote communications in .NET (i.e., remoting, COM+, web services, etc.). Others are requirements related, such as understanding what the business user really needs versus what the developer thinks the user needs.

**Decide as late as possible** - The idea here is to wait until what the authors term “the last responsible moment” to make a decision. This is the moment at which, if the team does not make a decision, the decision will be made for them (doing nothing is a choice). The benefits of this are avoiding or delaying the costs of change, which obviously cannot be incurred if you have not limited your options yet.

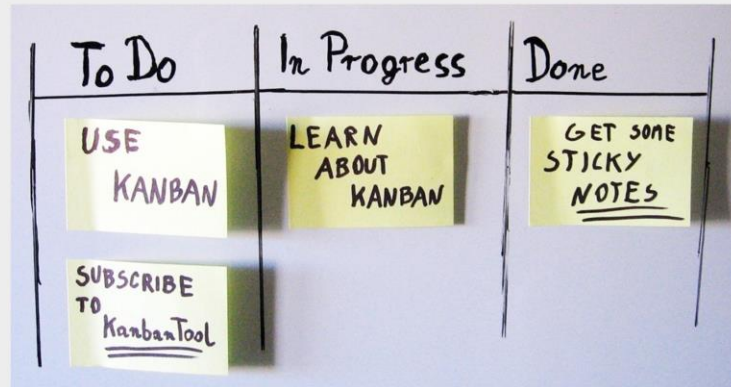
**Deliver as fast as possible** - This is the foundation of iterative development. Requirements change as a percentage of the original requirements increases non-linearly as the amount of time increases. Typical 9-12 month projects generate roughly a 25 percent change in requirements. However, the amount of requirements change over a month averages only 1-2 percent. And it is much easier to get users to accept waiting until next month rather than next year.

#### 2.4: Agile Methods and Practices – Kanban

##### What is Kanban?



- The word Kan means "visual" in Japanese and the word "ban" means "card". So Kanban refers to "visual cards"



## 2.4: Agile Methods and Practices – Kanban



## What is Kanban? (Cont.)

- Kanban is way for teams and organizations to visualize their work, identify and eliminate bottlenecks and achieve dramatic operational improvements in terms of throughput and quality
- Kanban is a method to gradually improve whatever you do – whether software development, IT/ Ops, Staffing, Recruitment, Marketing and Sales
- in fact, almost any business function can benefit from applying Kanban to bring about significant benefits such as reduced lead time, increased throughput and much higher quality of products or services delivered



### Differences

Scrum	Kanban
<b>Timeboxed iterations prescribed.</b>	<b>Timeboxed iterations optional.</b>
<b>Team commits</b> to a specific amount of work for this iteration.	<b>Commitment optional.</b>
Uses <b>Velocity</b> as default metric for planning and process improvement.	Uses <b>Lead time</b> as default metric for planning and process improvement.
<b>Cross-functional teams</b> prescribed.	Cross-functional teams optional. <b>Specialist teams allowed.</b>
<b>Items broken down</b> so they can be completed within 1 sprint.	No particular item size is prescribed.
<b>Burndown chart prescribed</b>	No particular type of diagram is prescribed
<b>WIP limited indirectly</b> (per sprint)	<b>WIP limited directly</b> (per workflow state)
<b>Estimation prescribed</b>	<b>Estimation optional</b>
<b>Cannot add items to ongoing iteration.</b>	<b>Can add new items whenever capacity is available</b>
<b>A sprint backlog is owned by one specific team</b>	<b>A kanban board may be shared by multiple teams</b> or individuals
<b>Prescribes 3 roles</b> (PO/SM/Team)	<b>Doesn't prescribe any roles</b>
<b>A Scrum board is reset</b> between each sprint	<b>A kanban board is persistent</b>
<b>Prescribes a prioritized product backlog</b>	<b>Prioritization is optional.</b>

## Summary



- In this lesson, you have learnt
  - Introduction to SCRUM
  - Different Scrum Roles and Responsibilities in Agile
  - Scrum Core Practices and Artifacts
  - Definition of "Done"
  - An introduction to Extreme Programming
  - Lean Software Development
  - Kanban



Add the notes here.