



## Learning objectives

- ◆ Why Agile?
- ◆ Scrum Framework
- ◆ Vision
- ◆ Product Owner and stakeholders
- ◆ Product Backlog and requirement techniques
- ◆ Other Scrum roles
- ◆ Scrum events
- ◆ Done Increment
- ◆ Value optimizing
- ◆ Planning, Estimating and Monitoring progress

## Professional Scrum Product Owner I

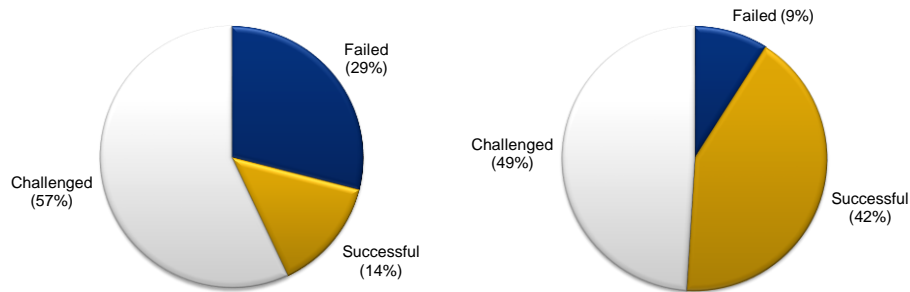
- ◆ Scrum.org
- ◆ Online assessment
- ◆ Time limit: 60 minutes
- ◆ Pass mark: 85% (68 out of 80)
- ◆ Open assessments allow you to gauge your basic knowledge of Scrum



Global Knowledge.



## Waterfall vs. Agile



Source: The CHAOS Manifesto, Copyright 2011

## Waterfall

- ◆ A plan-driven (predictive, prescriptive) process:
  - ◆ attempting to plan and anticipate up front all of the features a user might want in the end product and to determine how best to build these features
  - ◆ the work plans are based on execution of a sequential set of work-specific phases
- ◆ Progress is measured by (percentage) completion of work
- ◆ Milestones are related to the approval of documents that describe the work done, e.g.
  - “*The Functional Specifications Document has been signed off*”

## Waterfall

The waterfall (predictive) process depends on:

- ◆ Requirements not changing and being completely understood
- ◆ Technology working without any problems
- ◆ People being as predictable and reliable as machines

The predictive process is not suitable for developing and sustaining complex products, where there is more that we do not know than we do know

## Empiricism

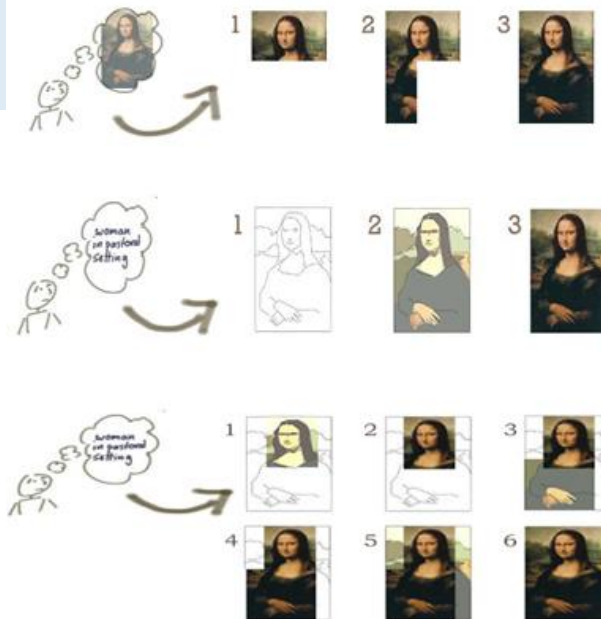
*Scrum is founded on empirical process control theory (empiricism)*

- ◆ Information is gained by observation, rather than prediction
- ◆ Decisions are made based on what is known
- ◆ Empirical processes are best for complex problems (like software development)

The three pillars of an empirical process:

1. Transparency
2. Inspection
3. Adaptation

**Scrum employs an incremental and iterative approach to optimize predictability and control risk**



Patton J.

## History of Agility

- ◆ “The New New Product Development Game,” by Hirotaka Takeuchi, Ikujiro Nonaka. Harvard Business Review, (*Rugby approach*), 1986
- ◆ Dynamic System Development Method, DSDM Consortium, 1994
- ◆ “Scrum Development Process,” Ken Schwaber and Jeff Sutherland, OOPSLA, 1995
- ◆ Extreme Programming (Beck, Cunningham, Jeffries), 1996
- ◆ Agile Manifesto, 2001

# Manifesto for Agile Software Development

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

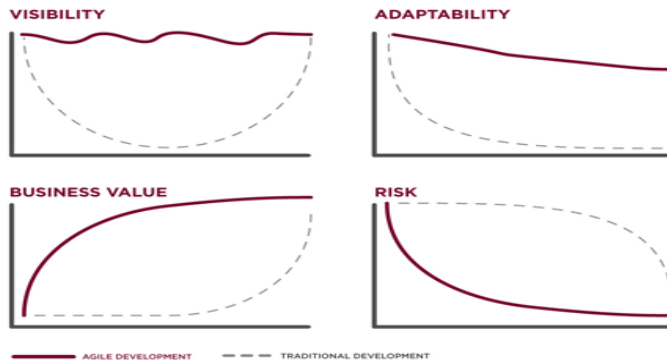
*That is, while there is value in the items on the right,  
we value the items on the left more.*

## Principles behind the Agile Manifesto

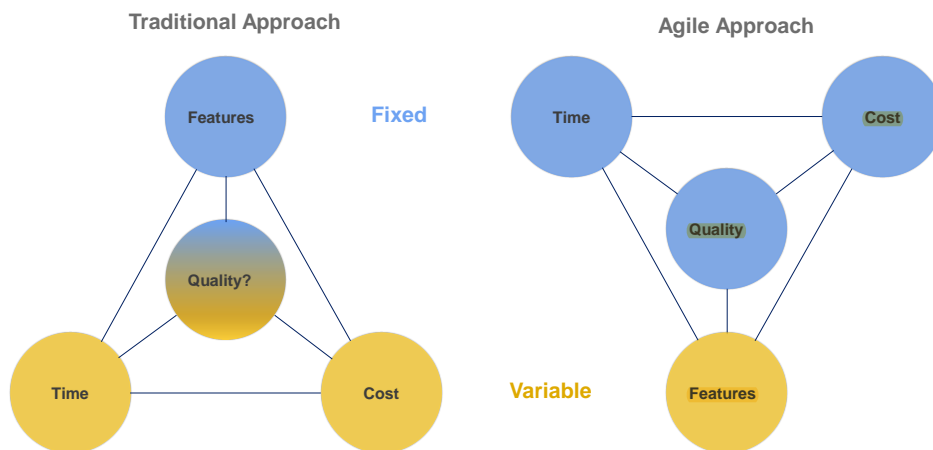
1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
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.
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.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

## Value of Agile development

### AGILE DEVELOPMENT VALUE PROPOSITION



## Traditional versus Agile



## What is Scrum?



Scrum is a framework for developing and sustaining complex products. A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value. It is not a process or a technique for building products (you can employ various processes and techniques within the framework)

*Scrum guide*

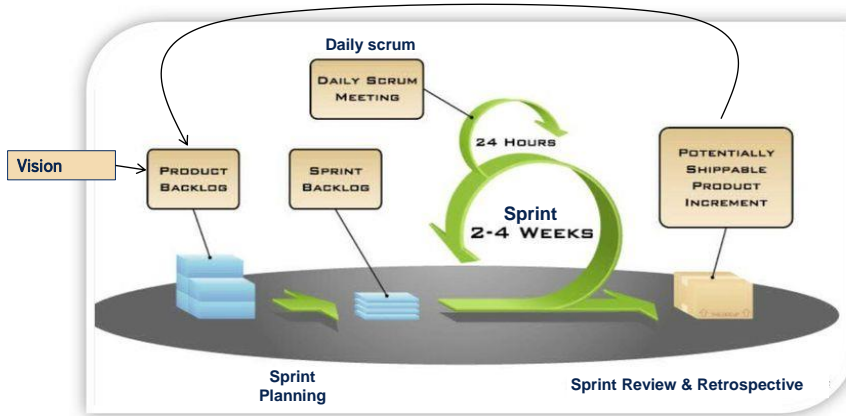


Global Knowledge.





# Scrum Framework



Copyright mountain goat software

# Scrum framework

◆ 3 Scrum Roles: Product owner, Scrum Master, Development team  
 (beside the scrum roles there are other stakeholders)

◆ 5 (prescribed) timeboxed events:

- ◆ Sprint planning
- ◆ Daily Scrum
- ◆ Sprint
- ◆ Sprint Review
- ◆ Sprint Retrospective

*Simple to understand,  
difficult to master!*

◆ 3 Scrum Artifacts:

- ◆ Product Backlog
- ◆ Sprint Backlog
- ◆ Increment

◆ Rules










## Product Vision

- ◆ A product vision represents the core essence of its product. It sets the end state for what a product will deliver in the future
- ◆ It acts as the project's true north, sets the direction and guides the Scrum team:
  - ◆ if adding an item to the Product Backlog will not help your product achieve its vision, then the value of that feature should be diminished.
- ◆ The overarching goal everyone must share – Product Owner, Scrum Master, team, management, customers and other stakeholders
- ◆ To write and share a vision: start with the product's end user in mind
- ◆ A strong product vision is supported by details of who the customers are, what they need
- ◆ Vision is not officially Scrum terminology

# Vision Board

	<b>Vision Statement</b> Crisp summary of the vision / idea.		
			
<b>Target group</b>  Which market segment does the product address?  Who are the target users and customers?	<b>Needs</b>  Which needs does the product fulfil?  How does it create value for its users?  Which emotions will it evoke?	<b>Product</b>  What are the three to five top features?  What are its unique selling points?	<b>Value</b>  How is the product going to benefit the company?  Will it, for instance, increase revenue, enter a new market, develop the brand, reduce cost, create valuable knowledge?

<http://www.romanpichler.com/>



# Vision Board

Vision Statement 	Phrase or sentence		
Target group 	Needs 	Product 	Value 
<div style="border: 1px solid gray; padding: 5px; margin-bottom: 10px; background-color: #fff9c4;">Target users</div> <div style="border: 1px solid gray; padding: 5px; background-color: #fff9c4;">Target customers</div>	<div style="border: 1px solid gray; padding: 5px; margin-bottom: 10px; background-color: #fff9c4;">Problem to solve</div> <div style="border: 1px solid gray; padding: 5px; background-color: #fff9c4;">Goal to achieve</div>	<div style="border: 1px solid gray; padding: 5px; background-color: #fff9c4;">3-5 top features</div>	<div style="border: 1px solid gray; padding: 5px; background-color: #fff9c4;">Business goals</div>



## Roles

### Scrum team

- ◆ Product owner + Scrum Master + Development team
- ◆ Self-organizing: chooses how to best accomplish its work, rather than being directed by others outside the team
- ◆ Cross- functional: has all competencies needed to accomplish the work without depending on others not part of the team

*The team model in Scrum is designed to optimize flexibility, creativity, and productivity*

### (other) Stakeholders (not an official Scrum Role)

*Definition Stakeholders:* Those who have an interest in the product being developed (and/or the Scrum process), either because he or she has funded it, will use it, or will be affected by it

## The Product Owner

- ◆ Maximizing Value (Product and work Development Team)
  - ◆ How this is done may vary (per organization, Scrum Team, individual)
- ◆ The sole person responsible for managing the Product Backlog:
  - ◆ Clearly expressing Product Backlog Items
  - ◆ Ordering the items in Product Backlog to best achieve goals and missions
  - ◆ Optimizing the value of work the Development Team performs
  - ◆ Ensuring that Product backlog is visible, transparent and clear to all, and shows what the Scrum Team will work on next
  - ◆ Ensuring the Development Team understands Product Backlog Items to the level needed
- ◆ The Product Owner may do the above work or have the Development Team do it, but he/she remains accountable



## The Product Owner (continued)

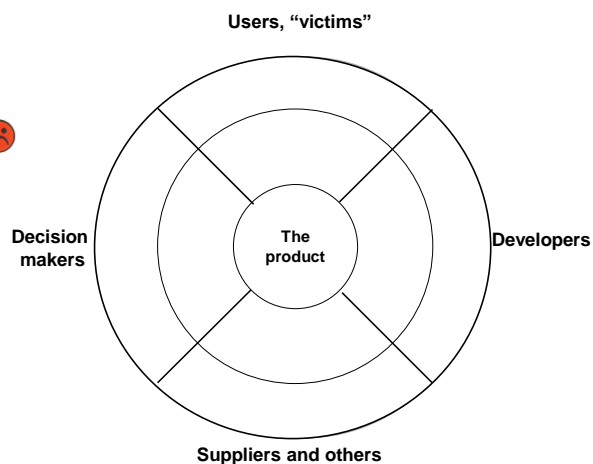
- ◆ Is one person
- ◆ Decisions are visible in content and ordering of product backlog
- ◆ Accepts or rejects work results
- ◆ The entire organisation must respect the decisions of the Product Owner
- ◆ The Development Team isn't allowed to act on what anyone else says

## The Product Owner and stakeholders

- ◆ The Product Owner represents the stakeholders
- ◆ The Product Owner actively asks for stakeholder input and expectations to process into the Product Backlog
- ◆ A Product Owner engages actively and regularly with stakeholders (*not only at the Sprint Review!*)

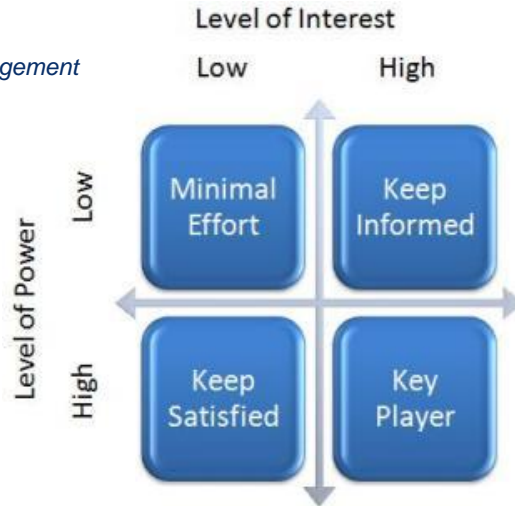
## Stakeholder Analysis

1. **Who** are involved and what is their interest?
2. Do they have a positive or negative **attitude** towards the initiative? 
3. **Power** - is their power to influence the project significant or relatively limited? *Onion diagram*
4. (Quality of) **the relationships** 



# Power Interest matrix

Determine stakeholder management approach



# Influencing styles

**Persuade**

*Convince  
Make proposals  
Arguments in front or against*

**When to use?**  
There is factual knowledge  
No tense relationships  
There are no emotions  
You are accepted as expert

**State**

*Judge  
Set goals, norms and expectations  
Excerpt pressure*

**When to use?**  
If you have (in)formal power  
Under time pressure  
You have a personal interest  
Solution doesn't need consensus

*Self exposure  
Acknowledge and involve  
Assert and show understanding*

**When to use?**  
If emotions play a huge role  
If you are not seen as an expert  
If consensus is needed

**Investigate**

*Define future possibilities  
Create an shared identity*

**When to use?**  
The other trusts you  
There is uncertainty or fear  
There are shared goals or values  
If consensus is needed

**Inspire**



## Product Backlog

- ◆ The Product Backlog is an ordered list of features, functions, (system) requirements, enhancements and fixes that might be needed in the product (to deliver the Vision)
- ◆ Is the single source of requirements for any changes to be made to the product
- ◆ The Product Owner is responsible for the PB, including its content, availability and ordering
- ◆ **A Product Backlog is never complete**
- ◆ The earliest development of it only lays out the initially known and best-understood requirements
- ◆ The Product backlog evolves as the product and the environment in which it will be used evolves
- ◆ **The product backlog is dynamic**
- ◆ As long as a product exists, its Product Backlog also exists



## Product Backlog (continued)

- ◆ Product backlog items have attributes of a description, order, estimate and value
- ◆ Higher ordered Product Backlog items are clearer and more detailed than lower ordered ones
- ◆ More precise estimates are made based on the greater clarity and increased detail
- ◆ As a product is used and gains value, and the marketplace provides feedback, the Product Backlog becomes a larger and more exhaustive list
- ◆ Changes in business requirements, market conditions, or technology may cause changes in the Product Backlog
- ◆ **When multiple Scrum Teams work together on the same product, only one Product Backlog is used. A product backlog attribute that groups items may be employed**

## Personas (Source: Romanpichler.com)

Name and Picture	Characteristics	Need
What does the persona look like? What is the persona's name?	What are the persona's relevant characteristics? For instance, demographics, job, and lifestyle information.	Why would the persona want to buy or use the product?

Who?

Why?

### "Auto Pilot" - Financial Goals



**Joe Jones**

- 34 year old
- Commercial airline pilot
- Married with newborn

- Joe is a 34 year old commercial airline pilot
- He jokes that he is a "pretty boring guy."
- Remembers his small town childhood fondly, but is grateful that he moved away when he did.
- Three months ago, their first child was born, an unplanned, but welcome surprise.
- Since the birth of the baby Joe has started to focus on the financial health of the family, he started a college fund for the baby, with small contributions each month.
- Small balances on his credit cards (he usually pays off each month) since the unexpected pregnancy and expenses that come along with preparing for a new family addition. "Who knew that diapers could cost so much?"
- Credit report is rarely considered, but requests copies at milestones, such as home purchase in 2020.

**Needs**

- Joe needs *security, comfort, stability and honesty*
- Joe expects *respect and humor*
- Joe hates being *underrated or deceived*

## User stories

- ◆ Product Backlog items in the form of user stories (*or any other requirements approach that the group finds useful*)
- ◆ Each story must describe some item of value to a user or to the product owner
- ◆ As a *<role>*  
I want to *<feature>*  
so that *<benefit>*
- ◆ Example:  
*As a frequent flyer, I want to rebook a past trip  
so that I save time booking trips I take often*
- ◆ Amount of detail depends on the priority
- ◆ An epic is a large user story (too large to fit in a Sprint)
- ◆ Try to avoid technical stories

## Examples

As a buyer I want to be able to put books into a “shopping cart”  
so that I can buy them at once when I am done shopping

As a member I can read profiles of other members  
so I can find someone to date

As a Flickr member I want to be able to assign different privacy levels to my photos  
so I can control who I share which photos with

As a frequent flyer, I want to rebook a past trip  
so that I save time booking trips I take often

As a conference attendee, I want to be able to register online,  
so I can register quickly and cut down on paperwork

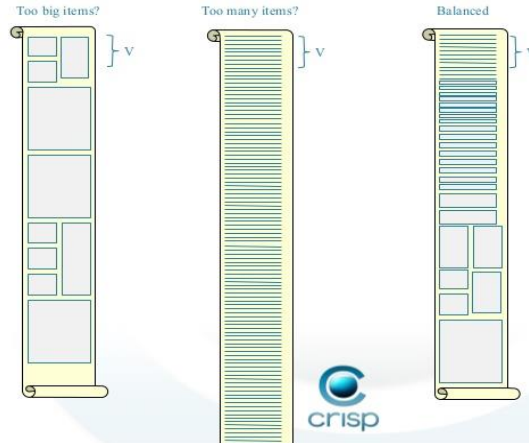
## Product backlog (example)

ID	Theme	As a/an	I want to...	so that...	Notes	Priority	Status
1	Game	moderator	create a new game by entering a name and an optional description	I can start inviting estimators	If games cannot be saved and returned to, the description is unnecessary	Required	done
2	Game	moderator	invite estimators by giving them a url where they can access the game	we can start the game	The url should be formatted so that it's easy to give it by phone.		done
5	Game	estimator	join a game by entering my name on the page I received the url for	I can participate			done
6	Game	moderator	start a round by entering an item in a single multi-line text field	we can estimate it			done
8	Game	estimator	see the item we're estimating	I know what I'm giving an estimate for			done
40	Game	participant	<del>always have the cards in the same order across multiple draws</del>	<del>it's easy to compare estimates</del>		Replaced with A08 because I didn't want the story to talk about "the same order" as that might be a UI implementation detail	todo
35	Non-functional	user	have the application respond quickly to my actions	I don't get bored			done
36	Non-functional	user	have nice error pages when something goes wrong	I can trust the system and it's developers			done
A11	Non-functional	Researcher	results to be stored in a non-identifiable way	I can study the data to see things like whether estimates converged around the first opinion given by "estimator A" for example	No names or story text should be stored but we should store each card of each hand, know who played it, and know the final accepted estimate		
A05	Game	moderator	edit an item in the list of items to be estimated	so that I can make it better reflect the team's understanding of the item			
22	Archive	moderator	export a transcript of a game as a CSV file	I can further process the stories and estimates	Exported file should be directly importable back into the system.		done

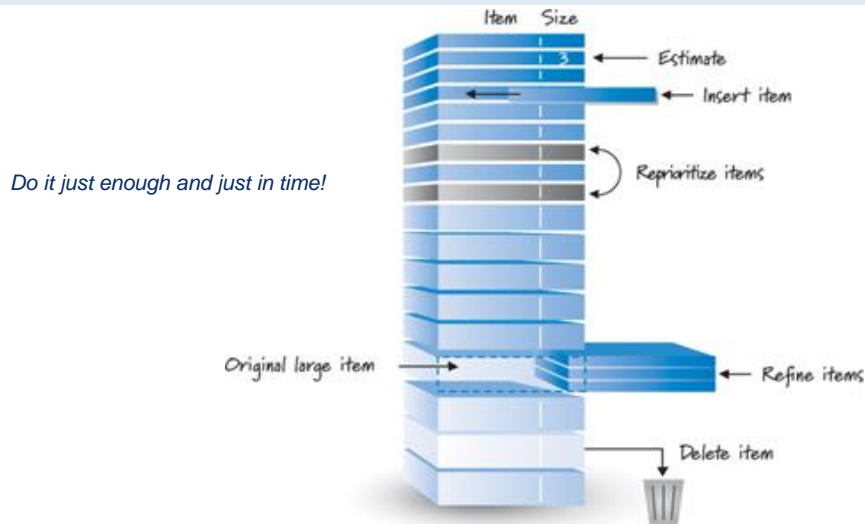
Moutangoat software

## Balanced Product Backlog

### Balance the product backlog



## Product backlog refining



## Product backlog refining

- ◆ The act of adding detail, estimates and order to items in the Product Backlog
- ◆ Includes also: splitting large items into smaller ones, estimation of new items, and re-estimation of existing items.
- ◆ Is a part time activity during a Sprint between the Product Owner and the Development team
- ◆ **The Scrum Team decides how and when refining is done**
- ◆ However, Product Backlog items can be updated at any time by the Product Owner (or at the Product Owner's discretion)
- ◆ Refining usually consumes no more than 10% of the capacity of the Development team
- ◆ Product Backlog items that can be "Done" by the Development Team within one Sprint are deemed "Ready" for selection in a Sprint Planning
- ◆ Purpose:
  1. Preparation for the next (two) sprint(s)
  2. Insight into the long term (for team and product owner)

## INVEST

- ◆ mnemonic created by Bill Wake
- ◆ as a reminder of the characteristics of a good quality user story

I ndependent

N egotiable

V aluable

E stimable

S mall

T estable

## Testable

- ◆ Before starting development of a User Story: add Acceptance Criteria to provide a way to clearly demonstrate if the Development Team has indeed made the User Story come true
- ◆ Guideline: three to five acceptance criteria, but no more than eight crisply defined tests that are the conditions of acceptance for a user story
- ◆ Acceptance Criteria must be expressed clearly, in simple language the customer would use, without ambiguity as to what the expected outcome is: what is acceptable and what is not acceptable.
- ◆ Acceptance Criteria must be testable: easily translated into one or more manual/automated test cases
- ◆ They should include functional and non-functional criteria

## User Story with acceptance criteria

***As an Administrator, I want to be able to create User Accounts so that I can grant users access to the system***

### Acceptance criteria (examples) :

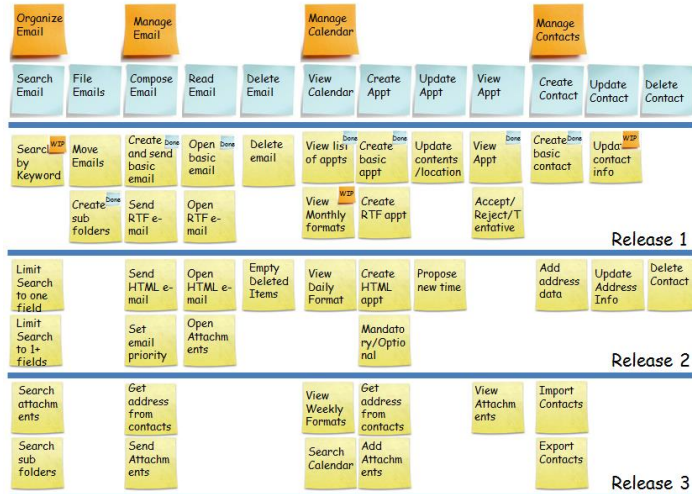
- ◆ If I am an Administrator, I can create User Accounts
- ◆ I can create a User Account by entering the following information about the User: a. Name, b. Email address, c. Phone Number d. License Number (Power/Basic/None), e. Account Status (Active/Inactive), f. Reports to (from a list of "Active" Users)
- ◆ I cannot assign a new User to report to an "Inactive" User
- ◆ I cannot assign a new User to report to a User if it creates a cyclical relationship (e.g., User 1 reports to User 2 who reports to User 1)
- ◆ The system notifies me that it sent an email to the new User's email address, containing a system-generated initial password and instructions for the person to log in and change their password.
- ◆ I am able to verify with the intended recipient of the email that it was received.

## Example definition of ready

- ◆ Acceptance Criteria defined
- ◆ Dependencies identified
- ◆ Size  $\leq$  1/4 of the velocity
- ◆ There is a demo scenario for the Product Backlog item
- ◆ Person who will accept the Product Backlog item is identified

*A definition of ready can be helpful (but is not mandatory)*

## Story mapping



## Story mapping

Allows you to add a second dimension to your backlog

To create your own user story map:

1. Form a group of 3-5 people who understand the purpose of the product
2. Start by gathering the major **"User Tasks"** of the application. Likely each post-it starts with a verb (eg. Compose E-mail, Create Contact, Add User, etc) *blue post-its*
3. Next ask the team to group the post-its
4. Using another colour of post-it, name each group (the groups are called **"User Activities"**) and put the post-it on top of the group *orange post-its*
5. Arrange the groups left to right in the order a user would typically complete the tasks
6. Add more detailed **user stories** below each user task before breaking your map into releases. You can use user scenarios and persona's. Once you have a good set of stories, then you can put your **release lines** in the map and ask your users to prioritize top to bottom *yellow post-its*

- ◆ Because all the stories end up on small post-its, you can forgo the traditional 'as a' format and just put the story title on the post-its.
- ◆ Do some serious user story slicing to make sure you have sliced the first release as thin as possible.

## Story mapping

Allows you to add a second dimension to your backlog

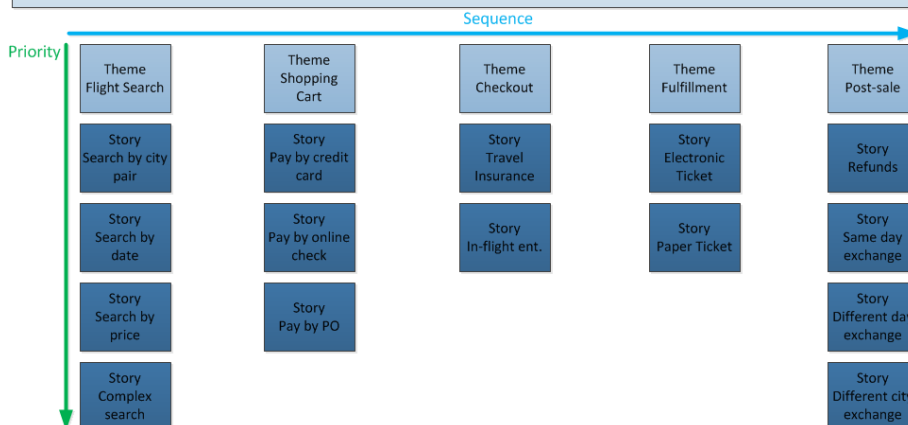
To create your own user story map:

1. Form a group of 3-5 people who understand the purpose of the product
2. Start by gathering the major “**User Tasks**” of the application. Likely each post-it starts with a verb (eg. Compose E-mail, Create Contact, Add User, etc) *blue post-its*
3. Next ask the team to group the post-its
4. Using another colour of post-it, name each group (the groups are called “**User Activities**”) and put the post-it on top of the group *orange post-its*
5. Arrange the groups left to right in the order a user would typically complete the tasks
6. Add more detailed **user stories** below each user task before breaking your map into releases. You can use user scenarios and persona's. Once you have a good set of stories, then you can put your **release lines** in the map and ask your users to prioritize top to bottom *yellow post-its*

- ◆ Because all the stories end up on small post-its, you can forgo the traditional 'as a' format and just put the story title on the post-its.
- ◆ Do some serious user story slicing to make sure you have sliced the first release as thin as possible.

## Story mapping

### Epic: Buy a plane ticket



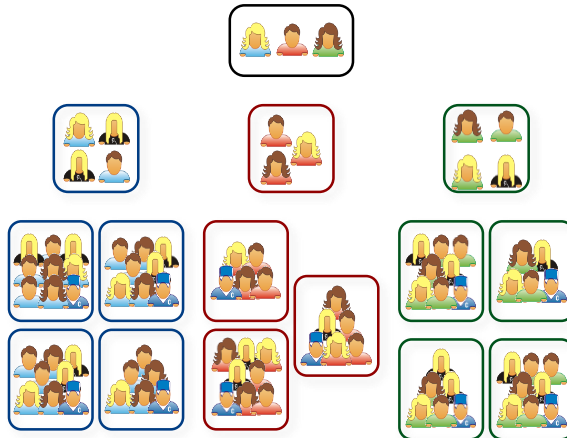




## The Development Team

- ◆ Delivers a potentially releasable product increment at the end of each Sprint
- ◆ Only members of the Development Team create the increment
- ◆ Structured and empowered by the organization to organize and manage their own work (the synergy optimizes the team's efficiency and effectiveness)
- ◆ Self-organizing, cross-functional
- ◆ Everybody has title Developer, regardless of the work being performed by the person
- ◆ Accountability belongs to team as a whole
- ◆ No sub teams
- ◆ Small: min 3, max 9
- ◆ The Development team is responsible for all estimates
- ◆ Demonstrates the working functionality to the Product Owner

# Scrum (of Scrums) of Scrums



*The most important concern for multiple Development Teams working on a single product: minimizing dependencies between teams*

Based on material from: Mountain Goat Software, LLC



## The Scrum Master

- ◆ The servant-leader for the Scrum Team
- ◆ Responsible for ensuring Scrum is understood and enacted
  - ◆ By ensuring that the Scrum team adheres to Scrum theory, practices and rules
- ◆ Helps those outside the ST understand which interactions are helpful and which aren't and helps everyone change these interactions to maximize the value created by the ST
- ◆ Serves Product Owner, Team and Organisation in several ways:
  - ◆ Finding techniques for effective Product Backlog management
  - ◆ Teaching the Development team to create clear and concise Product Backlog items
  - ◆ Coaching the team
  - ◆ Removing impediments to the Development Team's progress
  - ◆ Facilitating Scrum events as requested or needed
  - ◆ Causing change that increases the productivity of the Scrum Team
  - ◆ Leading and coaching the organization in its Scrum adoption



## Prescribed timeboxed events

- ◆ To create regularity and to minimize the need for meetings not defined in Scrum
- ◆ The Sprint is a container for all events
- ◆ Every event has a maximum duration
- ◆ Once a Sprint begins its duration is fixed
  - ◆ The remaining events may end whenever the purpose is achieved, ensuring an appropriate amount of time is spent without allowing waste in the process
- ◆ To enable Transparency, Inspection and Adaptation (feedback loops)

## Sprint

- ◆ A time-box of one month or less (fixed length), during which a useable and potentially releasable product increment is created
- ◆ Sprints best have consistent durations throughout a development effort
- ◆ A new Sprint starts immediately after the conclusion of the previous Sprint
- ◆ During the sprint:
  - ◆ No changes are made that would endanger the Sprint Goal
  - ◆ Quality goals do not decrease
  - ◆ Scope may be clarified end re-negotiated between the Product Owner and Development Team as more is learned
- ◆ Only the Product Owner has the authority to cancel the Sprint (it rarely makes sense)

## Sprint (continued)

- ◆ Each Sprint may be considered a project with no more than a one-month horizon
  - ◆ each sprint has a definition of what is to be built, a design and flexible plan that will guide building it
  - ◆ when a sprint's horizon is too long the definition of what is being built may change, complexity may rise, and risk may increase
- ◆ Sprints enable predictability by ensuring inspection and adaptation of progress toward a Sprint Goal at least every calendar month
- ◆ When a Sprint's horizon is too long, you increase the risk that what is being developed may no longer be desired
- ◆ Sprints limit risk to one calendar month or less of work (and cost)

## Sprint Planning

*Timeboxed to a maximum of 8h for a one-month Sprint, usually less for shorter sprints*

Answers the following:

- ◆ What can be done this Sprint? *Topic One*
- ◆ How will the chosen work get done? *Topic Two*

Input

- ◆ Product Backlog, ordered and estimated
- ◆ The latest product Increment
- ◆ **Development Team capacity for this sprint**
- ◆ Past performance of the Development Team

Attendees

- ◆ Scrum Team
- ◆ The Development team may also invite other people to provide additional business or technical information or advice

## Sprint Planning

### *Topic One: what can be done this Sprint?*

- ◆ The Development Team works to forecast the functionality that will be developed during the Sprint
- ◆ The Product Owner discusses the objective that the Sprint should achieve and the Product Backlog items that (if completed) would achieve the Sprint Goal
- ◆ The number of items selected from the Product Backlog for the Sprint is solely up to the Development Team
- ◆ After the Development Team forecasts the Product Backlog items it will deliver in the Sprint, the Scrum Team crafts a Sprint Goal

## Sprint Goal

A Sprint Goal:

- ◆ provides guidance to the DT on why it is building the increment and
- ◆ gives the team some flexibility regarding the functionality implemented within the Sprint and
- ◆ causes the DT to work together rather than on separate initiatives

During the Sprint: if the work turns out to be different than the DT expected, they collaborate with the Product Owner to negotiate the scope of Sprint Backlog

Example: *"This Sprint we will create a messaging service for customers to keep them updated about relevant flight information during their trip"*

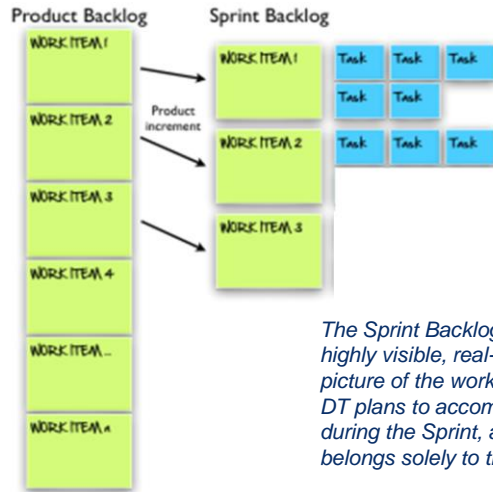
## Sprint Planning

### *Topic Two: How will the chosen work get done*

- ◆ The DT usually starts by designing the system and the work needed to convert the Product Backlog into a working product Increment
- ◆ Work may be of varying size
- ◆ Enough work is planned during the Sprint Planning for the DT to forecast what it believes it can do in the upcoming Sprint
- ◆ Work planned for the first days of the sprint is decomposed to units of one day or less
- ◆ The Product Owner can help clarify the selected Product Backlog items and make trade offs
- ◆ If the DT determines it has too much or too little work, it may renegotiate the selected Product Backlog items with the Product Owner.

## Sprint Planning

*Enough work is planned during the Sprint Planning for the DT to forecast what it believes it can do in the upcoming Sprint*



*The Sprint Backlog is a highly visible, real-time picture of the work that the DT plans to accomplish during the Sprint, and it belongs solely to the DT*

## Sprint Backlog

- ◆ The Sprint Backlog: the Product Backlog Items selected for this Sprint plus the plan for delivering the product Increment and realizing the Sprint Goal
- ◆ Is a forecast by the DT about what functionality will be in the next Increment and the work needed to deliver that functionality into a “done” Increment
- ◆ The Sprint Backlog is a plan with enough detail that changes in progress can be understood in the Daily Scrum
- ◆ The DT modifies the Sprint Backlog throughout the Sprint, and the Sprint Backlog emerges during the Sprint
- ◆ This emergence occurs as the DT works through the plan and learns more about the work needed to achieve the Sprint Goal
- ◆ As work is performed or completed, the estimated remaining work is updated
- ◆ When elements of the plan are deemed unnecessary, they are removed

## Sprint backlog day x



Kniberg

## Daily Scrum

*Timebox: 15 minutes*

- ◆ Every day same time and place to reduce complexity
- ◆ Only for the Development Team
- ◆ To synchronize activities and create a plan for the next 24 hours
- ◆ Optimizes the probability that the DT will meet the Sprint Goal
- ◆ Three questions:
  - ◆ *What did I do yesterday that helped the DT meet the Sprint Goal?*
  - ◆ *What will I do today to help the DT meet the Sprint Goal?*
  - ◆ *Do I see any impediment that prevents me or the DT from meeting the Sprint Goal?*



## Sprint Review

*Timeboxed to a maximum of 4h for a one-month Sprint, usually less for shorter sprints*

- ◆ Held at the end of the Sprint to inspect the Increment and adapt the Product Backlog if needed (*an informal meeting, not a status meeting*)
- ◆ The presentation of the Increment is intended to elicit feedback and foster collaboration.
- ◆ The result of the Sprint Review is a revised Product Backlog that defines the probable Product Backlog items for the next Sprint. The Product Backlog may also be adjusted overall to meet new opportunities

### Attendees

- ◆ Scrum Team
- ◆ Key Stakeholders invited by the Product Owner

## Sprint Review (continued)

It includes:

- ◆ Development Team discusses what went well during the Sprint, what problems it ran into, and how these problems were solved
- ◆ The Development team presents functionality that it has “Done” and answers questions about the increment
- ◆ The Product Owner discusses the Product Backlog as it stands. He or she projects likely completion dates based on progress to date (if needed)
- ◆ The entire group collaborates on what to do next, so that the Sprint Review provides valuable input to subsequent Sprint Planning
- ◆ Review of the marketplace or potential use of the product might have changed what is the most valuable thing to do next
- ◆ Review of the timeline, budget, capabilities and marketplace for the next anticipated release of the product

## Sprint Retrospective

*Timeboxed to a maximum of 3h for a one-month Sprint, usually less for shorter sprints*

Purpose:

- ◆ Inspect how the last Sprint went with regards to people, relationships, process, and tools
- ◆ Identify: what went well? What can be improved?
- ◆ Create a plan for implementing improvements (in the next Sprint), although improvements may be implemented at any time

The Definition of Done can be adapted (to increase product quality)

A dedicated event focused on Inspection and Adaptation

Attendees:

- ◆ The Scrum Team



Global Knowledge.



## Increment

- ◆ The purpose of each Sprint is to deliver an increment of potentially shippable functionality that adhere to the Scrum Team's Definition of "Done"
- ◆ This increment is useable, so a Product Owner may choose to immediately release it
- ◆ Each increment is additive to all prior Increments and thoroughly tested, ensuring that all increments work together

## Definition of done

- ◆ Is used to assess when work is complete on the product increment
- ◆ Team members must have a shared understanding of what it means for work to be complete (to ensure transparency)
- ◆ Can vary significantly per Scrum Team
- ◆ The development organization (or Development Team if none is available from the development organization) creates the Definition of done (as a minimum)
  - ◆ The Development Team of the Scrum Team can complement it with elements specific for the product or context
- ◆ Feedback from retrospective meetings should help to improve the definition of done (as Scrum Teams mature)
- ◆ It guides the DT in knowing how many Product Backlog items it can select during a Sprint Planning
- ◆ Non-functional requirements are often included in the Definition of Done

## Definition of Done

- ◆ A shared understanding of what it means for work to be complete

### Team "Done" List

<p><i>...With a Story</i></p> <ul style="list-style-type: none"> <li>• All Code (Test and Mainline) Checked in</li> <li>• All Unit Tests Passing</li> <li>• All Acceptance Tests Identified, Written &amp; Passing</li> <li>• Help File Auto Generated</li> <li>• Functional Tests Passing</li> </ul>	<p><i>...With a Sprint</i></p> <p>All Story Criteria, Plus...</p> <ul style="list-style-type: none"> <li>• Product Backup Updated</li> <li>• Performance Testing</li> <li>• Package, Class &amp; Architecture Diagrams Updated</li> <li>• All Bugs Closed or Postponed</li> <li>• Code Coverage for all Unit Tests at 80% +</li> </ul>
<p><i>...Release to INT</i></p> <p>All Sprint Criteria, Plus...</p> <ul style="list-style-type: none"> <li>• Installation Packages Created</li> <li>• MOM Packages Created</li> <li>• Operations Guide Updated</li> <li>• Troubleshooting Guides Updated</li> <li>• Disaster Recovery Plan Updated</li> <li>• All Test Suites Passing</li> </ul>	<p><i>...Release to Prod</i></p> <p>All INT Criteria, Plus...</p> <ul style="list-style-type: none"> <li>• Stress Testing</li> <li>• Performance Tuning</li> <li>• Network Diagram Updated</li> <li>• Security Pass Validated</li> <li>• Threat Modeling Pass Validated</li> <li>• Disaster Recovery Plan Tested</li> </ul>



Global Knowledge.



## Manage and measure value

- ◆ The value delivered by a product is not only determined by revenue, value is likely to vary across products and organizations
- ◆ The Product Owner manages the Product Backlog against the assumption that value will be generated
- ◆ This assumption remains invalidated when not checked against users and market
- ◆ Indications of value on the Product Backlog (value points) are useful but are only a prediction until validated against users and market
- ◆ Market reception is the best measure of value
- ◆ Through frequent delivery of Increments of the product into the market a Product Owner can obtain feedback from users and the market
- ◆ What helps the Product Owner to manage the value of a product:
  - ◆ Validating assumptions of value through frequent releases
  - ◆ The order of the Product Backlog

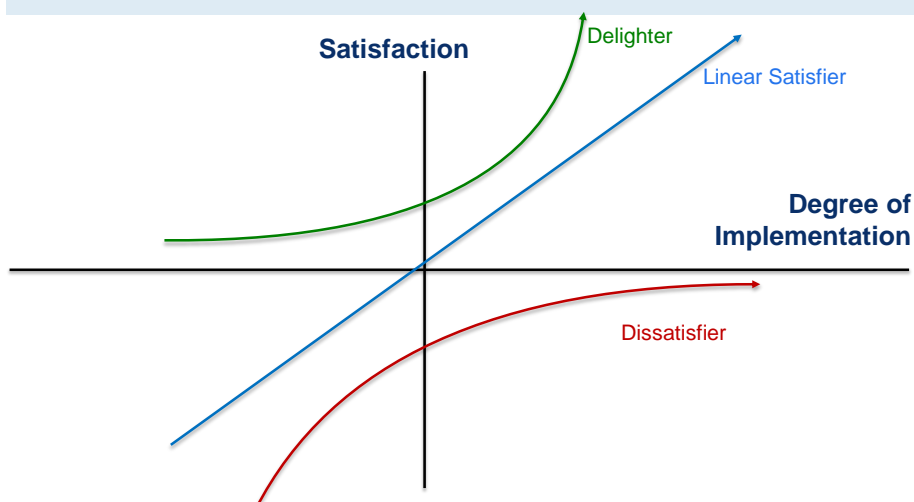
## Validated learning

- ◆ Learning turns into 'validated learning' when assumptions and goals can be assessed through results
- ◆ A key way for a Product Owner to apply validated learning is releasing an Increment to the market to learn about the business assumptions built into the product.
- ◆ In a startup productivity is not about how much “stuff” you are building, but how much validated learning you are getting for your efforts

## Kano Model Analysis

- ◆ Kano Model Analysis is a useful technique for deciding which features you want to include in a product
- ◆ It helps you think more subtly about the features you include.
- ◆ It helps you develop products that will truly delight your customers.
- ◆ According to the Kano Model (developed by Dr Noriaki Kano in the 1980s), a product can have three types of attribute (or property):
  - ◆ Baseline Expectations: Threshold attributes which customers expect to be present in a product
  - ◆ Linear Satisfiers: Performance attributes which are not absolutely necessary, but which are known about and increase the customer's enjoyment of the product.
  - ◆ Delighters: Excitement attributes which customers don't even know they want, but are delighted when they find them
- ◆ Brainstorm possible features and where possible, get your customers to do the classification for you

## Prof. Kano-model for customer preferences



## Investments and budgeting

- ◆ The Product Owner is not only accountable for development and release of a product, but also the cost of maintaining and operating the product: the Total Cost of Ownership (TCO)
- ◆ Budgeting is ideally revisited as frequently as each Sprint to ensure value is being delivered
- ◆ Ken Schwaber recommends in his book *Agile Project Management with Scrum*: The Product Owner's focus is on return on investment (ROI)
  - ◆ This means that product owners will have to look after products over an extended period of time – at least until ROI can be determined – if not after the product's entire lifecycle

## Technical Debt

- ◆ Technical debt is a concept in programming that reflects the extra development work that arises when code that is easy to implement in the short run is used instead of applying the best overall solution
- ◆ Technical Debt comes from various sources, some of which can be good and some bad
- ◆ A little debt speeds development so long as it is paid back promptly with a rewrite
- ◆ The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt
- ◆ It implies that restructuring existing code (refactoring) is required as part of the development process

## Technical Debt limits the value

Technical debt limits the value a Product Owner can get from a product:

- ◆ it causes a greater percentage of the product's budget to be spent on maintenance of the product
- ◆ the velocity at which new functionality can be created is reduced when you have technical debt

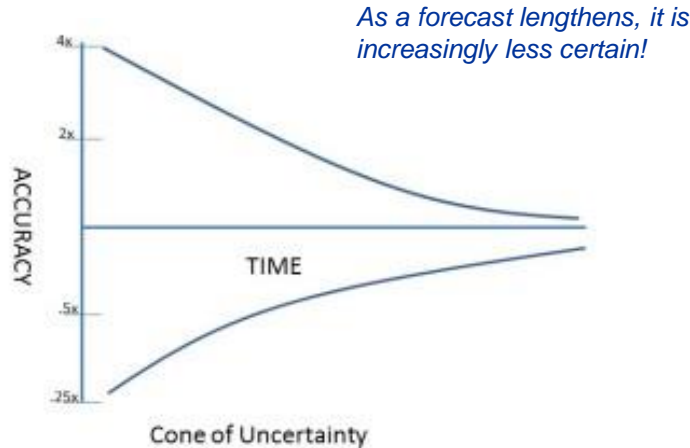


Global Knowledge.

## Planning, Estimating and Monitoring progress

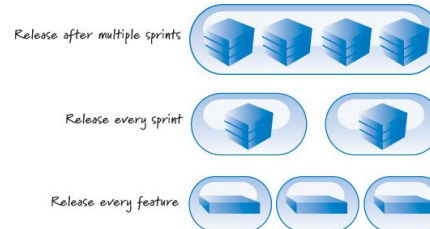


## Cone of uncertainty



## Planning in Scrum

- ◆ The Product Backlog is the plan. It is updated as new information and insights emerge
- ◆ The Sprint Backlog contains the plan for the Sprint
- ◆ It is possible to successfully use Scrum without using release planning
- ◆ Release planning (or “*Medium Term Planning*”, *Mike Cohn*) is still valuable in many cases: a way to come up with a plan, a roadmap of what can be expected after a couple of sprints, and what will be part of the release and what won't
- ◆ It is not always practical to constantly deliver what is being built. Scrum requires that a product Increment at the end of a Sprint be potentially *usable*. This does not mean that it is necessarily *useful*. So, there is often a need to batch usable Increments from subsequent Sprints into something that is potentially useful for a customer



## Using story points for estimating user stories

- ◆ Used to measure the effort required to implement a user story.
- ◆ Relative estimates of the complexity, effort or duration of a story
- ◆ Makes it easy for scrum teams to think abstract about the effort required to complete a story (points are unit-less)
- ◆ Find a Baseline story. A one that all in the team can relate too. Give it size 2 or 3 (story points) for instance
- ◆ From then on all sizing should be done compared to that baseline.
- ◆ Precision decreases as story size increases (therefore: constrain estimates to specific pre-defined values such as ½, 1,2,3,5,8,13,20,40,100)
- ◆ Tasks are often estimated in hours (in Sprint Planning)

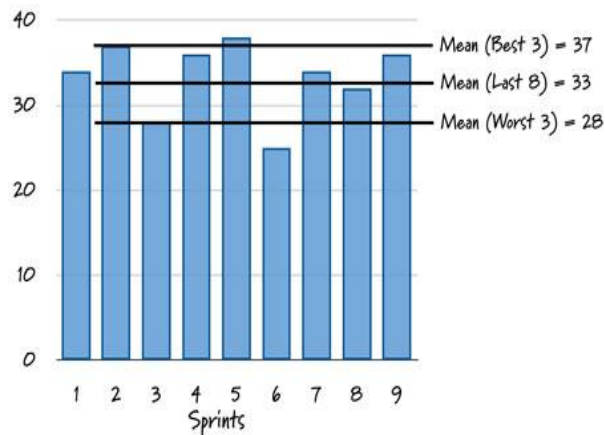
## Planning poker



## Velocity

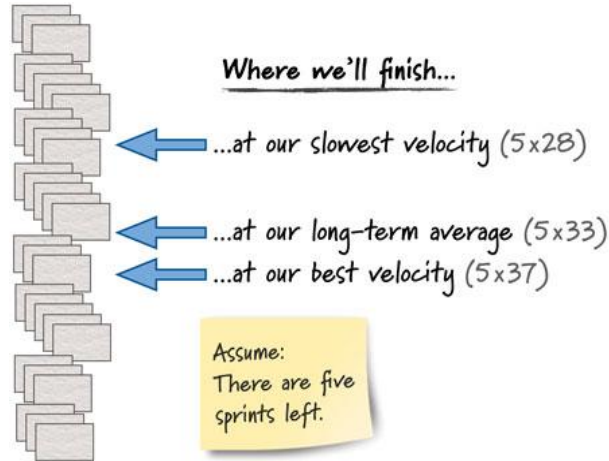
- ◆ Velocity in Scrum is how much product backlog effort a team can handle in one sprint
- ◆ Can be estimated by viewing previous sprints, assuming the team composition and sprint duration are kept constant
- ◆ Often a good estimation can be reached after 3 sprints
- ◆ Once established, velocity can be used to plan projects and forecast release and product completion dates.

## Velocity

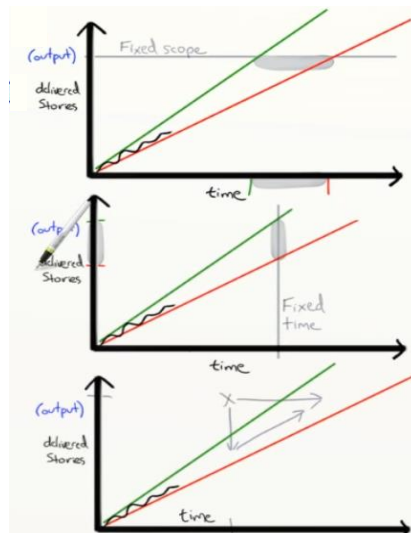


# Medium term planning

The product backlog is the plan



# Optimistic and pessimistic forecast



Fixed scope

Fixed time

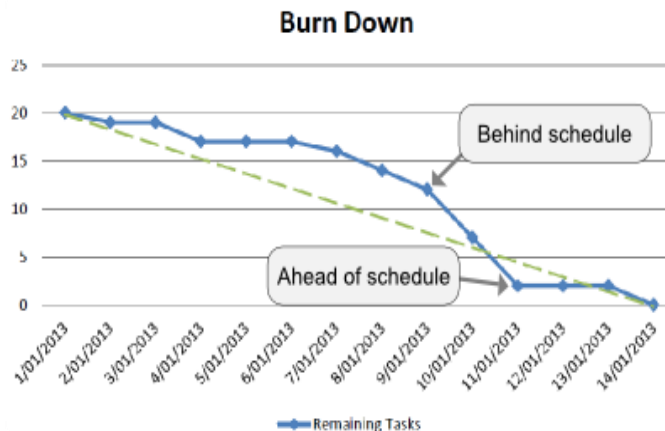
Fixed time preferable to fixed scope...

Henrik Kniberg

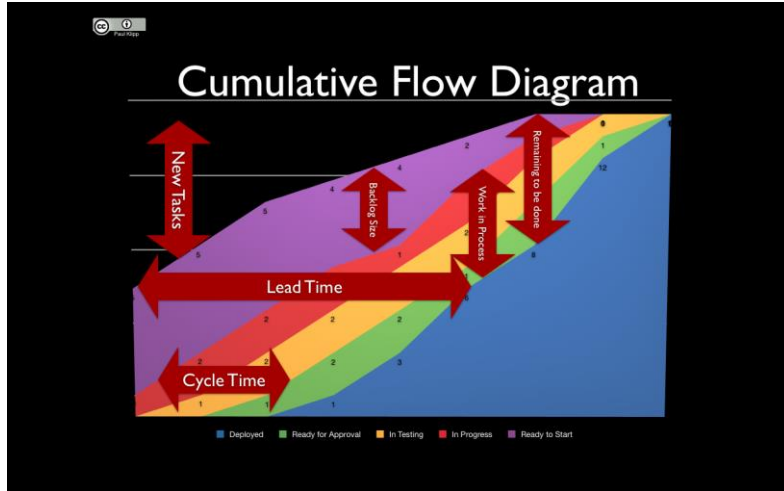
## Monitoring progress toward a goal

- ◆ At any point in time, the total remaining work to reach a goal can be summed
- ◆ The Product Owner tracks this total remaining work at least every Sprint Review
  - ◆ The Product Owner compares this amount with work remaining at previous Sprint Reviews to assess progress toward completing projected work by the desired time for the goal
- ◆ This information is made transparent to all stakeholders
- ◆ Various projective practices upon trending can be used, like burn-downs, burn ups, or cumulative flows
  - ◆ However, these do not replace the importance of empiricism: what will happen is unknown. Only what has happened may be used for forward-looking decision-making
- ◆ The Development tracks the total remaining work at least every Daily Scrum to project likelihood of achieving the Sprint Goal. By doing so the Development Team can manage its progress

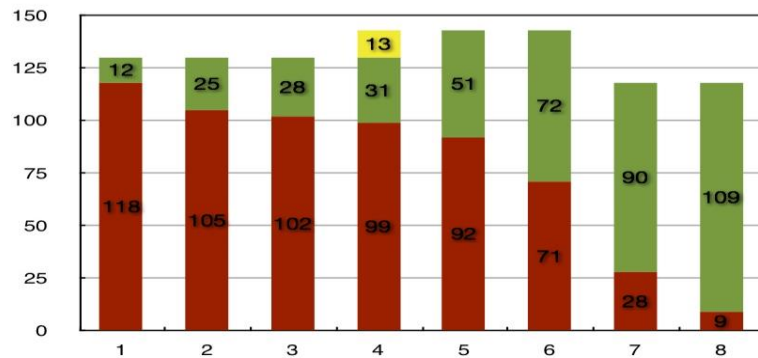
## Track progress: Sprint burn-down chart



## Track progress: Cumulative flow diagram

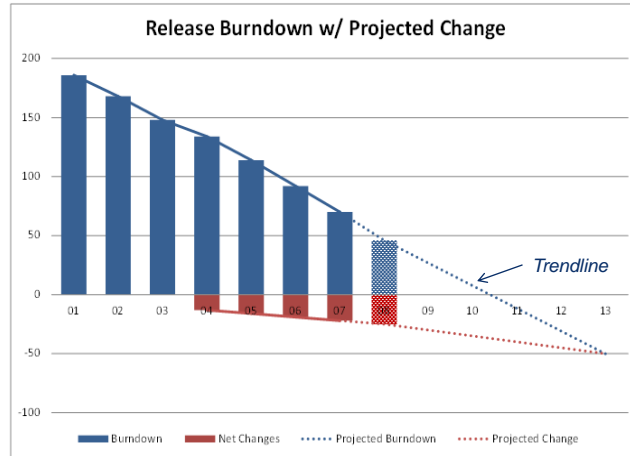


## Track progress: Burn down chart



At the start : 130 story points planned in 8 sprints      Delivered    Remained    Increased scope

## Track progress: Release Burn-down chart



*Trendline*: when the work remaining will likely be completed if nothing changes on the backlog or the Development Team