Hochschule Hof

# Agile Project Management

## Scrum

**Prof. Dr. Matthias Meitner**

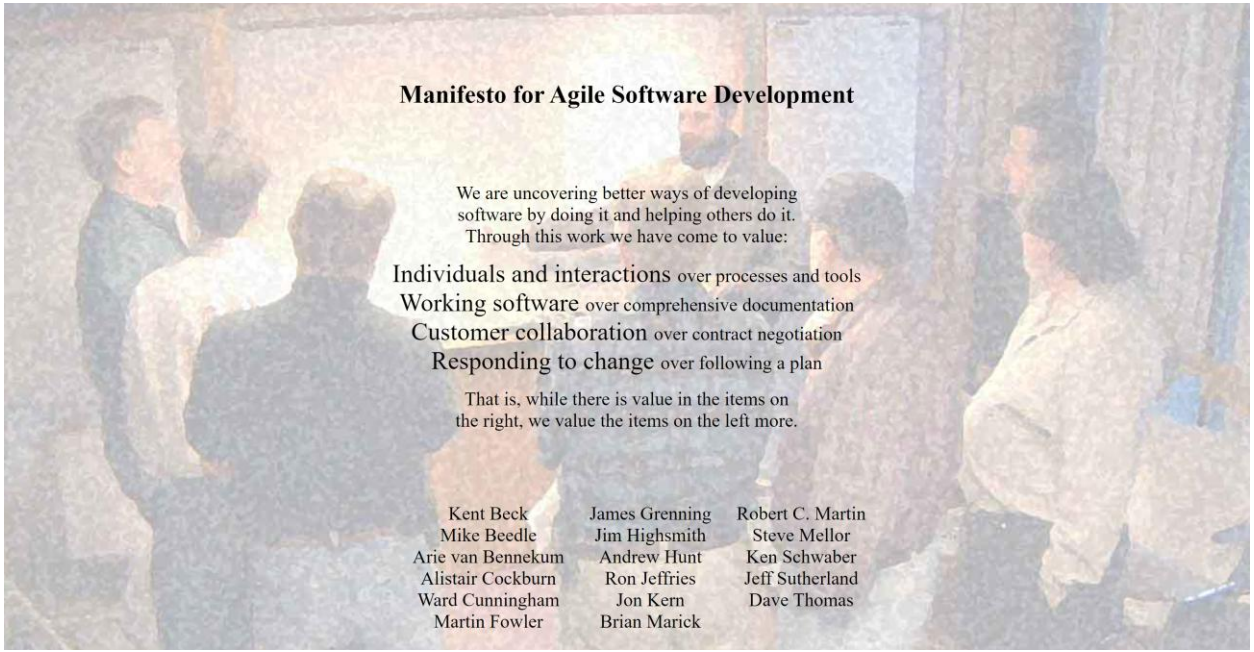
# Literature

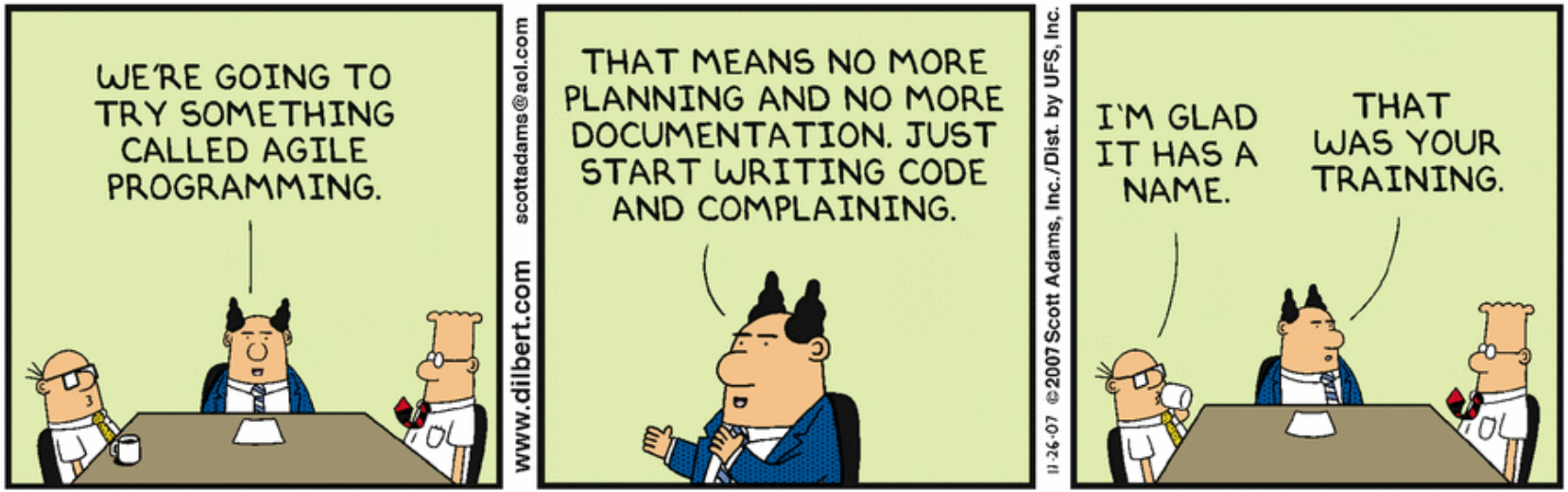

- Ken Schwaber, Jeff Sutherland: The Scrum Guide
- Jeff Sutherland, James Coplien: A Scrum Book: The Spirit of the Game, O'Reilly UK Ltd.
- Kenneth S. Rubin: Essential Scrum: Practical Guide to the Most Popular Agile Process, Addison-Wesley Signature
- Mike Cohn: User Stories Applied: For Agile Software Development, Addison-Wesley Professional
- Jeff Patton: User Story Mapping: Discover the Whole Story, Build the Right Product, O'Reilly and Associates

# Scrum

- Agile
- Scrum Introduction
- The Scrum Guide
- User Stories
- Product Discovery
- Scrum Monitoring

# http://agilemanifesto.org




WE'RE GOING TO TRY SOMETHING CALLED AGILE PROGRAMMING.
THAT MEANS NO MORE PLANNING AND NO MORE DOCUMENTATION. JUST START WRITING CODE AND COMPLAINING.
I'M GLAD IT HAS A NAME.
THAT WAS YOUR TRAINING.
www.dilbert.com scottadams@aol.com
11-26-07 ©2007 Scott Adams, Inc./Dist. by UFS, Inc.

# 12 Agile principles

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.

# 12 Agile principles

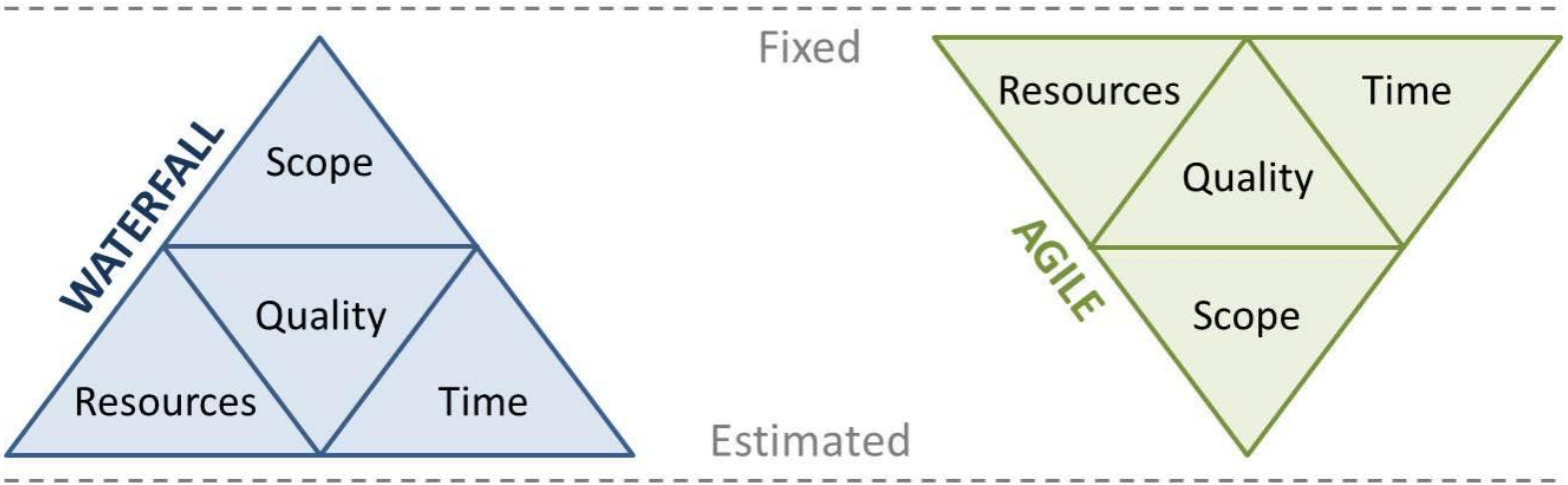
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity – the 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.


DILBERT
BY SCOTT ADAMS
ARE THESE USER SPECIFICATIONS COMPLETE?
I ASK BECAUSE ANY LATER CHANGES WILL CAUSE ME TO MISS THE DEADLINE.
WHAT IF I ONLY NEED A TINY CHANGE LATER?
I'M COUNTING ON IT.
THAT WAY I CAN BLAME YOU WHEN I MISS THE DEADLINE.
HOW DO MOST PEOPLE HANDLE THIS SITUATION?
WELL, THE PESSIMISTS KNOW THEY'RE DOOMED, SO IT'S NO SURPRISE TO THEM WHEN IT HAPPENS.
WHAT DO THE OPTI-MISTS DO?
THEY BECOME PESSI-MISTS.
DilbertCartoonist@gmail.com
© 2016 Scott Adams, Inc./Dist. by Universal Uclick
www.dilbert.com
9-4-16

# Exercise
# Agile principles

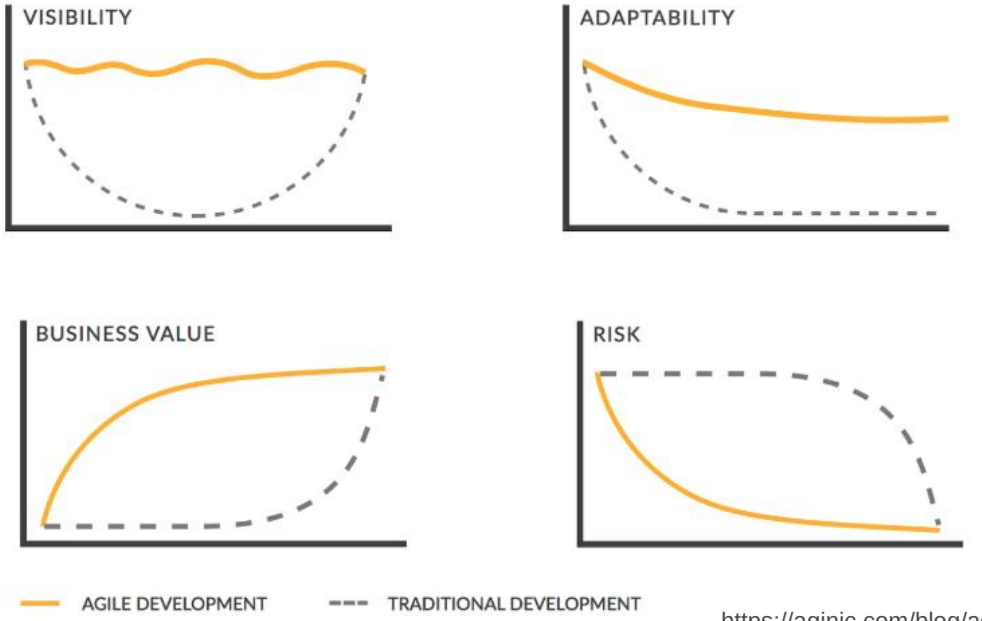
- Find the 3 most important agile principles from your point of view!

# Triple constraints
# Waterfall vs Agile



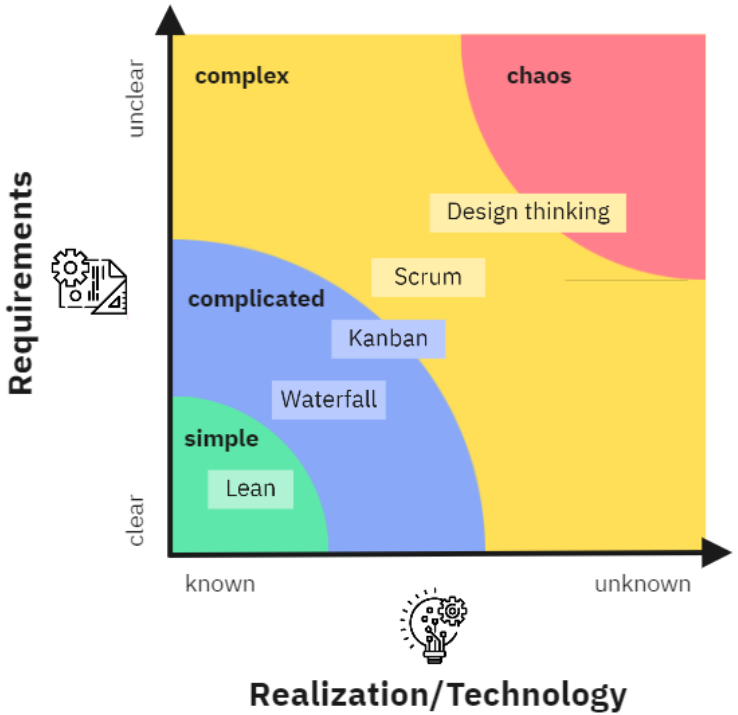

https://www.scrum.org/resources/blog/scrum-myths-scrum-conflicts-fixed-dates

# Benefits of Agile




https://aginic.com/blog/agile-delivery-comparison/

# When to use agile?
# Stacey Matrix



# Traditional vs agile project management

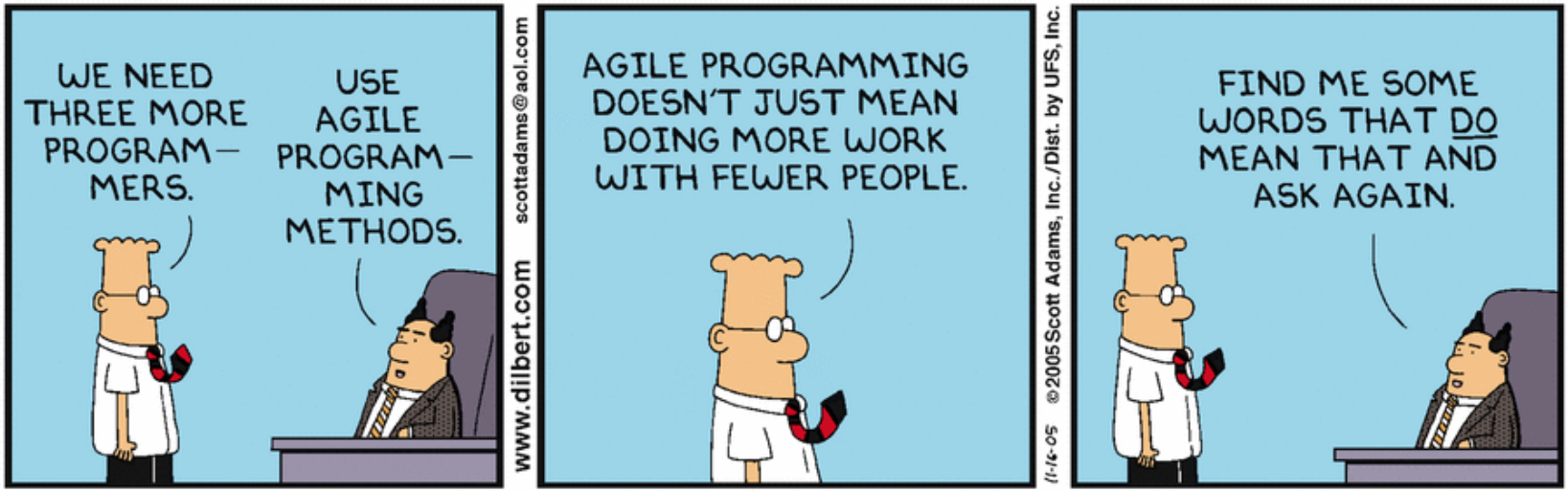
- Requirements known at the beginning
- Changes to requirements are difficult
- High costs for late changes to requirements
- Requirements description from a technical point of view
- Sequential development process
- Rigid project management process
- Customer only sees end result

- Requirements fuzzy at the beginning
- Changes to requirements are expected
- Moderate costs for late changes to requirements
- Requirements description from customer perspective
- Iterative development process
- Continuous process improvements
- Customer evaluates intermediate results

# Traditional vs agile project management

- When things get tight, more likely to push milestones
- Large teams possible
- Clear hierarchy
- Many specialists in the team
- Team is distributed and active in several projects
- Assign tasks from the top
- Much communication via documents and long meetings
- Effort estimation by project manager or experts

- When things get tight, rather reduce effort
- Relatively small teams
- Self-managed teams
- Much shared responsibility
- Team sits together and has focus on one project
- Take on tasks independently
- Lots of informal communication and standup meetings
- Effort estimation together as a team


WE NEED THREE MORE PROGRAM— MERS.
USE AGILE PROGRAM— MING METHODS.
www.dilbert.com scottadams@aol.com
AGILE PROGRAMMING DOESN'T JUST MEAN DOING MORE WORK WITH FEWER PEOPLE.
11-14-05 ©2005 Scott Adams, Inc./Dist. by UFS, Inc.
FIND ME SOME WORDS THAT DO MEAN THAT AND ASK AGAIN.

# Scrum

- Agile
- Scrum Introduction
- The Scrum Guide
- User Stories
- Product Discovery
- Scrum Monitoring

# Why is it called “Scrum”?

# What is Scrum?

- Scrum is **not a standardized process** where you methodically follow a series of sequential steps that are guaranteed to produce, on time and on budget, a high-quality product that delights customers.
- Scrum is a **framework** for organizing and managing work.
- The Scrum framework is based on a set of values, principles and practices that provide the **foundation** to which an organization will add its **unique implementation** of relevant engineering practices and its specific approaches for realizing the Scrum practices.

# Uses of Scrum

- Scrum has been used to develop software, hardware, autonomous vehicles, schools, government, marketing, managing the operation of organizations and almost everything we use in our daily lives, as individuals and societies.
- As technology, market and environmental complexities and their interactions have rapidly increased, Scrum's utility in dealing with **complexity** is proven daily.
- Scrum proved especially effective in **iterative** and **incremental** knowledge transfer. Scrum is now widely used for products, services and the management of the parent organization.
- In **interrupt-driven environments** (software maintenance and support areas) you would be better off considering an alternative agile approach called *Kanban*. Kanban is not a stand-alone process solution, but instead an approach that is overlaid on an existing process.

# Scrum

- Agile
- Scrum Introduction
- The Scrum Guide
- User Stories
- Product Discovery
- Scrum Monitoring

# Scrum definition

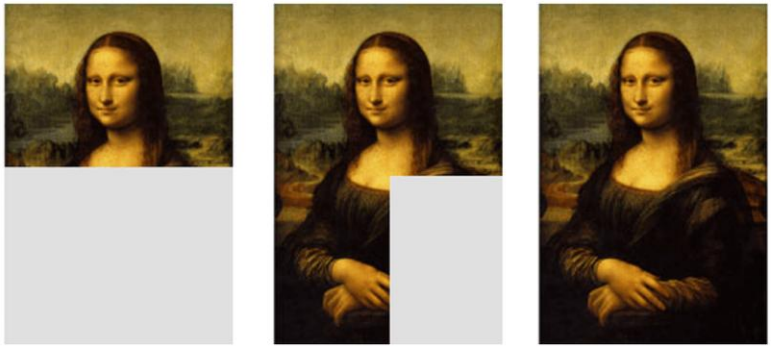
- Scrum is a **lightweight framework** that helps people, teams and organizations generate value through adaptive solutions for complex problems.
- Scrum is **simple**.
- The Scrum framework is **purposefully incomplete**, only defining the parts required to implement Scrum theory.
- Rather than provide people with detailed instructions, the rules of Scrum guide their **relationships** and **interactions**.
- Various processes, techniques and methods can be employed within the framework. Scrum **wraps around existing practices** or renders them unnecessary.
- Scrum makes visible the relative efficacy of current management, environment and work techniques, so that **improvements** can be made.

# Scrum theory

- Scrum is founded on empiricism and lean thinking.
  - **Empiricism** asserts that knowledge comes from experience and making decisions based on what is observed.
  - **Lean thinking** reduces waste and focuses on the essentials.
- Scrum employs an **iterative**, **incremental** approach to optimize predictability and to control risk.
- Scrum engages **groups of people** who collectively have **all the skills** and expertise to do the work and share or acquire such skills as needed.
- Scrum combines four formal **events** for inspection and adaptation within a containing event, the **Sprint**.
- These events work because they implement the empirical Scrum **pillars** of **transparency**, **inspection** and **adaptation**.

## Iterative

## Incremental



from: https://www.zsolt.blog/2021/01/agile-product-management-in-roam-part-1.html

# #1 Empirical Scrum pillar Transparency

- The emergent process and **work must be visible** to those performing the work as well as those receiving the work. With Scrum, important decisions are based on the perceived state of its three formal **artifacts**.
- Artifacts that have low transparency can lead to decisions that diminish value and increase risk. **Transparency enables inspection**. Inspection without transparency is misleading and wasteful.

# #2 Empirical Scrum pillar Inspection

- The Scrum artifacts and the progress toward agreed goals must be inspected frequently and diligently to **detect potentially undesirable variances** or problems. To help with inspection, Scrum provides cadence in the form of its five events.
- **Inspection enables adaptation**. Inspection without adaptation is considered pointless. Scrum events are designed to **provoke change**.

# #3 Empirical Scrum pillar Adaptation

- If any aspects of a process deviate outside acceptable limits or if the resulting product is unacceptable, the process being applied or the materials being produced must be **adjusted**. The adjustment must be made as soon as possible to **minimize further deviation**.
- Adaptation becomes more difficult when the people involved are not empowered or self-managing. A Scrum Team is expected to adapt the moment it **learns** anything new through inspection.

## Scrum values

- Commitment
- Focus
- Openness
- Respect
- Courage

# Scrum values

- **Commitment** can transform a team. It’s a **promise** to yourself, your teammates and your organization to do the very best work you can. If everyone on a Scrum Team is committed to delivering an Increment of valuable product each and every Sprint, they can accomplish great things together.
- **Focus** allows us to do our very best. Valuing focus means that we give people the time they need to think about their work. After all, creativity is hard enough without being constantly interrupted. Allowing Developers to focus just on one product, the current Sprint and the current Sprint Goal gives them the best chance of succeeding.

# Scrum values

- **Openness** is the **core of transparency**, which is what makes Scrum work. If the members of a Scrum Team aren't open with each other and the wider organization, they can't solicit honest feedback or adapt their work accordingly. You need to be open and honest, even when you're struggling or there's a tough issue to address.
- **Respect** creates a feeling of **safety**. Being open with others can be scary and admitting when you're stuck is hard, but respect makes these actions easier. A high-performing Scrum Team is built on mutual respect and honest discussions create the safety needed to tackle difficult issues.
- **Courage** is the linchpin of the other Scrum values. It takes courage to commit, to focus amid distractions and to be open to new ideas. And it takes courage and faith in your teammates to count on having respectful interactions when you need to discuss problems.

# Exercise
# Scrum values

- Assign the following behaviors to one of the five agile values!
  - Creating realistic goals and sticking to them
  - Being present in the meetings
  - Having no fear to raise impediments on a daily basis
  - Having a clear role and clear goals within that role
  - Collaborating across disciplines and skills, also sharing feedback and learning from one another
  - Accepting all different kinds of people who comprise a team
  - Trusting the Scrum process to guide the work needed to satisfy the requirements of the product

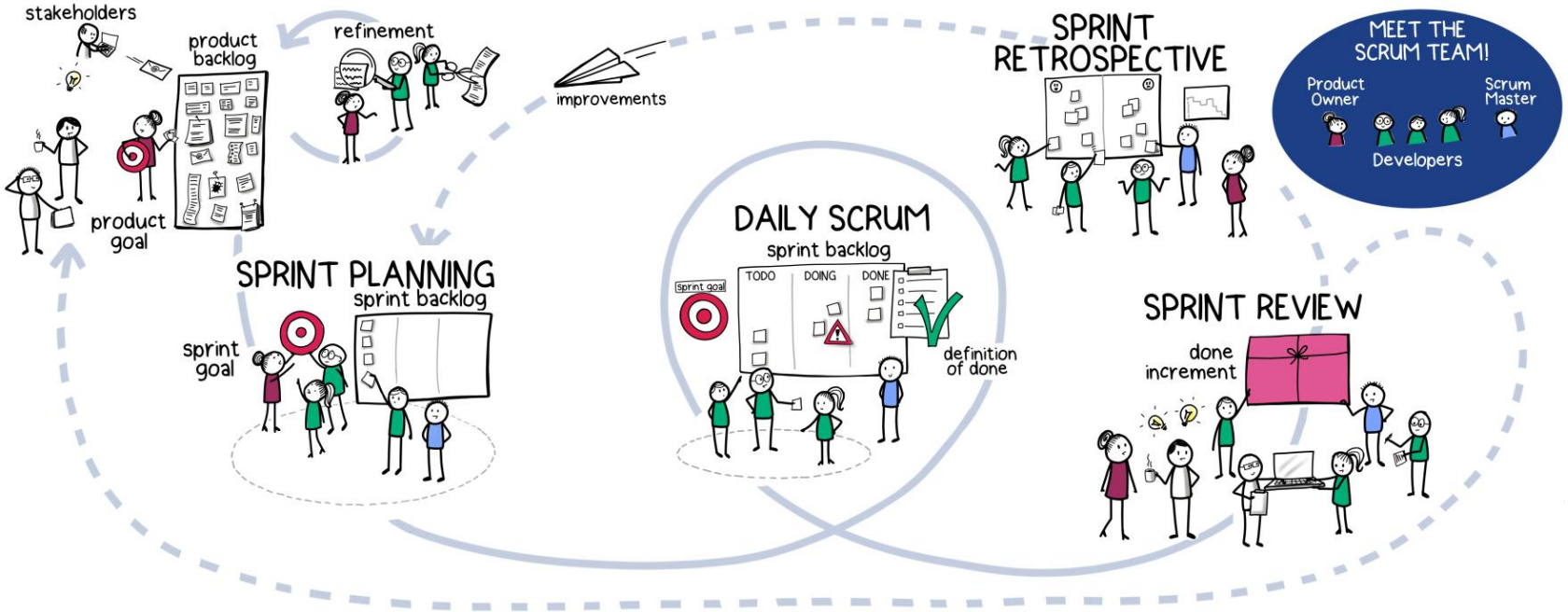
# Exercise
# Scrum values anti patterns

- Which of the five agile values do the following behaviors contradict?
  - Adding new features to the Sprint Backlog
  - Helping other teams
  - Keeping relevant knowledge to yourself
  - Reworking the code for your colleague
  - “I have always done it that way”-mentality
  - Not directly communicating with team members when there are issues

# Scrum overview

- 3 Accountabilities
  - Developers
  - Product Owner
  - Scrum Master
- 3 Artifacts
  - Product Backlog
  - Sprint Backlog
  - Product Increment
- 5 Events
  - Sprint
  - Sprint Planning
  - Daily Scrum
  - Sprint Review
  - Sprint Retrospective

# Scrum overview




from: theliberators.com

# Scrum

- Agile
- Scrum Introduction
- The Scrum Guide – Scrum Accountabilities
- User Stories
- Product Discovery
- Scrum Monitoring

# Scrum Team

- The Scrum Team is small enough to remain nimble and large enough to complete significant work within a Sprint, typically **10 or fewer people**.
- Scrum Teams are **cross-functional**, meaning the members have all the skills necessary to create value each Sprint. They are also **self-managing**, meaning they internally decide who does what, when and how.
- The Scrum Team is **responsible for all product-related activities** from stakeholder collaboration, verification, maintenance, operation, experimentation, research and development and anything else that might be required. They are structured and **empowered** by the organization to **manage their own work**.
- The entire Scrum Team is accountable for creating a **valuable, useful Increment every Sprint**. Scrum defines three specific accountabilities within the Scrum Team: the Developers, the Product Owner and the Scrum Master


WE'RE MOVING TO AN AGILE METHODOLOGY FOR SOFTWARE DEVELOPMENT.
I DON'T KNOW ALL OF THE DETAILS, BUT I THINK ONE OF YOU HAS TO BE DESIGNATED THE SCRUMBAG.
DOES THAT SOUND RIGHT?
IT'S BETTER THAN I EXPECTED.
Dilbert.com @ScottAdamsSays
© 2017 Scott Adams, Inc./Dist. by Andrews McMeel
2-6-17

# Developers

- Developers are the people in the Scrum Team that are committed to creating any aspect of a usable **Increment** each Sprint.
- The specific skills needed by the Developers are often broad and will vary with the domain of work.
- The Developers are always **accountable** for
  - Creating a plan for the Sprint, the Sprint Backlog
  - Instilling quality by adhering to a Definition of Done
  - Adapting their plan each day toward the Sprint Goal
  - Holding each other accountable as professionals

# Developers

- Traditional software development approaches define various job types, such as architect, programmer, tester, database administrator, UI designer and so on.
- Scrum defines the **accountability “Developers”**, which is simply a **cross-functional collection** of these types of people. In particular, the Developer is one of the three accountabilities on every Scrum Team.
- The Developers, collectively, have the **skills required to deliver the business value** requested by the Product Owner.
- They are self-managing and composed of members with a **T-shaped** skill profile.

# Product Owner

- The Product Owner is accountable for **maximizing the value of the product** resulting from the work of the Scrum Team. How this is done may vary widely across organizations, Scrum Teams and individuals.
- The Product Owner is also accountable for effective **Product Backlog management**, which includes:
  - Developing and explicitly communicating the Product Goal
  - Creating and clearly communicating Product Backlog items
  - Ordering Product Backlog items
  - Ensuring that the Product Backlog is transparent, visible and understood
- The Product Owner should rather be an **initiating** than a receiving role.
- The Product Owner accountability should be performed by **one person per product**, not a committee.

# Product Owner

- The Product Owner manages and is responsible for the **Product Backlog**.
- The Product Owner **maximizes the value** of what the Developers are working on by ordering the Product Backlog.
- The Product Owner is in charge of creating and maintaining the **Product Vision**.
- The Product Owner uses Scrum as a catalyst to inspect the product and adapt its direction based on **customer and stakeholder feedback**.
- The Product Owner keeps tabs on the marketplace and **adapts the product** with it.
- The Product Owner works with stakeholders to gather opinions and ideas but is **empowered to make final decisions**.

# Scrum Master

- The Scrum Master is accountable for **establishing Scrum** as defined in the Scrum Guide. They do this by helping everyone understand Scrum theory and practice, both within the Scrum Team and the organization.
- The Scrum Master is accountable for the **Scrum Team's effectiveness**. They do this by enabling the Scrum Team to **improve its practices**, within the Scrum framework.
- Scrum Masters are **true leaders** who serve the Scrum Team and the larger organization.

## Scrum Master

- The Scrum Master serves the Scrum Team in several ways, including
  - Coaching the team members in **self-management** and **cross-functionality**
  - Helping the Scrum Team focus on creating **high-value Increments** that meet the Definition of Done
  - Causing the removal of **impediments** to the Scrum Team's progress
  - Ensuring that all **Scrum events** take place and are positive, productive and kept within the timebox

# Scrum Master

- The Scrum Master serves the Product Owner in several ways, including
  - Helping find techniques for effective **Product Goal** definition and **Product Backlog management**
  - Helping the Scrum Team understand the need for clear and concise **Product Backlog items**
  - Helping establish empirical **product planning** for a complex environment
  - Facilitating **stakeholder collaboration** as requested or needed

# Scrum Master

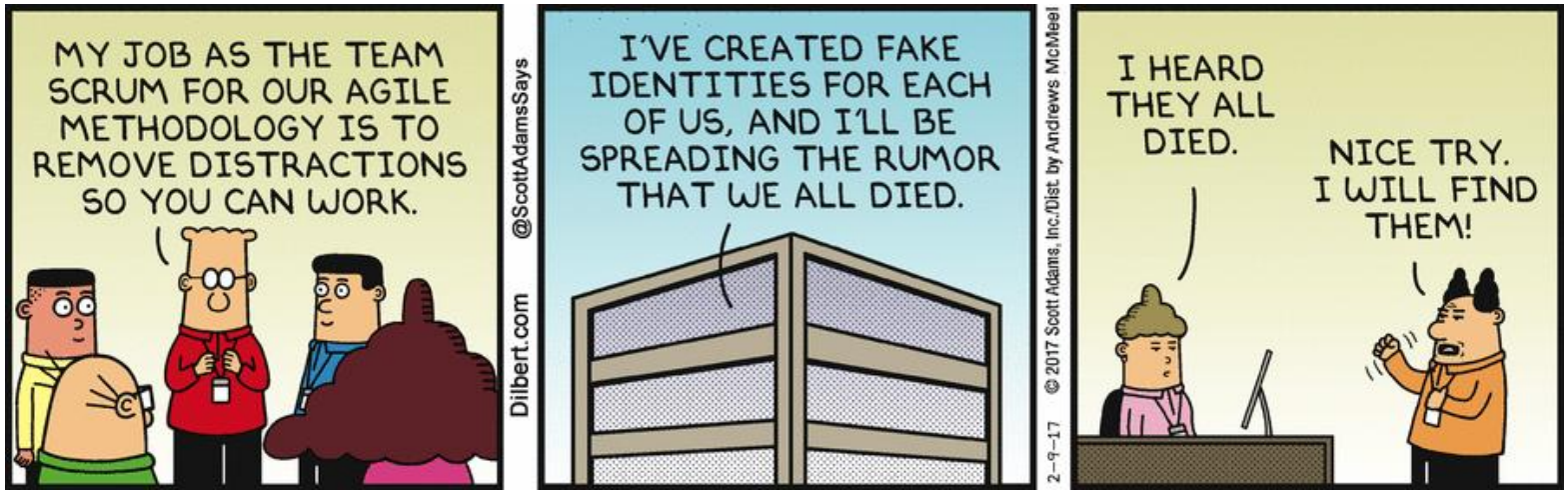
- The Scrum Master serves the organization in several ways, including
  - Leading, training and coaching the organization in its **Scrum adoption**
  - Planning and advising **Scrum implementations** within the organization
  - Helping employees and stakeholders understand and enact an **empirical approach** for complex work
  - **Removing barriers** between stakeholders and Scrum Teams

# Removing impediments

- First rule for a Scrum Master: “**Help people to help themselves**! Do not solve the problems for others. Instead, help the Scrum Team and the developers to solve the problems themselves.”
- We often distinguish between **blockages** and **obstacles**. Blockages prevent further work on a task. Obstacles make the team slower or more ineffective than it could be. However, further work is possible.
- Examples of blockages include:
  - Missing supplies
  - Open technical questions that cannot be clarified immediately
  - Failure of specialized qualifications that are only available once in the team

# Removing impediments

- Examples of obstacles are:
  - Developers don’t have enough time (because they have to do other tasks)
  - Unclear goals / product vision
  - Interference of the Developers by superiors or stakeholders
  - No understanding of the organization for Scrum
  - No understanding of the responsibilities in Scrum
  - Chief architect in the team who makes decisions for the others
  - Poorly equipped computers
  - Missing or poor development tools
  - No team room
  - Rarely available meeting rooms


MY JOB AS THE TEAM SCRUM FOR OUR AGILE METHODOLOGY IS TO REMOVE DISTRACTIONS SO YOU CAN WORK.
Dilbert.com @ScottAdamsSays
I'VE CREATED FAKE IDENTITIES FOR EACH OF US, AND I'LL BE SPREADING THE RUMOR THAT WE ALL DIED.
2-9-17 © 2017 Scott Adams, Inc./Dist. by Andrews McMeel
I HEARD THEY ALL DIED.
NICE TRY. I WILL FIND THEM!

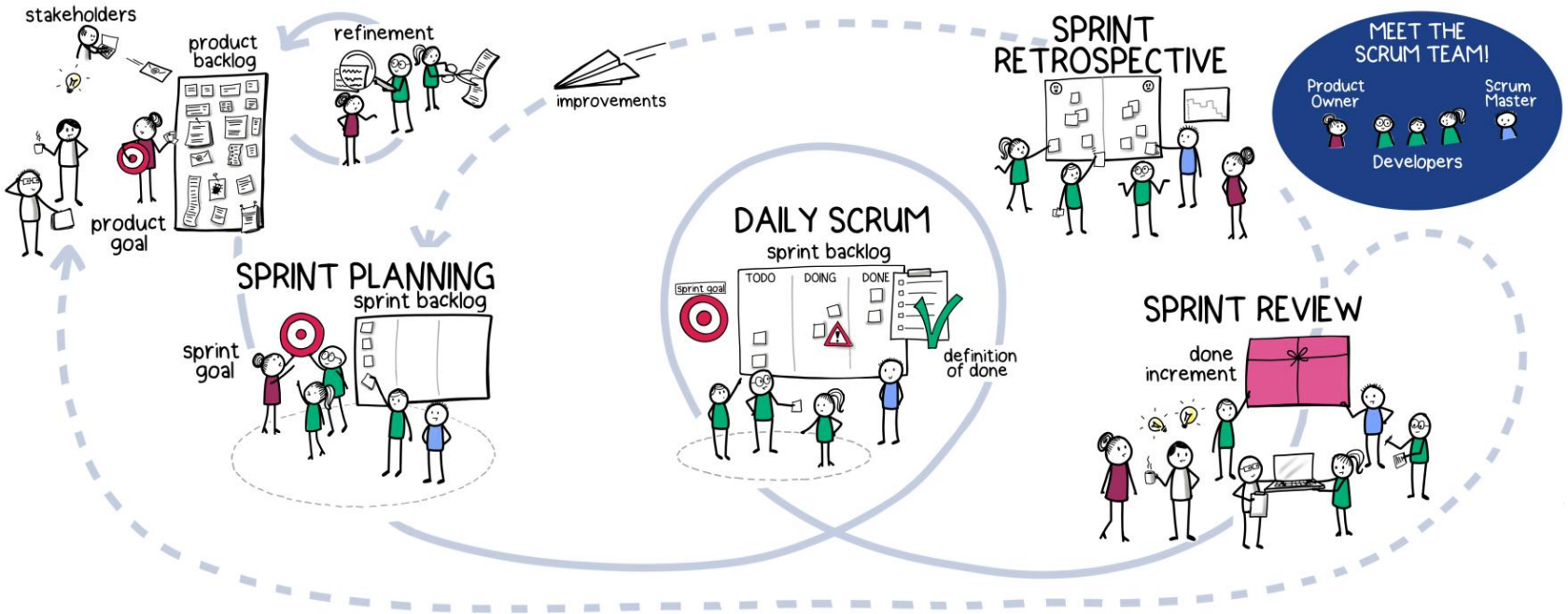
# Questions

- Is it ok if the Scrum Master is also a Developer?
- Is it ok if the Product Owner and Scrum Master are one and the same person?
- What do you think of having a part-time Scrum Master in your Scrum Team?

# Scrum

- Agile
- Scrum Introduction
- The Scrum Guide – Scrum Events
- User Stories
- Product Discovery
- Scrum Monitoring

# Scrum overview




from: theliberators.com

# Scrum events

- The **Sprint** is a **container** for all other events. Each event in Scrum is a formal **opportunity to inspect and adapt** Scrum artifacts. These events are specifically designed to enable the transparency required.
- Events are used in Scrum to create **regularity** and to minimize the need for meetings not defined in Scrum. Optimally, all events are held at the same time and place to reduce complexity.
- Scrum events
  - Sprint
  - Sprint Planning
  - Daily Scrum
  - Sprint Review
  - Sprint Retrospective

# Sprint

- Sprints are the **heartbeat of Scrum**, where ideas are turned into value.
- They are **fixed length** events of one month or less to create consistency. A new Sprint starts immediately after the conclusion of the previous Sprint.
- All the work necessary to achieve the Product Goal, including Sprint Planning, Daily Scrums, Sprint Review and Sprint Retrospective, happen within Sprints.
- During the Sprint
  - No changes are made that would endanger the Sprint Goal.
  - Quality does not decrease.
  - The Product Backlog is refined as needed.
  - Scope may be clarified and renegotiated with the Product Owner as more is learned.

# Sprint

- Sprints enable **predictability** by ensuring inspection and adaptation of progress toward a Product Goal at least every calendar month. When a Sprint's horizon is too long the Sprint Goal may become invalid, complexity may rise and risk may increase. Shorter Sprints can be employed to generate more learning cycles and limit risk of cost and effort to a smaller time frame. Each Sprint may be considered a short project.
- Various practices exist to **forecast progress**, like burn down charts. While proven useful, these do not replace the importance of **empiricism**. In complex environments, what will happen is unknown. Only what has already happened may be used for **forward-looking decision making**.
- A Sprint could be cancelled if the Sprint Goal becomes **obsolete**. Only the Product Owner has the authority to cancel the Sprint.

## Exercise
## What are the benefits of using a short Sprint length?

## Exercise
## What are the benefits of using a short Sprint length?

- Ease of planning
- Fast feedback
- Bounded error
- Improved return on investment
- Rejuvenated excitement
- Frequent checkpoints

# Sprint termination

- When a Sprint is cancelled, any completed and “Done” Product Backlog items are **reviewed** and **accepted** by the Product Owner if they are potentially releasable. All incomplete Product Backlog items are re-estimated and put back on the Product Backlog.
- Next Sprint length after abnormal Sprint termination
  1. Stay with the original Sprint length – This has the advantage of keeping a uniform Sprint length throughout development. If multiple Scrum Teams are collaborating on the same development effort, using the original Sprint length will put the Scrum Team that terminated its Sprint out of sync with the other teams.
  2. Make the next Sprint just long enough to get to the end date of the terminated Sprint.
  3. Make the next Sprint longer than a normal Sprint to cover the remaining time in the terminated Sprint plus the time for the next full Sprint.

## Scrum Planning principles

- You can‘t get the plans right up front.
- Up-front planning should be helpful without being excessive.
- Keep planning options open until the **last responsible moment**.
- Focus more on **adapting and replanning** than on conforming to a plan.
- You should correctly manage the planning inventory.
- You should favor **smaller** and more **frequent releases**.
- You should plan to **learn fast** and **pivot** when necessary.

# Sprint Planning

- Sprint Planning initiates the Sprint by laying out the work to be performed for the Sprint. This resulting **plan** is created by the collaborative work of the entire Scrum Team.
- The Product Owner ensures that attendees are prepared to discuss the most important Product Backlog items and how they map to the **Product Goal**.
- The Scrum Team may also invite other people to attend Sprint Planning to provide **advice**.

# Sprint Planning topics

- Topic One: **“Why is this Sprint valuable?”**
  - The Product Owner proposes how the product could increase its value and utility in the current Sprint. The whole Scrum Team then collaborates to define a **Sprint Goal** that communicates why the Sprint is valuable to stakeholders. The Sprint Goal must be finalized prior to the end of Sprint Planning.
- Topic Two: **“What can be done this Sprint?”**
  - Through discussion with the Product Owner, **the Developers select items** from the Product Backlog to include in the current Sprint. The Scrum Team may refine these items during this process, which increases understanding and confidence.
  - Selecting how much can be completed within a Sprint may be challenging. However, the more the Developers know about their past performance, their upcoming **capacity** and their Definition of Done, the more confident they will be in their Sprint forecasts.

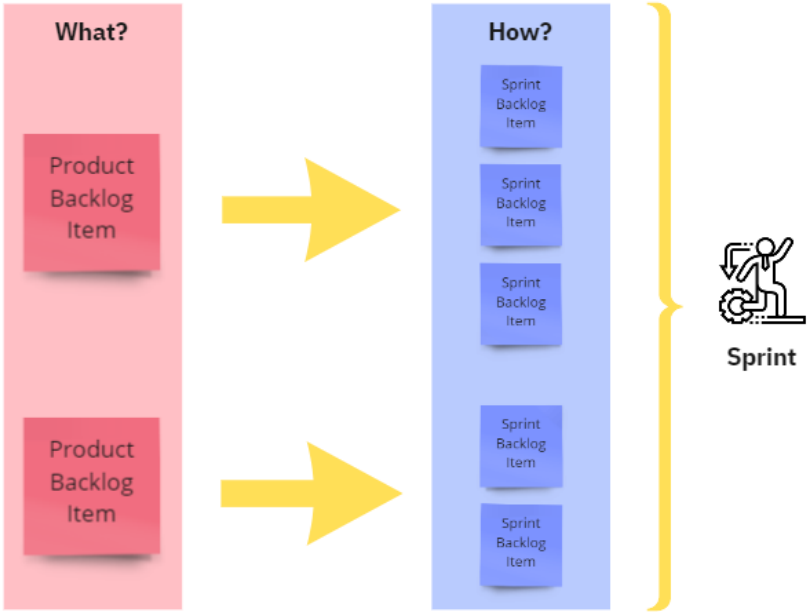
# Sprint Planning topics

- Topic Three: **“How will the chosen work get done?”**
  - For each selected Product Backlog item (PBI), the Developers plan the work necessary to create an Increment that meets the Definition of Done. This is often done by **decomposing Product Backlog items** into smaller work items of one day or less. How this is done is at the sole discretion of the Developers. No one else tells them how to turn Product Backlog items into Increments of value.
  - The Sprint Goal, the Product Backlog items selected for the Sprint, plus the plan for delivering them are together referred to as the **Sprint Backlog**.
  - Sprint Planning is **timeboxed** to a maximum of eight hours for a one-month Sprint. For shorter Sprints, the event is usually shorter.

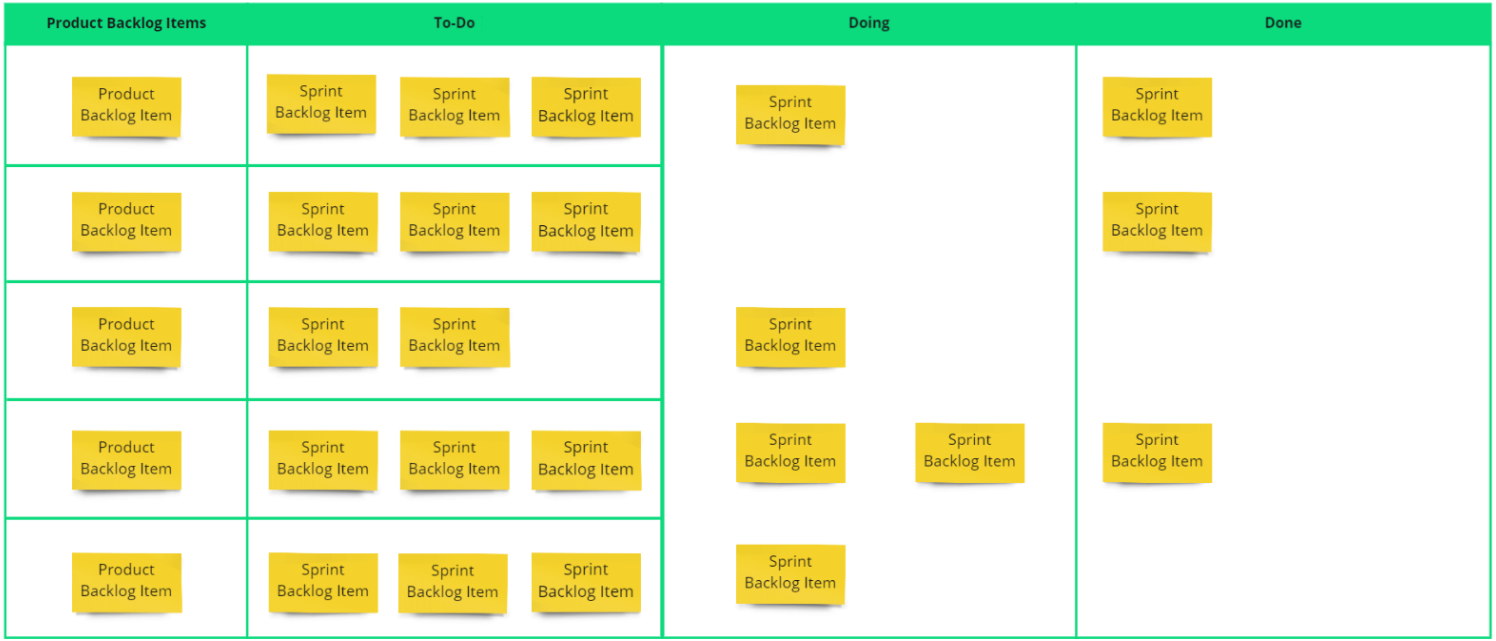
# From Product Backlog items to Sprint Backlog items

- A **Product Backlog item** (PBI) does not define how to arrive at a solution or implementation.
- Therefore, the selected PBIs are broken down into **work items** and are assembled into a plan called Sprint Backlog. Each work item is a Sprint Backlog item (SBI).
- Sprint Planning only produces the **initial version** of the Sprint Backlog.
- The collection of SBIs is **dynamic** and changes as the team learns more about the product.
- SBIs can emerge as **discovered work**.
- The team should estimate new SBIs as they emerge.
- It is a common practice to write SBIs on sticky notes and put them on the **Scrum Board** to make progress on work visible.

# From Product Backlog items to Sprint Backlog items



# Scrum board


Product Backlog Items
To-Do
Doing
Done
Product Backlog Item
Sprint Backlog Item

# Definition of Ready

- Refining the Product Backlog should ensure that items at the top of the Product Backlog are **ready to be moved into a Sprint** so that the Developers can confidently commit and complete them by the end of a Sprint.
- Some Scrum Teams formalize this idea by establishing a **Definition of Ready**. The Definition of Ready and the Definition of Done can be considered as two states of Product Backlog items during a Sprint.
- Both the Definition of Done and the Definition of Ready are **checklists** of the work that must be completed before a Product Backlog item can be considered to be in the respective state.

# Exercise
# Definition of Ready

- When can a Product Backlog item be considered “ready” to be moved into the next Sprint?

## Definition of Ready Example

- Business value is clearly articulated.
- Details are sufficiently understood by the Developers so it can make an informed decision as to whether it can complete the PBI.
- Dependencies are identified and no external dependencies would block the PBI from being completed.
- Team is staffed appropriately to complete the PBI.
- The PBI is estimated and small enough to be completed in one Sprint.
- Acceptance criteria are clear and testable.
- Scrum Team understands how to demonstrate the PBI at the Sprint Review.

# Sprint Planning questions

- “What did we learn at Sprint Review and Sprint Retrospective that needs to be considered in our planning conversations today?”
- “What is at the top of the Product Backlog?”
- “What do we need to know about these Product Backlog items in order to fully commit to getting them done in the upcoming Sprint?”
- “How much of the Product Backlog do we think we can complete in the upcoming Sprint?”
- “On which features do we need more clarification?”

from: https://www.lucidchart.com/blog/what-questions-to-ask-agile-team

# Sprint Planning questions

- “How is this yet-to-build Product Backlog item similar to other work we have done in the past? How does this comparison help us estimate the relative size of the new work?”
- “What outside help will we need to succeed with our plans?”
- “What’s the biggest risk that may prevent us from completing this Sprint?”
- “Who is taking time off during the next Sprint? How should we plan differently given the team’s (adjusted) capacity?”

from: https://www.lucidchart.com/blog/what-questions-to-ask-agile-team

# Daily Scrum

- The purpose of the Daily Scrum is to **inspect progress** toward the Sprint Goal and **adapt** the Sprint Backlog as necessary, adjusting the upcoming planned work.
- The Daily Scrum is a **15-minute** event for the Developers of the Scrum Team. To reduce complexity, it is held at the same time and place every working day of the Sprint. If the Product Owner or Scrum Master are actively working on items in the Sprint Backlog, they participate as Developers.
- The Developers can select whatever structure and techniques they want, as long as their Daily Scrum focuses on progress toward the Sprint Goal and produces an **actionable plan for the next day of work**. This creates focus and improves self-management.

# Daily Scrum

- Daily Scrums
  - improve communications
  - identify impediments
  - promote quick decision-making
  - eliminate the need for other meetings
- The Daily Scrum is not the only time Developers are allowed to adjust their plan. They often meet throughout the day for more detailed discussions about adapting or re-planning the rest of the Sprint’s work.

# Collaboration during Daily Scrum

- **Focus on the work** – Use your team board to guide the conversation. For each PBI discuss **blockers**, progress and the tasks the team will work on today. Seek insights from multiple team members. Doing so will give everyone a shared understanding of what work is in progress, where the dependencies are and how any risks will be mitigated.
- **Keep the team board up to date** – There are people in your organization who want true status reports. By encouraging the Developers to keep their board current, you help create a tool that can provide valuable information to others. Management can walk by the board and see at a glance how things are progressing. This transparency into the work is a great way to build trust with stakeholders.
- **Keep the Developer's eyes on the Sprint Goal** – The purpose of the Sprint Goal is to remind the team why they're building the current Increment of software. As new work, ideas or decisions come up in the Daily Scrum, use the Sprint Goal as your guidepost to filter out the noise and keep the team focused on the high-priority work.

# Benefits of the Daily Scrum

- **Reduces time wasted** because the team makes impediments visible daily
- Helps find opportunities for **coordination** because everyone knows what everyone else is working on
- Reinforces a **shared vision** of the Sprint Goal
- Promotes **team building** by ensuring at least one global interaction every day
- Promotes **knowledge sharing** and the identification of knowledge gaps
- Increases the overall sense of **urgency**
- Encourages trust and honesty among Developers through a **verifiable daily status**
- Strengthens the culture of the Developers through **shared rituals** and encouragement of active participation

# Daily Scrum
# Questions proposed in the Scrum Guide 2017

- “What did I do yesterday that helped the Development Team meet the Sprint Goal?”
- “What will I do today to help the Development Team meet the Sprint Goal?”
- “Do I see any impediment that prevents me or the Development Team from meeting the Sprint Goal?”

## Daily Scrum
## Alternative set of questions

- “Are we on track with the Sprint Goal?”
- “Are we working on the most important items?”
- “Is anything stuck?”
- “If something is stuck, how can we all work together to get this work unstuck?”
- “Who needs help?”
- “What’s the most important thing we need to accomplish today?”
- “How do we increase the odds that the most important things get to done?”
- “Have we identified any new risks?”

# Sprint Review

- The purpose of the Sprint Review is to **inspect the outcome** of the Sprint and determine future adaptations. The Scrum Team presents the results of their work to key **stakeholders** and progress toward the Product Goal is discussed.
- During the event, the Scrum Team and stakeholders **review** what was accomplished in the Sprint and what has changed in their environment. Based on this information, attendees collaborate on **what to do next**. The Product Backlog may also be adjusted to meet new opportunities. The Sprint Review is a working session and the Scrum Team should avoid limiting it to a presentation.
- The Sprint Review is the second to last event of the Sprint and is timeboxed to a maximum of four hours for a one-month Sprint. For shorter Sprints, the event is usually shorter.

# Sprint Review

- The Sprint Review gives everyone with input to the product development effort an opportunity to inspect and adapt what has been built so far. The Sprint Review provides a **transparent look** at the current state of the product, including any inconvenient truths. It is the time to ask questions, make observations or suggestions and have discussions about how to best move forward given current realities.
- Because it helps ensure that the organization is creating a **successful product**, the Sprint Review is one of the most important learning loops in the Scrum framework.
- And, because Sprints are short, this loop is a quick one, which allows for **frequent course corrections** to keep the product development moving in the right direction. If, instead, we were to defer this feedback until much later and assume that everything is going according to some baseline plan, we likely would get what many are accustomed to – surprise, disappointment and frustration.

# Sprint Review agenda

- Explanation of the Product Owner’s **vision** for the product
- Review of the release plan and budget
- Description of the **Sprint Goal** for this Sprint
- **Review** of the work and the Product Increment that the team built
- **Hands-on** exploration of the product by the users
- Assessment of current **customer analytics** and data
- Discussion of **impediments** the team is currently facing
- Collaboration with stakeholders to **refine** the Product Backlog
- Summary of where we’re going next and what we discovered in this event

# Sprint Review questions

- “On a scale from 0 to 10, how likely is it that you’d recommend this product to friend or colleague of yours? Why would you give that number?”
- “Imagine that this product is purchasable on Amazon. How many stars would you give? What review would it get from you?”
- “Would you use this product for yourself? Why? Why not?”
- “Would you use this product for the next 5 years? Why? Why not?”

# Sprint Review questions

- “If you could change only one thing about this product, what would you change? Why is that?”
- “Imagine one product feature can/should be eliminated, which one is this for you? Why did you choose this feature?”
- “What’s your favorite feature or characteristic about this product? Why is that?”
- “From which other product on the market could we learn from? Why is that?”
- “How can we make this product really bad? What should we do to make this a nightmare of a product? Follow-up: Now how can we avoid doing any of these actions? What should we do instead of these actions?”

# Sprint Retrospective

- The purpose of the Sprint Retrospective is to plan ways to **increase quality and effectiveness**.
- The Scrum Team inspects how the last Sprint went with regards to **individuals**, **interactions**, **processes**, **tools** and their **Definition of Done**.
- The Scrum Team discusses what went well during the Sprint, what problems it encountered and how those problems were (or were not) solved.
- The Scrum Team identifies the most helpful changes to improve its effectiveness. The most impactful improvements are addressed as soon as possible. They may even be **added to the Sprint Backlog** for the next Sprint.
- The Sprint Retrospective concludes the Sprint. It is timeboxed to a **maximum of three hours** for a one-month Sprint. For shorter Sprints, the event is usually shorter.

# Sprint Retrospective

- **Inputs** to the Sprint Retrospective include the agreed-upon focus for the Sprint Retrospective and any exercises and materials that the team might decide to use during the Retrospective.
- The **outputs** of the Sprint Retrospective include a set of concrete **improvement actions** that the team has agreed to perform in the next Sprint.
- Retrospectives are basically a tool for **Continuous Improvement** (**PDCA**).
- While many Sprint Retrospective approaches exist, most seek to answer the following questions:
  - “What worked well this Sprint that we want to continue doing?”
  - “What didn’t work well this Sprint that we should stop doing?”
  - “What should we start doing or improve?”

# Sprint Retrospective agenda

- Set the stage – Establish the objectives of the meeting up front and create a comfortable atmosphere
- Gather data – Draw on everyone’s experience and perspective to create a **shared picture** of what happened in the Sprint
- Generate insights – Consider the data to identify strengths and issues from the previous Sprint and create a list of **potential action items**
- Determine actions – Pick the issues and challenges from your list of potential action items and create a **concrete plan** of how to achieve success for each one
- Close the Sprint Retrospective – Clarify and summarize the meeting and create a **follow-up plan** for the action items the team agreed on

# Sprint Retrospective activities

- Set the stage
  - Check-In
  - Focus On / Focus Off
- Gather data
  - Event timeline
  - Emotion seismograph
  - Mad/Sad/Glad
- Generate insights
  - Brainstorming
  - Five Whys
  - Fishbone
- Determine actions
  - Set SMART Goals
  - Starfish (Stop/Start/Keep/More/Less)
  - Hot-air balloon
  - Dot Voting
- Close the Sprint Retrospective
  - Appreciations
  - Feedback on Retrospective

# Sprint Retrospective questions

- “What did we do well? What didn’t go so well?”
- “Is our quality getting better?”
- “Do we feel like a team?”
- “Are we living the Scrum values?”
- “Should our Definition of Done become more strict?”
- “What’s slowing us down, both technically and organizationally?”
- “How are the relationships both inside the team and within the organization?”
- “Are we proud of the work we’re doing?”
- “Does anyone have an “appreciation” to share with another team member?”
- “Are we still confused or unclear on any of the items we discussed today?”

# Backlog Refinement

- Backlog Refinement refers to a set of four activities
  - Creating PBIs
  - Refining PBIs
  - Estimating PBIs
  - Prioritizing PBIs
- Refining the Product Backlog is an **ongoing collaborative effort** led by the Product Owner and including significant participation from internal and external stakeholders as well as the Scrum Master and Developers.
- Stakeholders should allocate a sufficient amount of time to Backlog Refinement (up to 10% of its time each Sprint).
- Backlog Refinement might take place e.g. up front, in a workshop during the Sprint, after the Daily Scrum or during Sprint Review.

# Backlog Refinement questions

- “What items are no longer relevant?”
- “What items need to be split?”
- “What items can be updated with new information?”
- “Does this update change previous estimations?”
- “Has the priority of specific items changed?”
- “Do we have any new topics or learnings that haven’t yet been considered?”

# Lengths of timeboxes

- Sprint – one month or less
- Sprint Planning – a maximum of eight hours for a one-month Sprint, usually shorter for shorter Sprint lengths
- Daily Scrum – a maximum of 15 minutes regardless of Sprint length
- Sprint Review – a maximum of four hours for a one-month Sprint, usually less for shorter Sprint lengths
- Sprint Retrospective – a maximum of three hours for a one-month Sprint, usually less for shorter Sprint lengths

## Exercise
## What are the benefits of timeboxes?

## Exercise
## What are the benefits of timeboxes?

- Establishes a WIP limit
- Forces prioritization
- Demonstrates progress
- Avoids unnecessary perfectionism
- Motivates closure
- Improves predictability

# Exercise
# Scrum but

- Why are the following “Scrum buts” not a good idea?
  - Skipping the Sprint Review
  - Holding the Daily Scrum biweekly
  - Canceling the Sprint Retrospective in favor of getting more work done
  - Ignoring the Scrum event timeboxes

# Scrum

- Agile
- Scrum Introduction
- The Scrum Guide – Scrum Artifacts
- User Stories
- Product Discovery
- Scrum Monitoring

# Scrum Artifacts

- Scrum's artifacts represent **work or value**. They are designed to maximize **transparency** of key information. Thus, everyone inspecting them has the same basis for adaptation.
- Each artifact contains a **commitment** to ensure it provides information that enhances transparency and focus against which progress can be measured:
  - For the Product Backlog it is the Product Goal
  - For the Sprint Backlog it is the Sprint Goal
  - For the Increment it is the Definition of Done
- These commitments exist to reinforce empiricism and the Scrum values for the Scrum Team and their stakeholders.

# Product Backlog

- The Product Backlog is an **emergent**, **ordered list** of what is needed to improve the product. It is the single source of work undertaken by the Scrum Team.
- Product Backlog items that can be done by the Scrum Team within one Sprint are deemed **ready for selection** in a Sprint Planning event. They usually acquire this degree of transparency after refining activities.
- Product Backlog refinement is the act of **breaking down** and further defining Product Backlog items into smaller more precise items. This is an ongoing activity to add details, such as a description, order and size. Attributes often vary with the domain of work.
- The Developers who will be doing the work are responsible for the **sizing**. The Product Owner may influence the Developers by helping them understand and select trade-offs.

## Product Backlog
## Commitment: Product Goal

- The Product Goal describes a future state of the product which can serve as a **target** for the Scrum Team to plan against. The Product Goal is in the Product Backlog. The rest of the Product Backlog emerges to define “what” will fulfill the Product Goal.
- A product is a vehicle to deliver value. It has a clear boundary, known stakeholders, well-defined users or customers. A product could be a service, a physical product or something more abstract.
- The Product Goal is the **long-term objective** for the Scrum Team. They must fulfill (or abandon) one objective before taking on the next.

# Good Product Backlog characteristics

- A Product Backlog should be
  - (**D**) etailed appropriately
  - (**E**) mergent
  - (**E**) stimated
  - (**P**) rioritized

# Product Backlog items

- Product Backlog items define a short-term, mid-term and long-term future for the product:
  - Short-term items (a.k.a. **stories**) should have the most details. It's good to have two to three Sprints worth of such items that are ready for the Developers to bring into a Sprint. These items should contain enough detail that the Developers feels confident bringing them into a Sprint, but not so much detail that there's little flexibility in the outcome.
  - Mid-term items (a.k.a. **features**) are things that will be accomplished 3-6 months from now. They should have enough detail for the Developers to know the direction that the product is heading.
  - Long-term items (a.k.a. **epics**) can be vague. They should describe what the product might contain in the distant future. These items are too large to be finished within a single Sprint and will need multiple refinement sessions to become actionable.

# Question

The Product Owner of your Scrum Team tends to add ideas of all kinds to the Product Backlog as a reminder to work on them at a later stage. Over time, this has led to over 300 items in various stages.

What are your thoughts on this?

# Sprint Backlog

- The Sprint Backlog is composed of the **Sprint Goal** (why), the set of **Product Backlog items selected** for the Sprint (what), as well as an **actionable plan** for delivering the Increment (how).
- The Sprint Backlog is a plan by and for the Developers. It is a highly visible, real-time picture of the work that the Developers plan to accomplish during the Sprint in order to achieve the Sprint Goal.
- Consequently, the Sprint Backlog is **updated** throughout the Sprint as more is learned. It should have enough detail that they can inspect their progress in the Daily Scrum.

# Sprint Backlog
# Commitment: Sprint Goal

- The Sprint Goal is the single **objective for the Sprint**. Although the Sprint Goal is a commitment by the Developers, it provides **flexibility** in terms of the exact work needed to achieve it. The Sprint Goal also creates **coherence and focus**, encouraging the Scrum Team to work together rather than on separate initiatives.
- The Sprint Goal is created during the Sprint Planning event and then added to the Sprint Backlog. As the Developers work during the Sprint, they keep the Sprint Goal in mind. If the work turns out to be different than they expected, they collaborate with the Product Owner to **negotiate the scope** of the Sprint Backlog within the Sprint without affecting the Sprint Goal.

# Sprint Goal Advantages

- The Sprint Goal **motivates** the whole Scrum Team. It makes the purpose of the work in the Sprint.
- The Sprint Goal generates some **flexibility** for the Sprint Backlog. If during the Sprint we find a better way to achieve the Sprint Goal, Developers and the Product Owner can discuss whether the Sprint Backlog should be adjusted.
- You can reach the Sprint Goal even if you did not implement 100% of the Sprint Backlog (often not all the features in the Sprint Backlog are necessary to reach the Sprint goal).
- You can **organize** the demo in the **Sprint Review** around the Sprint Goal and usually get more valuable feedback.

# Increment

- An Increment is a concrete stepping stone toward the Product Goal. Each Increment is **additive** to all prior Increments and thoroughly verified, ensuring that all Increments work together. In order to provide value, the Increment must be **usable**.
- Multiple Increments may be created within a Sprint. The sum of the Increments is presented at the Sprint Review thus supporting empiricism. However, an Increment may be delivered to stakeholders **prior** to the end of the Sprint. The Sprint Review should never be considered a gate to releasing value.
- Work cannot be considered part of an Increment unless it meets the **Definition of Done**.

## Increment
## Commitment: Definition of Done

- The Definition of Done is a formal description of the state of the Increment when it meets the **quality** measures required for the product.
- The moment a Product Backlog item meets the Definition of Done, an Increment is born.
- The Definition of Done creates transparency by providing everyone a **shared understanding** of what work was completed as part of the Increment. If a Product Backlog item does not meet the Definition of Done, it cannot be released or even presented at the Sprint Review. Instead, it returns to the Product Backlog for future consideration.

## Increment
## Commitment: Definition of Done

- If the Definition of Done for an Increment is part of the standards of the organization, all Scrum Teams must follow it as a minimum. If it is not an **organizational standard**, the Scrum Team must create a Definition of Done appropriate for the product.
- The Developers are required to conform to the Definition of Done. If there are **multiple Scrum Teams** working together on a product, they must mutually define and comply with the same Definition of Done.

# Definition of Done Example

## Definition of Done Example

- All code has been **developed**.
- All code meets the team's **coding standards**.
- Code has been checked into the source control **repository**.
- Unit **tests** are written and passing.
- The **continuous integration** build is passing.
- **Acceptance criteria** have been met and confirmed by a Developer who didn't write the code.
- The Product Owner has **accepted** the Product Increment.
- All code has been **deployed** to the staging environment.

# Scrum Alliance Done Thinking Grid

| | | | | |
|---|---|---|---|---|
| User Story Clarity | Environment Ready | Code Complete | Automated Code Review | Functional Testing |
| Task Identified | Design Complete | Unit Tests Executed | Peer Review | Regression Testing |
| Build Setup Changes | Unit Test Cases Written | Refactoring | Code Coverage | Performance Testing |
| Product Owner Approval | Documentation | Code Checkin | Burndown Chart Ready | Acceptance Testing |
| Product Backlog Updated | Pre-Release Builds | Code Merging and Tagging | Release Build | Closure |

# Agile software development practices

- Automated unit testing
- Continuous integration (CI)
- Pair programming
- Test-driven development (TDD)
- Acceptance test-driven development (ATDD)
- Refactoring

# Exercise Responsibilities

- Sets the Sprint target
- Finds a suitable Definition of Done
- Estimates the entries in the Product Backlog
- Decides how much will be implemented in the Sprint
- Writes user stories
- Removes obstacles
- Ensures that an adequate Product Backlog is in place
- Reports to management
- Decides whether the product can be delivered
- Keeps the Scrum board up to date
- Changes tasks in the Sprint
- Increases productivity
- Splits user stories which are too big

# Scrum

- Agile
- Scrum Introduction
- The Scrum Guide
- User Stories
- Product Discovery
- Scrum Monitoring

# Requirements in Scrum

- Scrum and sequential product development treat requirements very differently. With sequential product development, requirements are nonnegotiable, **detailed up front** and meant to stand alone. In Scrum, the details of a requirement are **negotiated** through **conversations** that happen continuously during development and are fleshed out **just in time** and just enough for the teams to start building functionality to support that requirement.
- When developing innovative products, **you can’t create complete requirements** or designs up front by simply working longer and harder. Some requirements and design will always **emerge** once product development is under way.
- Thus, when using Scrum, we don’t invest a great deal of time and money in working out the details of a requirement up front. Because we expect the specifics to **change** as time passes and as we learn more about what we are building, we **avoid overinvesting** in requirements that we might later discard.

# Requirements in Scrum

- Instead of compiling a large inventory of detailed requirements up front, in Scrum we create **placeholders** for the requirements, called Product Backlog items (PBIs). Each PBI represents desirable **business value**.
- Initially the PBIs are large and there is very little detail associated with them. Over time, these PBIs flow through a series of conversations among the stakeholders, the Product Owner and the Developers, getting **refined** into smaller, more detailed PBIs. Eventually a PBI is small and **detailed enough** to move into a Sprint. Even during the Sprint, however, more details will be exposed in conversations between the Scrum Team.
- In contrast to Scrum, with sequential product development all requirements must be at the **same level of detail** at the same time. In particular, the approved requirements document must specify each and every requirement so that the teams doing the design, build and test work can understand how to conform to the specifications. There are no details left to be added.

# Requirements in Scrum

- Forcing all requirements to the same level of detail at the same time has many disadvantages:

# Requirements in Scrum

- Forcing all requirements to the same level of detail at the same time has many disadvantages:
  - We must predict all of these details early during product development when we have the **least knowledge**.
  - We treat all requirements the same regardless of their **priority**, forcing us to dedicate valuable resources today to create details for requirements that we may never build.
  - We create a large inventory of requirements that will likely be very **expensive to rework** or discard when things change.
  - We reduce the likelihood of using conversations to elaborate on and clarify requirements because the requirements are already “complete”.

# What are user stories?

- User stories are a **convenient format for expressing the desired business value** for many types of Product Backlog items, especially features. User stories are crafted in a way that makes them understandable to both **business people** and **technical people**.
- They are structurally simple and provide a great **placeholder for a conversation**. Additionally, they can be written at various levels of **granularity** and are easy to progressively refine.
- User stories are not the only way to represent Product Backlog items. They are simply a **lightweight approach** that corresponds nicely with core agile principles.

# Why user stories?

- User stories emphasize **verbal** rather than written communication.
- User stories are comprehensible by both the Product Owner and the Developers.
- User stories are the **right size** for planning.
- User stories work for **iterative** development.
- User stories encourage **deferring detail** until you have the best understanding you are going to have about what you really need.
- User stories enhance the **tacit knowledge** of the team.

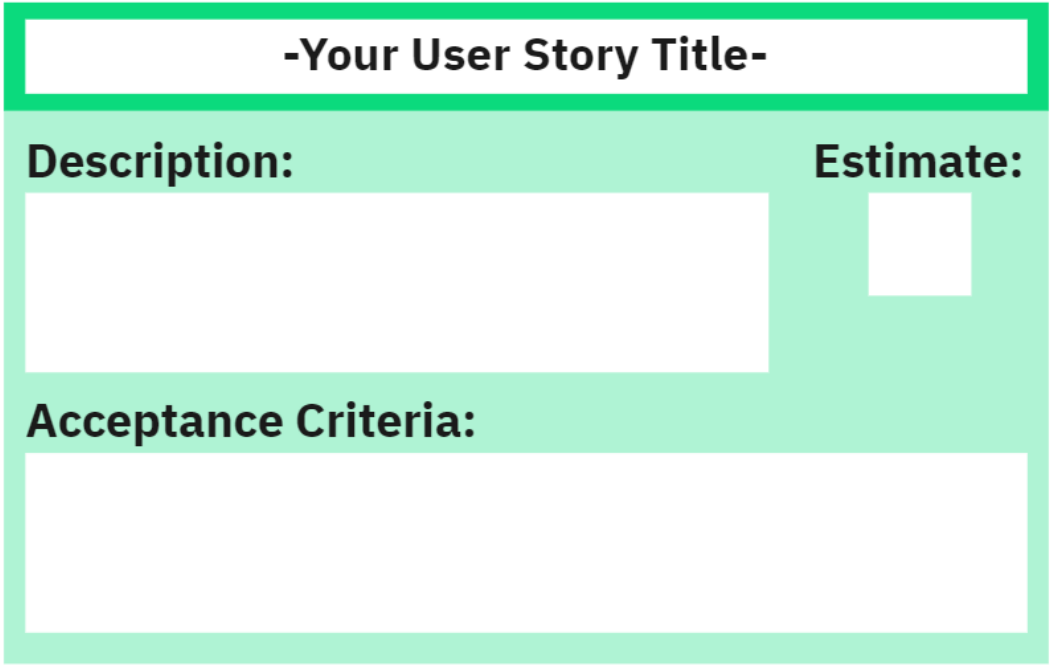
## Potential drawbacks

- On large projects it can be difficult to keep hundreds or thousands of user stories **organized**.
- User stories may need to be augmented with **additional documents**.
- Conversations do not **scale** adequately to entirely replace written documents on large projects.

# CCC of user stories

- Card
  - A written description of the story used for planning and as a reminder
- Conversation
  - Conversations about the story to work out the details
- Confirmation
  - Tests (acceptance criteria) that convey and document details and that can be used to determine when a story is complete

## Card


-Your User Story Title-
Description:
Estimate:
Acceptance Criteria:

# User story description Example

As a vegetarian pizza buyer,

I want to see only the vegetarian toppings

so that I can order a pizza without getting distracted by toppings I won’t choose.

Template:

As a (who wants to accomplish something -> **role**)

I want to (what they want to accomplish -> **feature**)

so that (why they want to accomplish that thing -> **business value**)

# INVEST criteria for user stories

- **Independent** – The story can stand alone from the other user stories. Independent stories can be prioritized in any order.
- **Negotiable** – There might be some tradeoffs and negotiation. Negotiation can be based on the priority of the user story, risks involved, value to the organization and how the user story can affect other features of the product.
- **Valuable** – The result creates business value. User stories must create business value when the team creates the feature in the product.

# INVEST criteria for user stories

- **Estimable** – You can create an estimate of effort based on the user story. The team will examine the user stories and determine an estimate of effort to create the feature.
- **Small** – The stories aren’t too big and don’t need further decomposition. User stories are usually between one and three days of work.
- **Testable** – The team can write tests for the product to confirm the successful inclusion of the user story. Test-driven development is based on user stories.

# Writing good user stories

- Keep user stories **short and simple** so that everyone can understand them.
- Write stories from the **perspective of the users** (Product Owners, Scrum Masters and Developers are [almost] never the users of the product they build).
- Try to identify **different kinds of users** (instead of starting each user story with „as a user…“).
- Avoid writing **technical** user stories.
- Describe **only one piece of functionality** in one user story. If a functionality is too big to be implemented in one Sprint, split it up.
- Do not leave out the **business value** of a user story („so that …“).
- Do not forget the **acceptance criteria** (later!).
- Stick to the **INVEST** criteria.

# Exercise
# User stories

You want to create a fan website for your favorite sports team.

Write at least 3 user stories describing some functionality you consider important for your website (e.g. news, results, forum, ticketing or fan shop).

Use the following template:

As a (who wants to accomplish something)

I want to (what they want to accomplish)

so that (why they want to accomplish that thing)

# Conversation

- The details of a requirement are exposed and communicated in a conversation among the Developers, Product Owner and stakeholders. The user story is simply a **promise to have that conversation**.
- It is usually not just one conversation, but rather an **ongoing dialogue**. There can be an initial conversation when the user story is written, another conversation when it's refined, yet another when it's estimated, another during Sprint Planning and finally, ongoing conversations while the user story is being designed, built and tested during the Sprint.
- One of the benefits of user stories is that they shift some of the focus away from writing and onto conversations. These conversations enable a **richer form of exchanging information** and collaborating to ensure that the correct requirements are expressed and understood by everyone.

# Different people, different conversations

- Product Owner
  - “How will this benefit our customers?”
- Project manager
  - “What are dependencies, time estimates, risks and status?”
- Tester
  - “What are edge-cases and likely points of failure?”
- Business analyst
  - “What are the behavior details and business rules?”
- UX designer
  - “Who uses this and what are their goals?”
- Developer
  - “How does the software behave?”

# Confirmation

- A user story also contains confirmation information in the form of **conditions of satisfaction**. These are **acceptance criteria** that clarify the desired behavior.
- They are used by the Developers to better understand what to build and test and by the Product Owner to **confirm** that the user story has been implemented to his satisfaction.
- If the front of the card has a description of the story, the **back of the card** could specify the conditions of satisfaction.
- These conditions of satisfaction can be regarded as **high-level acceptance tests**. They are an important way to capture and communicate, from the Product Owner’s perspective, how to determine if the story has been implemented correctly.

# Acceptance criteria

- Acceptance criteria or ‘conditions of satisfaction’ provide a detailed scope of a user’s requirements. They help the team to understand the **value of the story** and set expectations as to when a team should consider something done.
- Acceptance criteria are usually scenario-oriented (**Given/When/Then template**) or rule-oriented (**checklist**).
- Acceptance criteria goals
  - clarify what the team should build before they start work
  - ensure everyone has a common understanding of the problem
  - help the team members know when the story is complete
  - help verify the story via automated tests

from: easyagile.com

# Acceptance criteria

- Acceptance criteria can include
  - Functional and non-functional use cases
  - Negative scenarios of the functionality
  - Performance concerns and guidelines
  - UX concerns

# Acceptance criteria
# Given/When/Then template

As a writer, I want to receive notifications when others add comments
so that I am up-to-date.

Scenario-based acceptance criteria for the above user story:

1) **Given** I don't have the app open and my phone is locked, **when** a comment is added, **then** I should receive a banner notification.
2) **Given** I have the app open and am editing a document, **when** a comment is added to this document, **then** the bell icon should be updated to show unread notifications including their number.
3) **Given** a user was mentioned in a comment (using @), **when** the mentioned user opens the commented document, **then** a pop-up message should appear with a link to the new comment.

# Acceptance criteria Checklist

As a conference attendee, I want to be able to register online, so that registration is simple and paperless.

Rule-oriented acceptance criteria for the above user story:

1) User cannot submit a form without filling out all of the mandatory fields
2) Information from the form is stored in the registrations database
3) Protection against spam is working
4) Payment can be made via Paypal, Debit and Credit Card
5) An acknowledgment email is sent to the attendee after submitting the form

# Exercise
# Acceptance criteria

You want to create a fan website for your favorite sports team.

For each of the user stories you created, write some acceptance criteria using either a scenario-oriented (**Given/When/Then template**) or rule-oriented (**checklist**) approach.

# Handling nonfunctional requirements

- **Nonfunctional requirements** represent system-level constraints.
- They are usually handled in one of the following ways:
  - Written as **user stories**
  - Included in the **Definition of Done**
  - Written as **acceptance criteria** (confirmation)

# Kinds of Product Backlog items

## Kinds of Product Backlog items

- New features
- Changes
- Bugs
- Nonfunctional requirements
- Technical improvements
- Knowledge acquisition (prototypes, spikes)

# Splitting user stories

- When user stories are too big to be implemented within one Sprint, we need to further break them down into smaller stories
- There are some patterns available for splitting stories:
  - Workflow Steps
  - Business Rule Variations
  - Major Effort
  - Simple/Complex
  - Variations in Data
  - Data Entry Methods
  - Defer Performance
  - Operations (e.g. CRUD)
  - Break Out a Spike

# Exercise
# Splitting user stories

Split the following user story for a job portal:

As a user I want to be able to search for a job so that I can find a new employer.

# Criteria for prioritizing Product Backlog items

# Criteria for prioritizing Product Backlog items

- **Risk** that the story cannot be completed as desired
- **Impact** that the story will have on other stories if deferred
- Desirability of the story to a **broad base of users** or customers
- Desirability of the story to a small number of **important users** or customers
- **Cohesiveness** of the story in relation to other stories
- **Cost** to implement the story
- **Cost of delay** (business value, time criticality, risk reduction)

# Exercise
# Prioritization Game

# Estimating user stories

- To determine how many user stories, you can finish you need to determine the **effort** you‘ll spend on completing them.
- Estimates are often wrong but estimating (the **process** of coming up with the estimates) can still prove useful.
- Estimating gives you a chance to start **managing the uncertainty** that lies ahead. It can be used as a risk-management tool that allows you to discover misconceptions, inconsistencies and areas that need further investigation early on.
- Important estimating concepts
  - Estimate as a team
  - Estimates are not commitments
  - Focus on accuracy, not precision
  - Use relative sizes (like story points) instead of absolute sizes (like hours)

# Advantages of using story points

- Story points have many advantages compared to estimates in person days
  - The estimation is **decoupled** from the development speed of **individual** developers (with estimates in person days, one developer could estimate 1 day and another and another developer 5 days, and both could be right – depending on who ultimately implements the requirement).
  - It's easier to arrive at a team estimate.
  - The **Fibonacci sequence** expresses that with larger entries, we become less accurate in the estimates. This allows the team to arrive at an estimate more quickly.
  - The estimates remain **stable** even if the development speed of the team changes (due to changes to the team composition, improvement measures, etc.).

# Relative vs absolute sizes Example

| Country | Relative size | Absolute size in km$^2$ |
|---|---|---|
| Germany | 3 | 357,000 |
| France | 5 | 644,000 |
| England | 2 | 130,000 |
| Austria | 1 | 84,000 |
| Denmark | 0.5 | 43,000 |
| Denmark (incl. Greenland) | 20 | 2,240,000 |
| Switzerland | 0.5 | 41,000 |
| United States | 100 | 9,826,000 |

# Planning Poker

- In order to play Planning Poker, you need a deck of **numbered cards** (usually showing the Fibonacci sequence : 0, 1, 2, 3, 5, 8, 13, …) representing the estimated story points.
- The Product Owner **presents** the user story to be estimated.
- The Scrum Team asks **questions** about the user story and the Product Owner explains it in more detail.
- Each Developer **chooses a card** for themselves that they think corresponds to the effort of the story.
- All chosen cards are revealed at the **same time**.
- The participants with the **lowest** and **highest** estimates explain their rationale.
- The process is repeated until a **consensus** is reached.
- The game is repeated until all user stories are estimated.

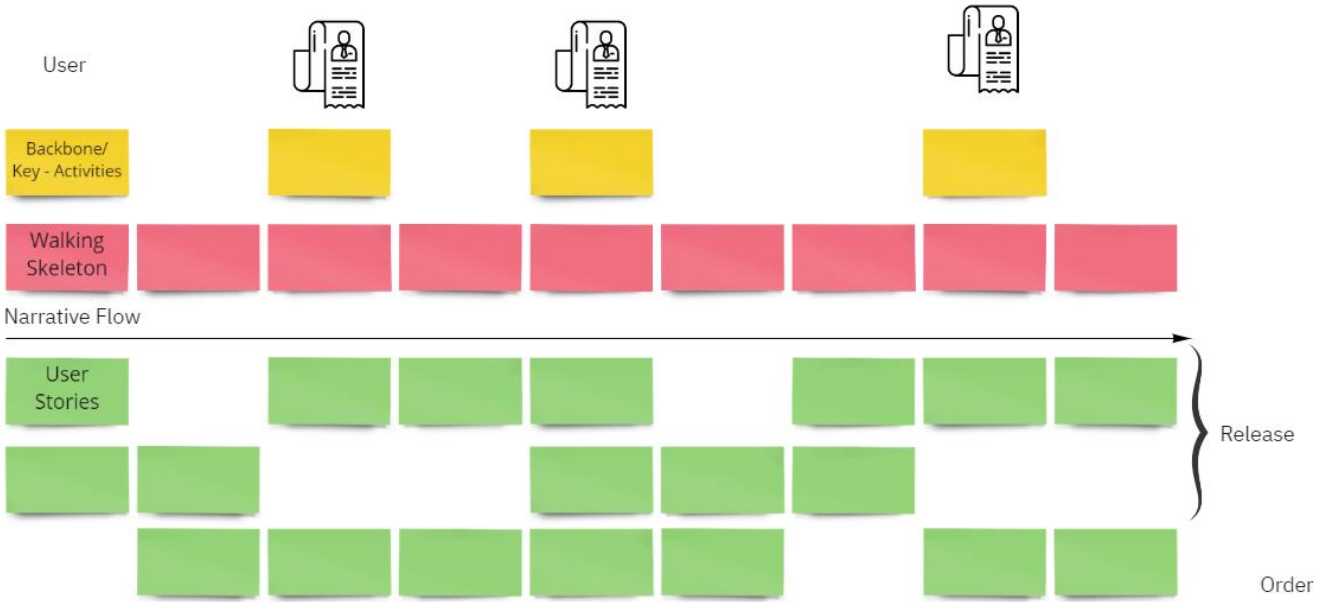
# Exercise
# Planning Poker

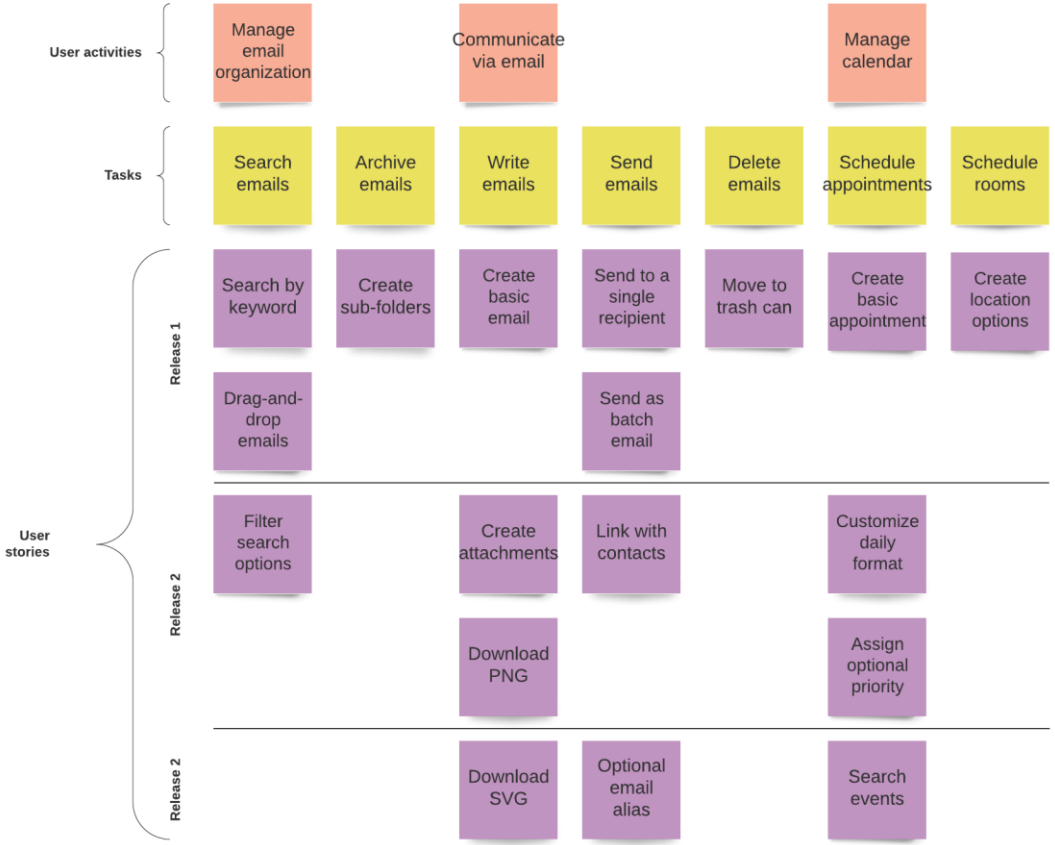
- Imagine you are planning a birthday party for yourself. Here are some tasks you came up with:
  - Choose a date, time & place
  - Reserve location
  - Send invitations
  - Plan entertainment
  - Create a playlist
  - Buy beverages and food
  - Bake a cake
  - Arrange tables and chairs
  - Clean the location after the party
- Play a round of Planning Poker to determine the effort for each of the tasks!

# User story mapping

- Having a large collection of user stories complicates understanding the **big picture**.
- User story mapping is a method to arrange user stories in a visual way to create a more **holistic view** of the product.
- It improves the teams understanding of their customers and helps to **prioritize** work.
- It provides an **alternative** to building a flat list of Product Backlog items.
- It is organized to tell the story of the **customer journey** and break it into parts.

# User story mapping Example






from: https://lucidspark.com/de/blog/user-story-mapping-von-a-bis-z

# Steps to create a user story map

- Find the **most important user**
- Find key activities as the **backbone**
- Break down key activities into epics to form the **walking skeleton**
- **Epics** are further broken down into **user stories**
- Discover additional key activities
- Enhance the map with additional users
- Explore the story map
  - Fill in and refine the story map
  - Collect feedback
  - Group it by releases (start with an **MVP**)

# Exercise
# User story mapping

- Create a story map from “waking up” to “start working at the office”!
- Write out your stories one step at a time.
- Organize your stories in a narrative flow.
- Fill in missing details.
- Explore alternative stories.
- Aggregate your stories to make a backbone of activities.
- Slice out tasks that help you reach a specific outcome.
  - Waking up too late
  - Doing home office

# MVP

- An MVP (Minimum Viable Product; term from Lean Startup) is the first minimally functional iteration of a product that serves to learn from **user feedback** as quickly as possible.
- The goal of this strategy is to avoid products that customers don't even want.
- Purposes
  - Testing a market niche (a **hypothesis**) with as little development effort as possible
  - Focusing the developers on the **essential** product components
  - Accelerated learning
  - Earliest possible provision of a product to users (**early adopters**)
  - Proof of **competence** of the software developers

# MVP

# Scrum

- Agile
- Scrum Introduction
- The Scrum Guide
- User Stories
- Product Discovery
- Scrum Monitoring

# Product Discovery

All you need is a product vision and enough top priority items on the backlog to begin one iteration, or Sprint, of incremental development on the product.

Ken Schwaber, Mike Beedle

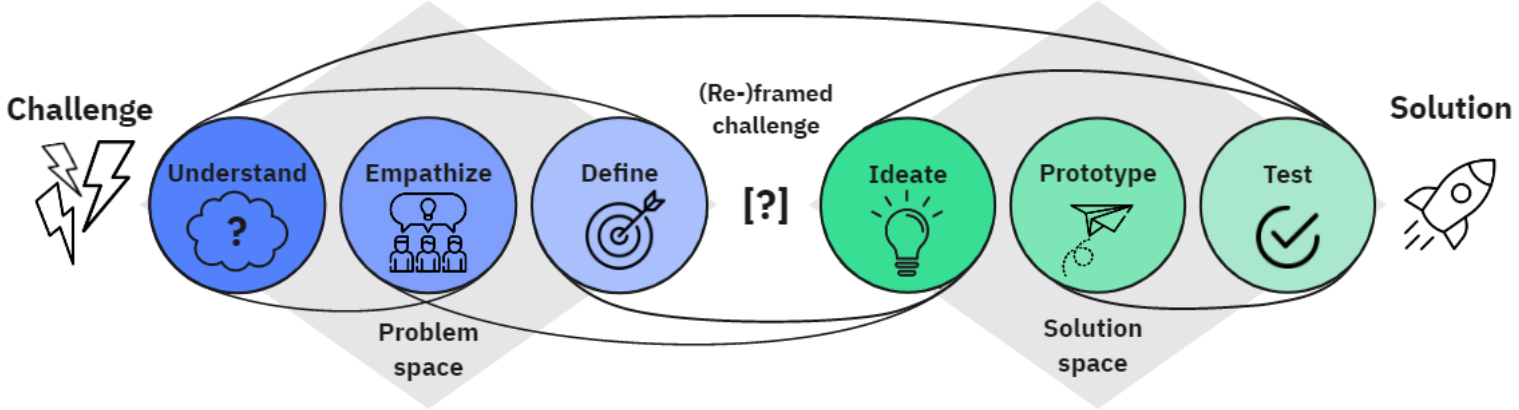
# Product Discovery in Scrum

- Scrum doesn’t elaborate on the process that enables a Product Owner to add valuable, usable and feasible user stories to the Product Backlog or how to come up with a **product idea**.
- Frameworks and methodologies suitable for the **product discovery** process include
  - Design Thinking
  - Design Sprints
  - Lean Startup
  - Lean UX
  - A/B testing
  - Business Model Canvas

# What is Design Thinking?

- A human-based approach to innovation that aims to establish creative ideas and effective business models by focusing on the **needs of people**
- The basic idea is to apply the approaches and methods of **designers** to the development of **innovations** (this is what the word *design* stands for) while also engaging in a systematic, fact-based **analysis** of the feasibility and economic viability of these innovations (this is what the *thinking* part of the term stands for).
- Designers start with their **customers' problems or wishes** in mind and consider them from the perspective of their **target users**. With this knowledge, designers develop the first user-oriented ideas, visualize their creative solutions at an early stage and then design **prototypes**.
- The approach relies on quickly getting **customers feedback** and change the concept on this basis. By doing this, the designers approach the best solution for their target users.

# Design Thinking process



# Design Thinking phases

- **Understanding the problem**: In the first phase, you create an in-depth understanding of your target users' problem or need. You have to clarify which information you're still lacking about the target users, their needs and their problems.
- **Observing customers**: This phase consists of detailed research and on-site observations about the customer's need or problem. It utilizes observations and surveys so that you can put yourself in the customer's shoes.
- **Defining the point of view**: After the observations and surveys, you should focus the insights on a selected group of customers or users and summarize their problems and needs in a defined question.

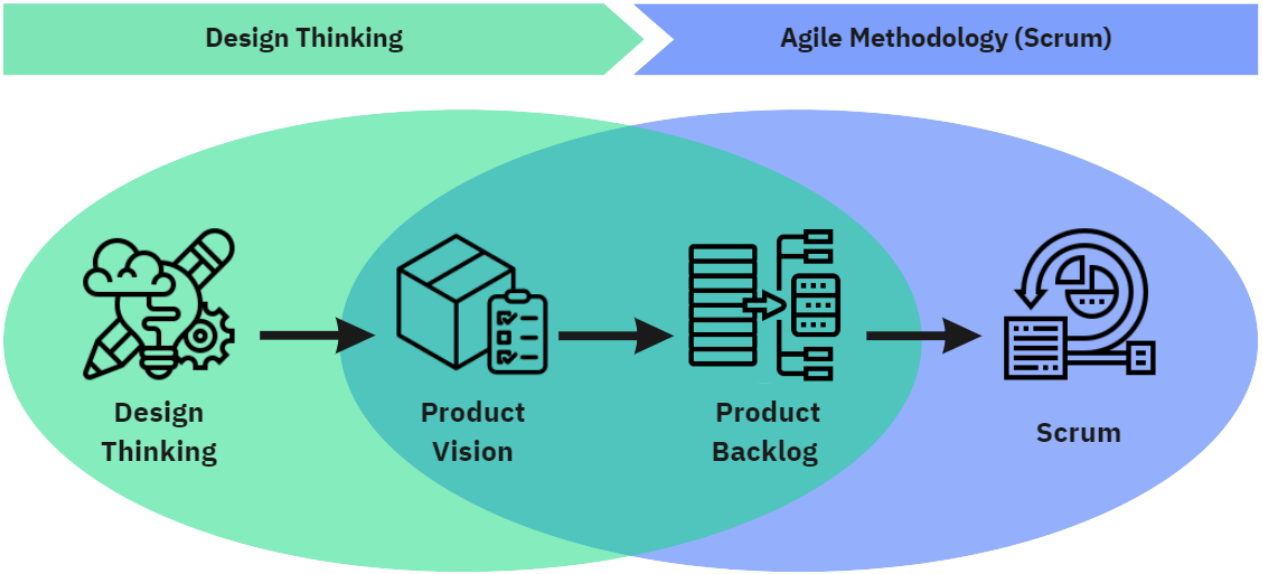
# Design Thinking phases

- **Finding and selecting ideas**: Only in this phase do you actually find ideas. You need to employ creative principles and techniques so that you prepare multiple possible solutions. Evaluate the usefulness, economic viability and feasibility of your ideas and make a selection.
- **Developing prototypes**: In this phase, you should visualize the ideas, make them tangible and then outline, design, model or simulate them so that the potential customer understands your idea and can test it.
- **Testing assumptions**: In this concluding phase, you test your assumptions or ideas with systematic customer feedback. You receive responses, learn from them and continue developing your idea.

# Design Thinking method catalogue

- Understand
  - 5 Whys
  - Brainwriting
  - Emotional Journey Map
  - Six Ws
- Empathize
  - Interviews
  - 5 Whys
- Define Point of View
  - Empathy Map
  - Personas
- Ideate
  - Brainstorming
  - 6-3-5 Method
  - Mindmapping
  - How-Now-Wow Matrix
- Prototype
  - Paper Prototype
  - Mockup
  - Role Play
- Test
  - Presentation
  - User Tests

# From Design Thinking to Scrum



# From Design Thinking to Scrum

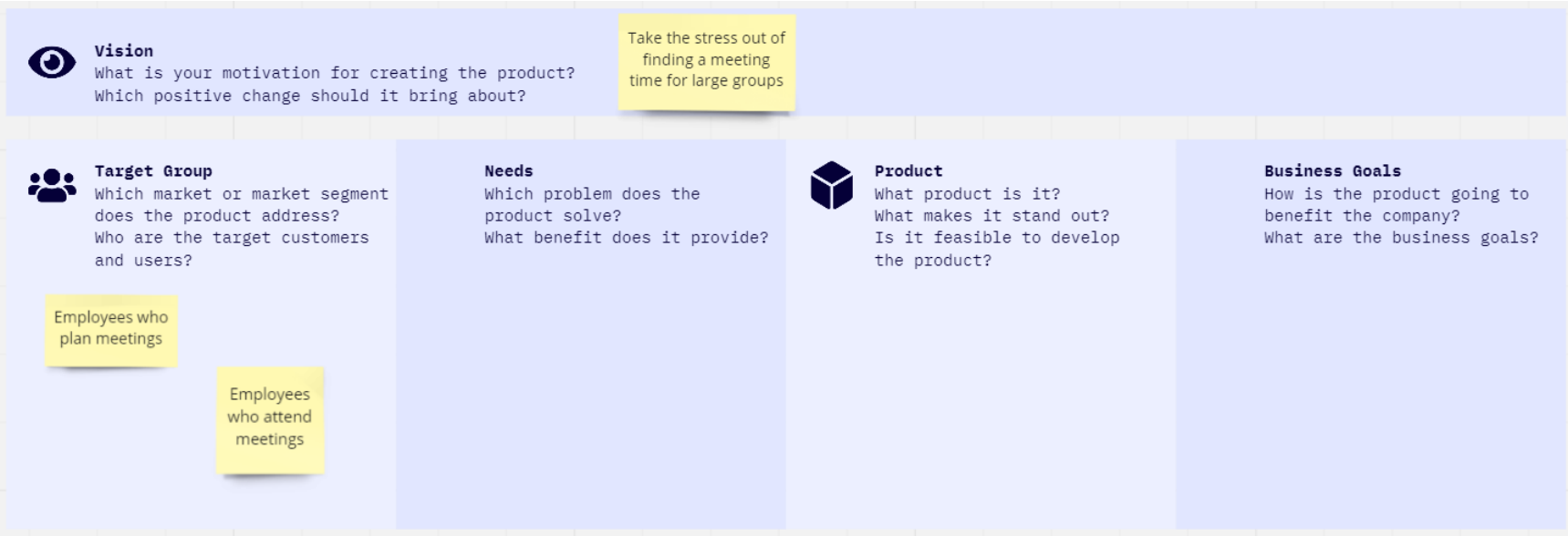
- The **output** of Design Thinking can be used as an **input** for the Scrum framework.
- After applying Design Thinking we are able to formulate a **Product Vision** (the Product Vision provides a clear description of the areas in which the stakeholders, such as users and customers, get value).
- This is followed by generating the initial Product Backlog and frequently the definition of a **Product Roadmap** (a high-level, strategic plan, that describes the initial idea for development of the product over the next period of time and which helps keeping the stakeholders aligned).

# Popular Product Vision formats

- Capturing a Product Vision as
  - Product box
  - Newspaper article
  - Magazine review
  - Press release
  - Conference slides
- Product Vision Board
- Business Model Canvas
- Lean Canvas

- Mission statement
  - For (target audience)
  - who (description of the need or opportunity)
  - (product name) is a (product category)
  - the (main advantage, reason to buy this product)
  - unlike (alternative of the competition)
  - our product (description of the main difference)

# Product Vision Board




from: Miro

# Scrum

- Agile
- Scrum Introduction
- The Scrum Guide
- User Stories
- Product Discovery
- Scrum Monitoring

# Monitoring

- Knowing the **current status** of your project is absolutely necessary for every Scrum Team and for the management.
- The principle of "**Inspect and Adapt**" only works if each individual in a Scrum Team is able to correctly assess the current situation.
- In Scrum, we use a series of **charts** for this kind of transparency and monitoring.
- The **individual performance** counts a lot in Scrum, because without the creativity, the commitment and the willingness to perform, no high-quality products can be created.
- But when measuring the overall performance, we use metrics concerning the **whole team**, because the team members deliver together. Most of these metrics are either business-related, product-related or team-related.

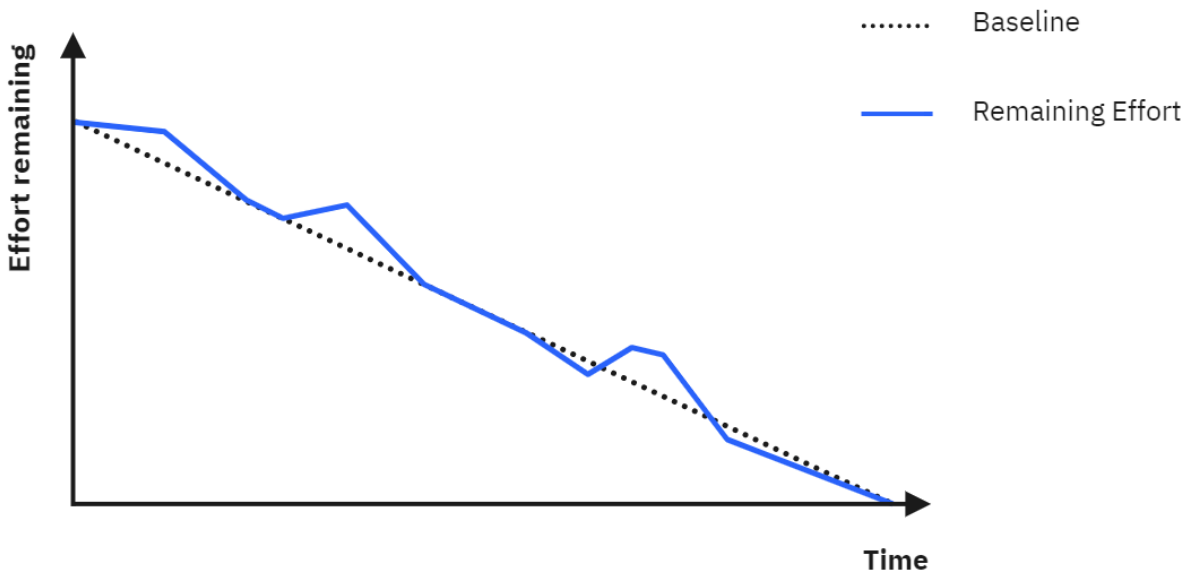
# Scrum metrics

- Business-related
  - Time to market
  - Return of investment
  - Customer satisfaction
- Team-related
  - Team satisfaction
  - Team member turnover
- Product-related
  - Sprint goal success
  - Escaped defects and defect density
  - Development progress
    - Sprint burn down chart
    - Iteration burn down chart
  - Velocity
    - Velocity chart

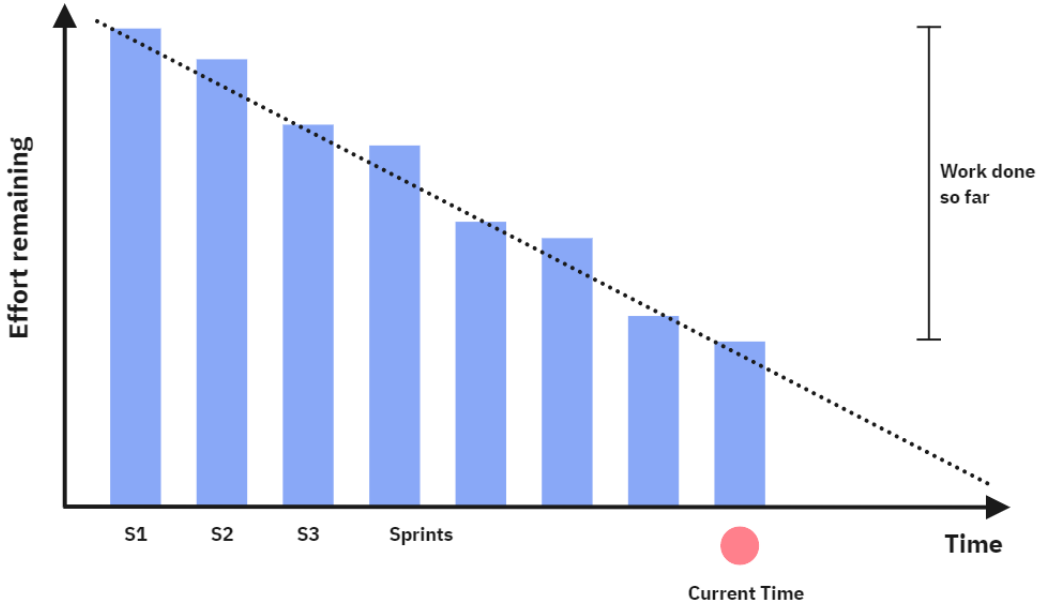
# Scrum charts

- The basic idea of reports in Scrum is that anyone who takes a look at the metrics can **quickly** see the **status of product development** and the performance of the Scrum Team.
- This is achieved with one or more simply constructed graphs, called charts in Scrum. Among the most common are:
  - Sprint burn down chart
  - Iteration burn down chart
  - Velocity chart

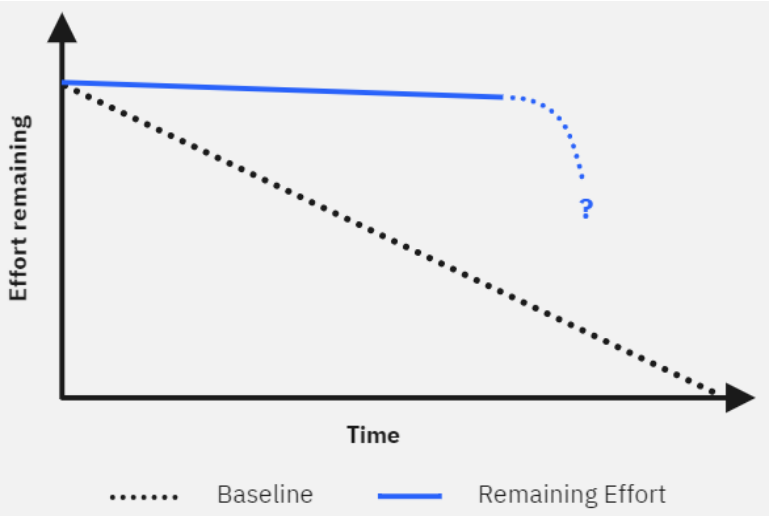
# Sprint burn down chart

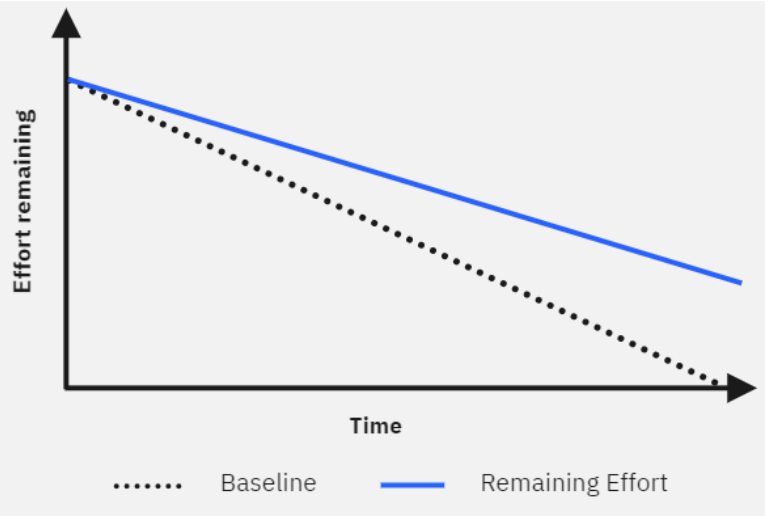


# Iteration burn down chart

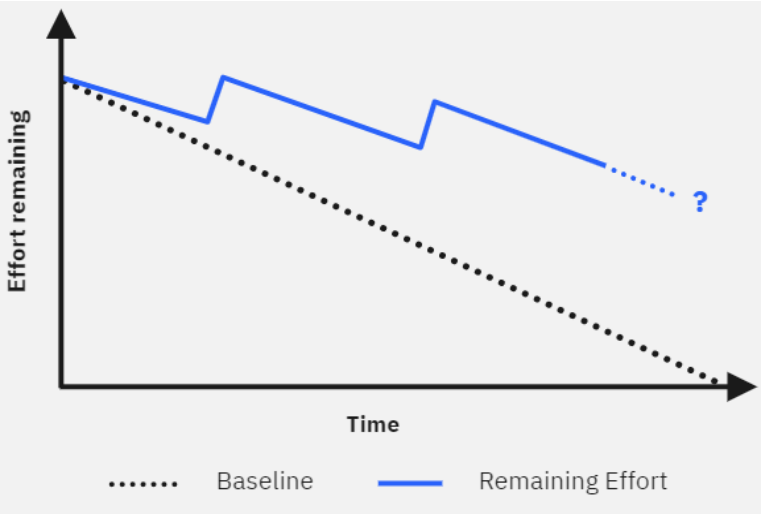


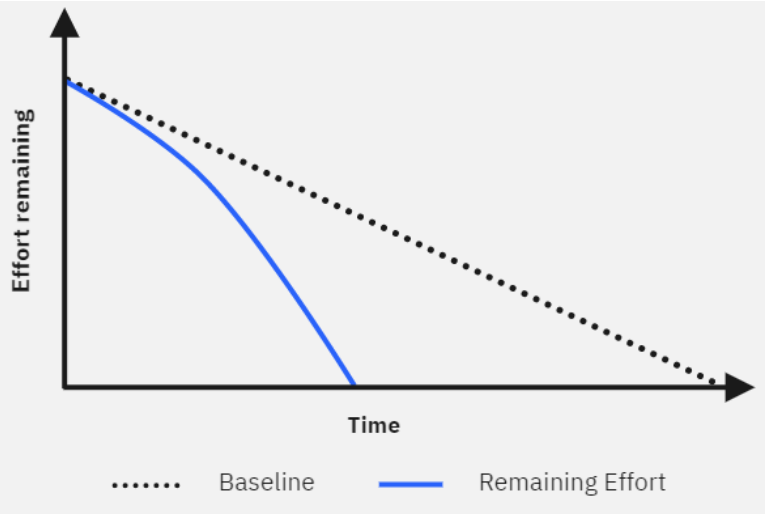
# Exercise
## What might be the reason for these Burn Down Chart patterns?



# Exercise
## What might be the reason for these Burn Down Chart patterns?

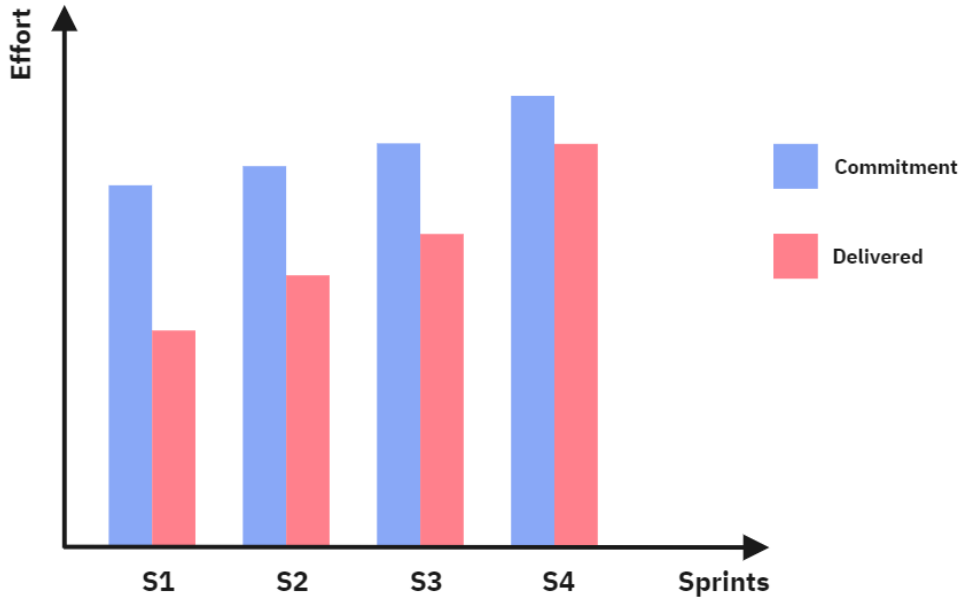


# Velocity

- **Velocity** is the amount of work completed each Sprint. It is measured by adding the sizes of the PBIs that are completed by the end of the Sprint.
- A PBI is either done or it’s not done. The Product Owner doesn’t get any value from undone items, so velocity does not include **partially completed** PBIs.
- Velocity measures **output** (the size of what was delivered), **not outcome** (the value of what was delivered).
- It is an essential concept for Scrum Planning (Release-level Planning and Sprint Planning).
- Velocity is also a **diagnostic metric** that the team can use to evaluate and improve its use of Scrum to deliver customer value. By observing its own velocity over time, the team can gain insight into how specific process changes affect the delivery of measurable customer value.

# Velocity chart



## Exercise
## What might be the reasons for a volatile velocity?

## Exercise
## What might be the reasons for a volatile velocity?

- New team members join; others leave the company.
- Developers are assigned to multiple projects or are addressing unplanned bugs.
- Product Backlog items are either too big or unclear.
- The Product Owner is absent.
- Impediments are piling up.
- Technical debt is slowing down the team.
- The Developers don’t have all skills it needs to deliver a done Increment.
- Team members aren’t working together.
- The team working in uncharted territory.
- The team working with (undocumented) legacy code.
- Product Backlog items are being poorly prepared, thus making them difficult for the team to estimate.

# Scrum

- Questions

# Questions

- What are the main differences between traditional and agile project management?
- Where do architecture and design take place in Scrum?
- Why is it necessary to have a Product Vision?
- What can happen when a Definition of Done is missing?
- What’s meant with the pull principle?
- How do you avoid misallocating resources on features or products, that no one wants?
- Which elements of Scrum contribute to transparency?
- What do you think are the character attributes that make a Scrum Master stand out?