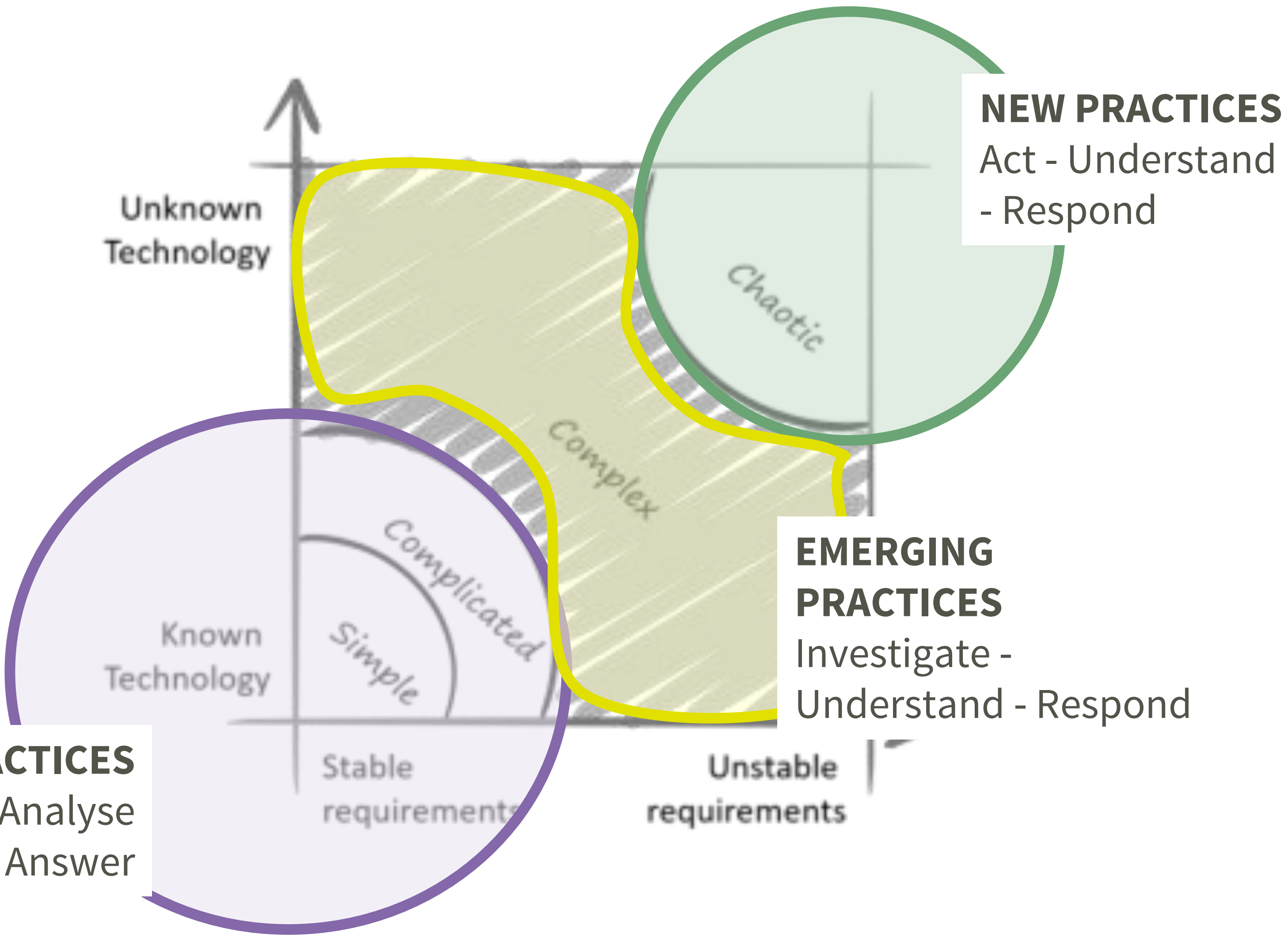
Digital Technology

Agile Project Management

Giulio Nicelli

giulio.nicelli@polimi.it



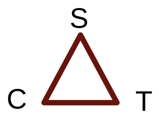
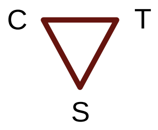
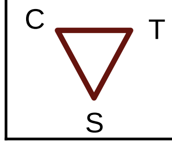
NEW PRACTICES
Act - Understand - Respond

EMERGING PRACTICES
Investigate - Understand - Respond

GOOD PRACTICES
Understand - Analyse or Categorise - Answer

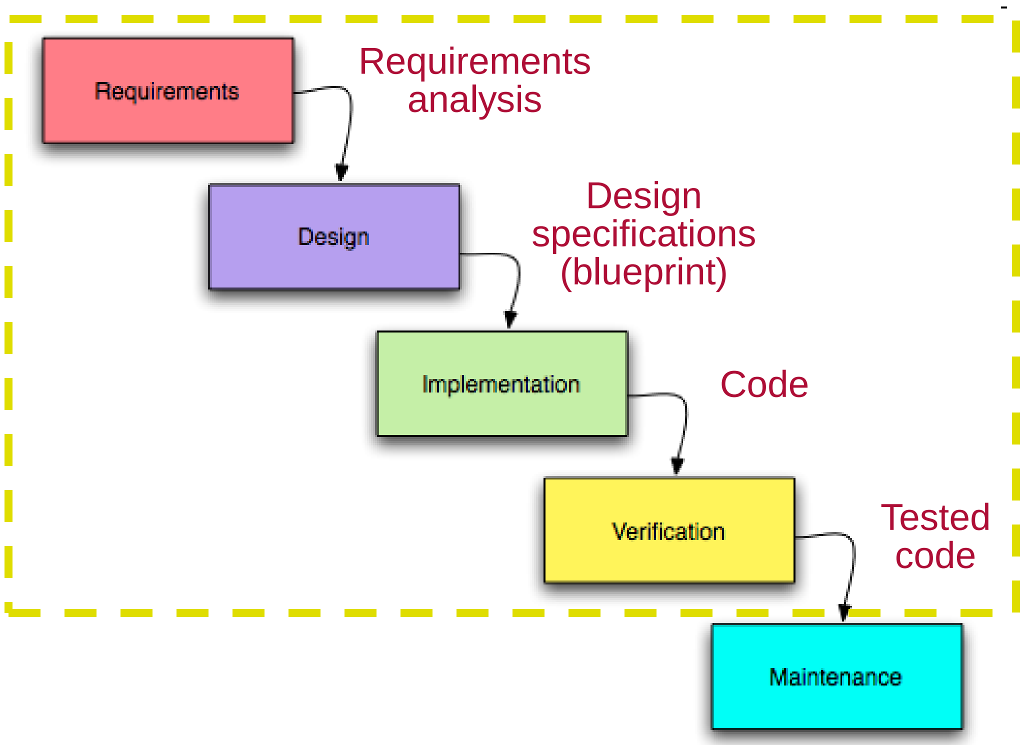
Project Management Types

- **Traditional** (TPM): objectives and solutions are clear and known since the beginning of the project
- **Adaptive/Agile** (APM): clear objectives, but no clear solutions. The approach is iterative / incremental.
- **Extreme** (xpm): Typical of research and development projects, with vague objectives and solutions to develop/understand

		SCOPE	TIME	COSTS
	TPM	FIXED	FIXED	FIXED
	APM	VARIABLE	POSSIBLY FIXED	POSSIBLY FIXED
	XPM	NOT WELL DEFINED	RE-PLANNABLE	RE-FINANCEABLE

Traditional PM in IT Projects: Waterfall methodology

- Founded in 1970 by W. Royce (*Managing the development of large software systems*)
- From manufacturing industry
- Each phase has a specific output (e.g. Blueprint for the development)
- Each phase must be completed to continue with the next step



Waterfall: Core Assumptions & Rationale

- **Up-front effort saves downstream cost**
  - Requirement defects become 50 – 200 % more expensive to fix once the project is past the analysis phase (McConnell 1996)
- **Strict phase-gate progression**
  - Each stage must be *fully* completed, reviewed, and signed-off before the next begins
- **Stable, well-understood requirements**
  - Minimal expected change after the requirements baseline is approved
  - Customer involvement is front-loaded; feedback loops later in the cycle are limited
- **Heavy emphasis on documentation**
  - Detailed specs, design docs, test plans, and user manuals accompany every phase
  - Benefits:
    - Simplifies long-term maintenance and onboarding of new team members
    - Eases compliance, auditing, and contractual reporting
- **Low tolerance for risk and uncertainty**
  - Best suited to projects with mature technology, predictable scope, or safety-critical/regulatory constraints (e.g., aerospace, medical devices, defense)

Waterfall: key advantages

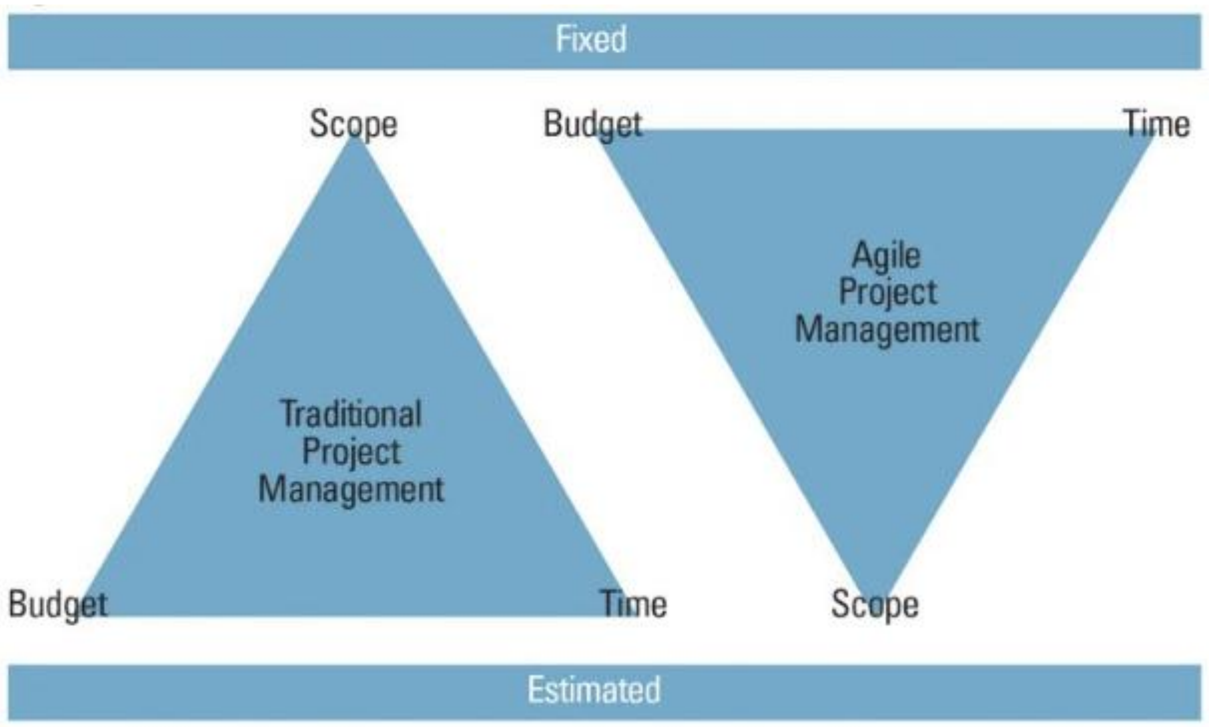
- **Simple, intuitive, and communicable**
  - Linear flow is easy for non-technical stakeholders to grasp and explain
- **Management-friendly structure**
  - Clear division of labour – teams or individuals can be assigned to discrete phases (requirements, design, code, test, etc.)
  - Well-defined milestones & sign-offs aid schedule tracking, budgeting, and contractual reporting
  - Predictable resource planning – staffing levels and skill-sets can be forecast per phase
- **Up-front planning enables accurate projections**
  - Scope, cost, and delivery dates can be estimated with higher confidence when requirements are stable
  - Facilitates fixed-price or compliance-driven contracts
- **Comprehensive documentation by design**
  - Artefacts produced in every stage (SRS, design specs, test plans, user manuals) support maintenance, onboarding, audits, and knowledge retention
- **Disciplined change and risk control**
  - Formal change-request process minimises scope creep
  - Early identification of integration or regulatory risks before significant coding begins

Waterfall: main drawbacks

Waterfall Model – Principal Drawbacks

- **Rigid, phase-gated workflow**
  - Little or no overlap between stages; any discovery forces expensive “back-ups” to earlier phases
- **Late discovery of defects & risks**
  - Coding and integration happen near the end, so technical surprises and usability issues surface when change is costliest
- **Unrealistic requirement assumptions**
  - Presumes that *all* functional and non-functional needs can be captured up front
    - Leads to *analysis paralysis* (over-specifying edge cases)
    - Or to “paper agreements” that collapse once real users see the product
- **Weak adaptability to change**
  - Formal change-request process adds heavy overhead; scope creep or market shifts can derail schedules
- **Front-loaded customer involvement**
  - Stakeholders disengage after sign-off, so mis-alignment may go unnoticed for months
- **Heavy documentation burden**
  - Producing and maintaining exhaustive artefacts inflates effort without guaranteeing quality
- **Optimistic planning bias**
  - Budget and timeline estimates rely on perfect knowledge and flawless execution—rare in complex or innovative work
- **Poor fit for high-volatility or research-oriented projects**
  - When technology, domain, or user expectations evolve rapidly, Waterfall’s linear model becomes a liability

Waterfall versus Agile



Lean Software Development

- From the automotive world, **lean principles** have been applied to project with a similar level of complexity: software.
- **The seven principles**
  - **Eliminate waste**
  - **Amplify learning**
    - Iteration plus feedback
    - Keep the solution change tolerant
  - **Decide as late as possible**
  - **Deliver as fast as possible**
    - Delivery value to customer
    - Limit work in progress (Stop starting – Start Finishing)
  - **Create knowledge**
  - **Build integrity** (No Flaws, No Defects, No Decay)
    - Technical practices: align development with business, test-driven design, refactoring…
  - **See the whole**
    - And avoid sub-optimization

Agile Software Development values

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

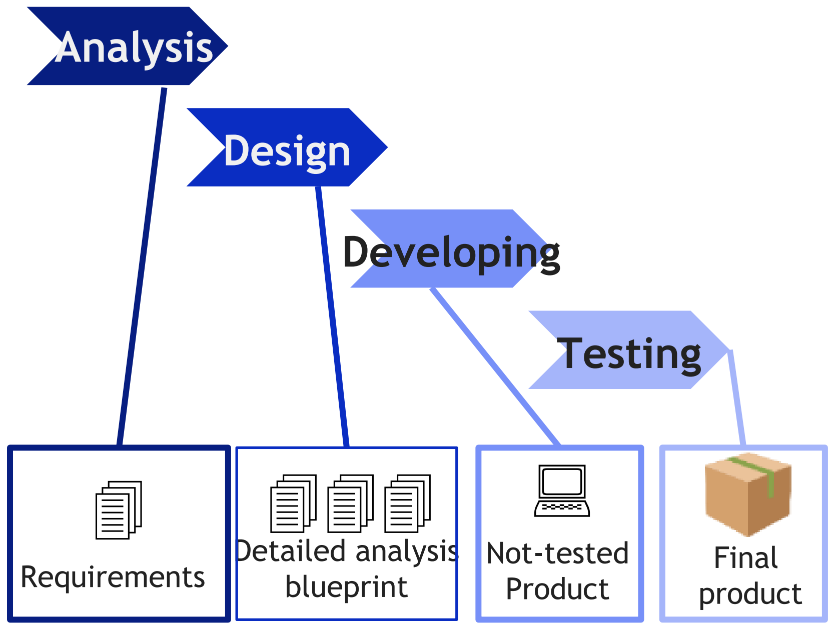
Agile Manifesto ©2001

Principles behind the Agile Manifesto

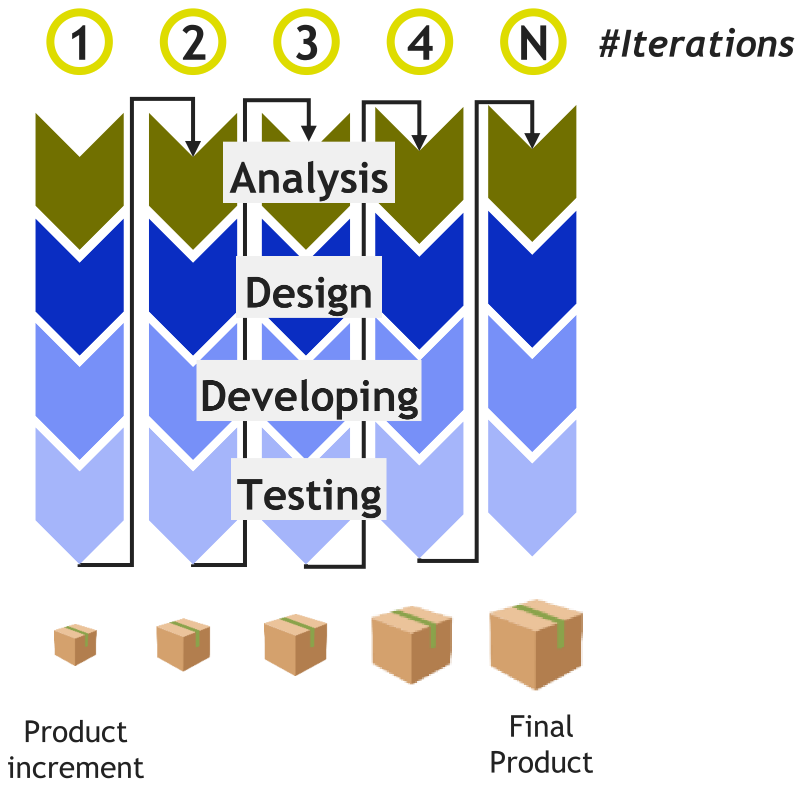
1. Satisfy the Customer
2. Welcome Change
3. Deliver Frequently
4. Work as a Team
5. Motivate People
6. Communicate Face-to-Face
7. Measure Working Software
8. Maintain Constant Pace
9. Excel at Quality
10. Keep it Simple
11. Evolve Designs
12. Reflect Regularly

Waterfall vs Agile

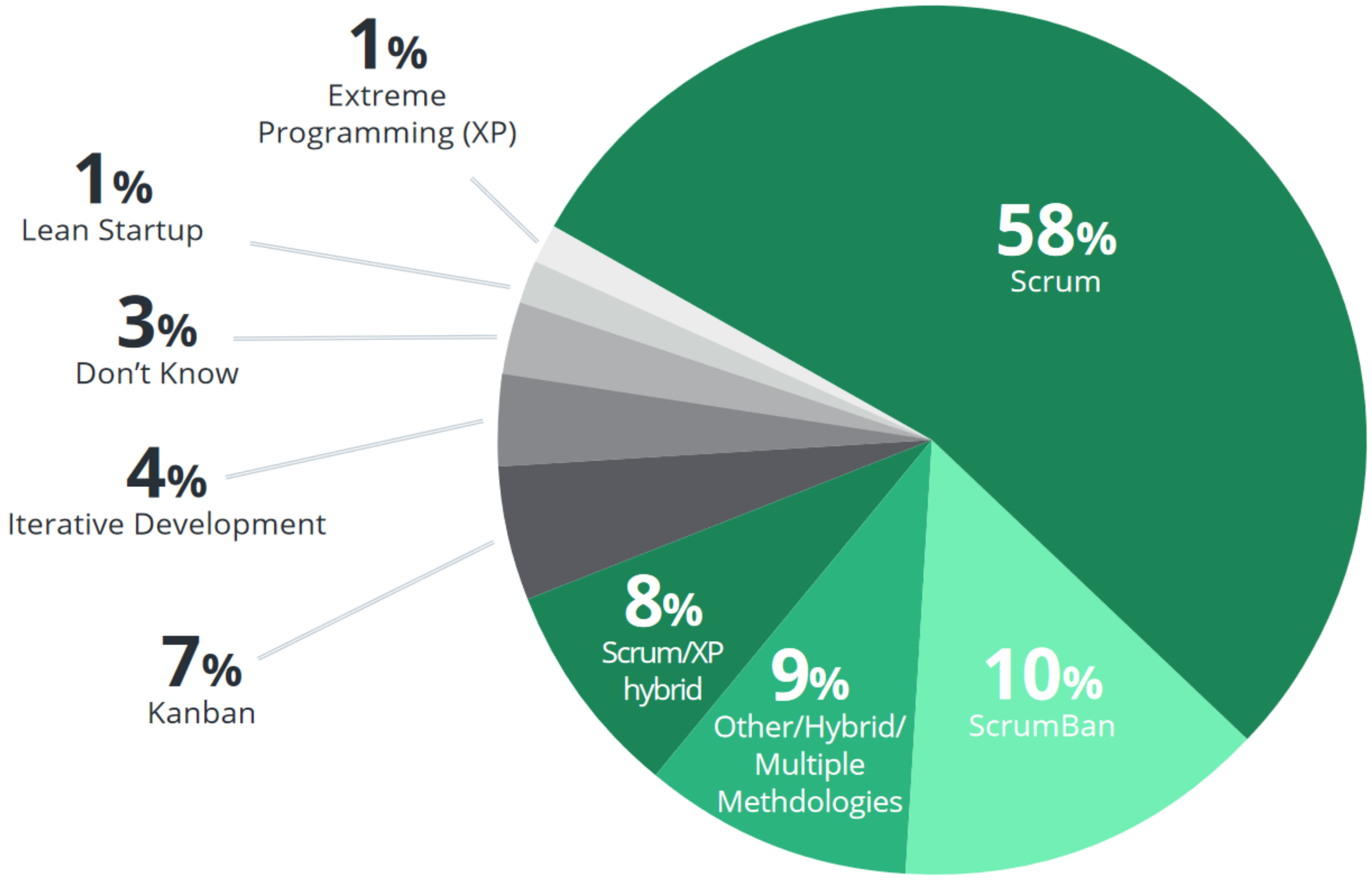
TRADITIONAL PROJECT MANAGEMENT



AGILE PROJECT MANAGEMENT



Agile methodologies



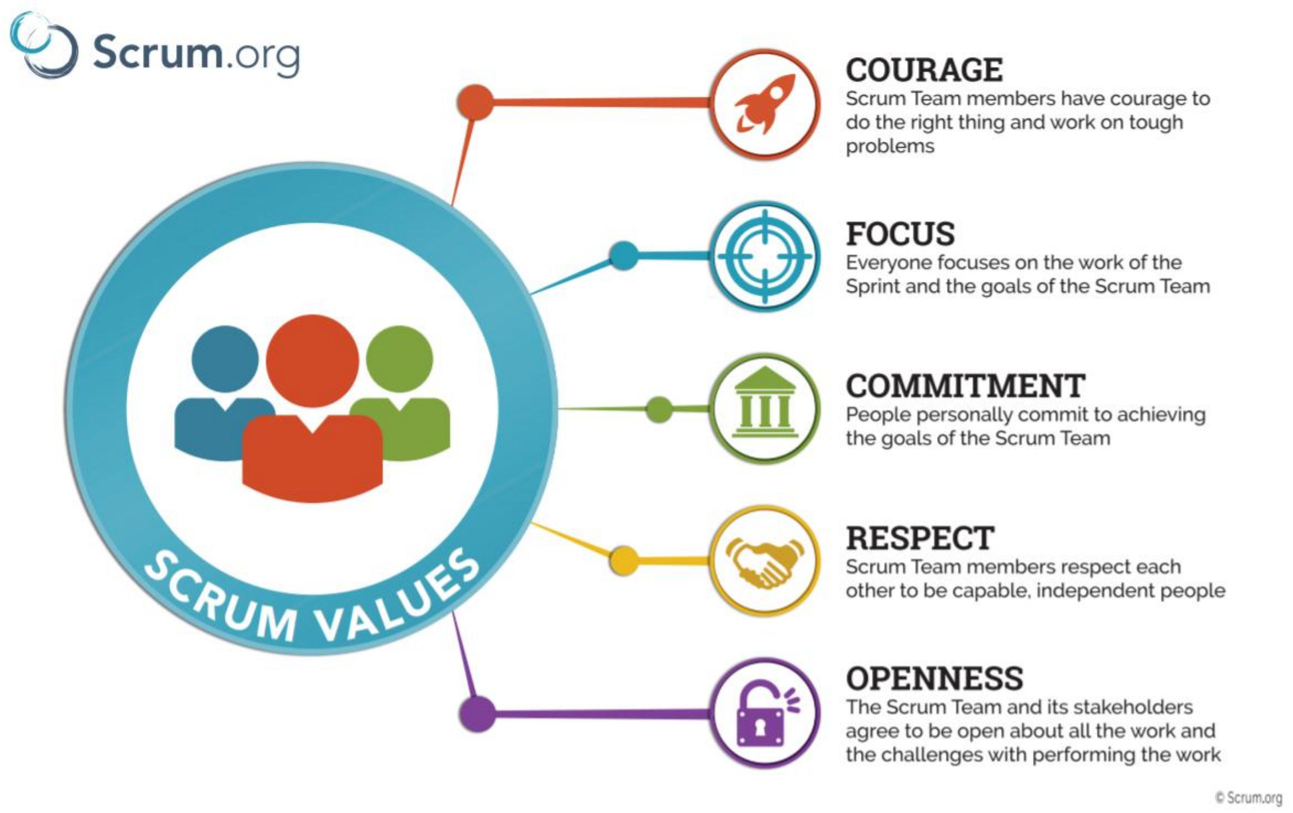
Total exceeds 100% due to rounding.

Source: 14° annual state of Agile Report

Scrum Overview

- Scrum is a framework for Agile Project Management / Product Development. It gets its name from the rugby “scrum” and encourages teams to learn through experiences, self-organize while working on a problem, and reflect on their wins and losses to continuously improve
- Goal-orientation
    - Processes and best practices are the tools used to create working software
    - Results are the most important thing
- Clear definition of roles and responsibilities:
    - Team is completely responsible for its output
    - Team is self-managed
    - Team must deliver outputs at a predetermined pace

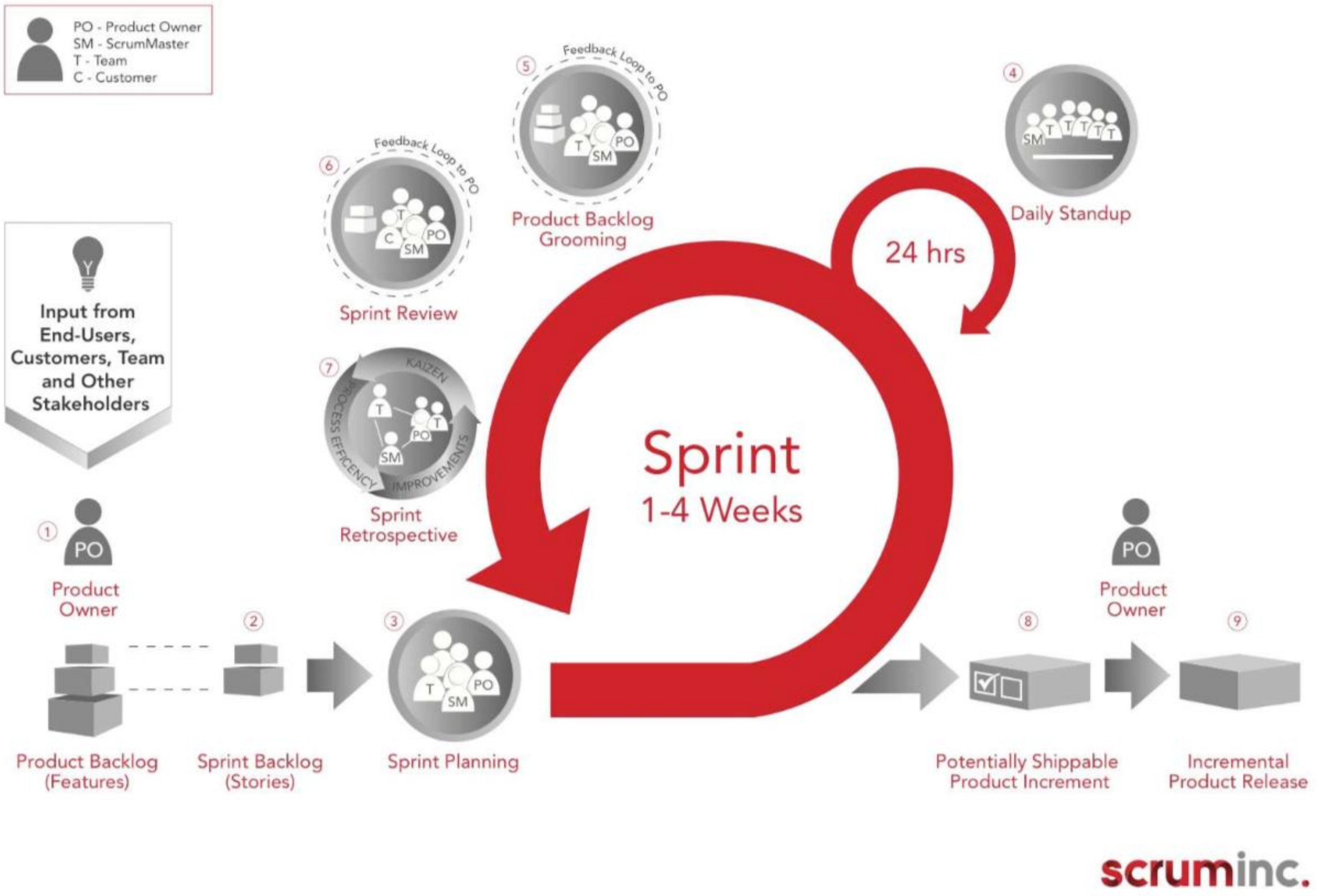
SCRUM Values


Scrum.org
SCRUM VALUES
COURAGE
Scrum Team members have courage to do the right thing and work on tough problems
FOCUS
Everyone focuses on the work of the Sprint and the goals of the Scrum Team
COMMITMENT
People personally commit to achieving the goals of the Scrum Team
RESPECT
Scrum Team members respect each other to be capable, independent people
OPENNESS
The Scrum Team and its stakeholders agree to be open about all the work and the challenges with performing the work
© Scrum.org

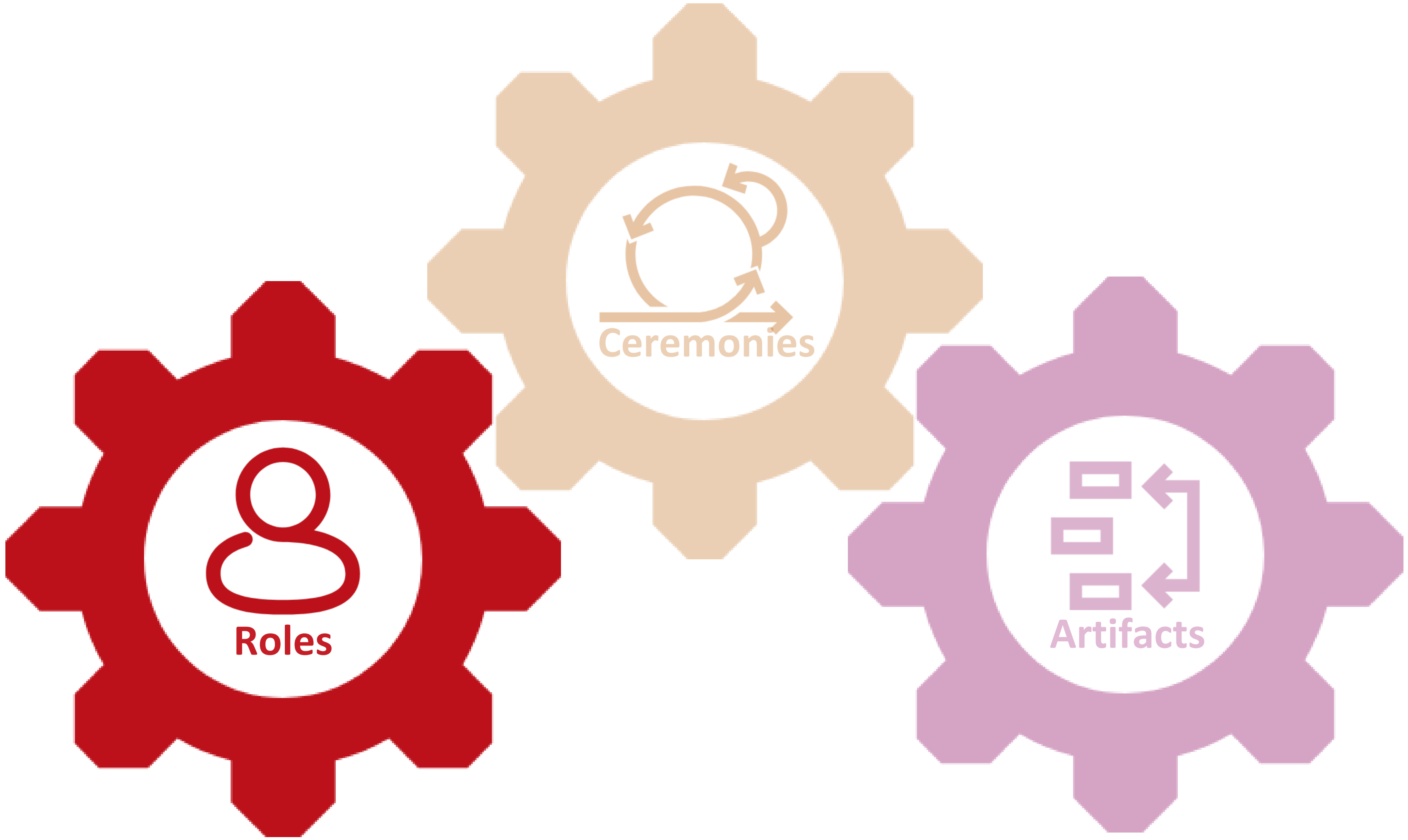
Scrum focuses the organization on:

- Realization of a working product
- Functional features released at regular intervals
- Expecting changes in requirements, architecture and design

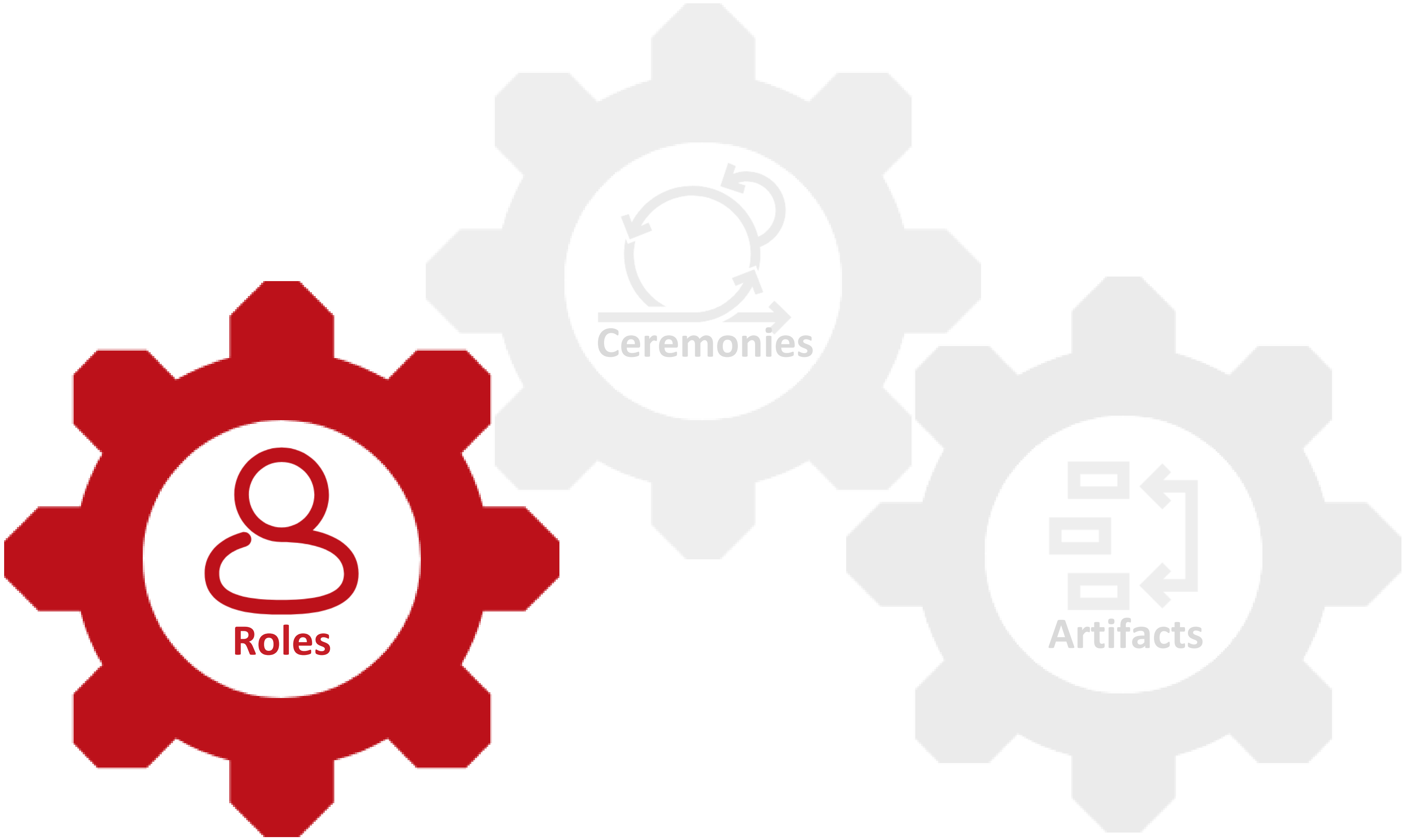
How does SCRUM work?


PO - Product Owner
SM - ScrumMaster
T - Team
C - Customer
Input from End-Users, Customers, Team and Other Stakeholders
1 PO Product Owner
Product Backlog (Features)
2 Sprint Backlog (Stories)
3 T SM PO Sprint Planning
4 SM T T T T T Daily Standup
24 hrs
5 Feedback Loop to PO T SM PO Product Backlog Grooming
6 Feedback Loop to PO T C SM PO Sprint Review
7 KAIZEN PROCESS EFFICIENCY IMPROVEMENTS T T PO SM Sprint Retrospective
Sprint
1-4 Weeks
8 Potentially Shippable Product Increment
PO Product Owner
9 Incremental Product Release
scruminc.

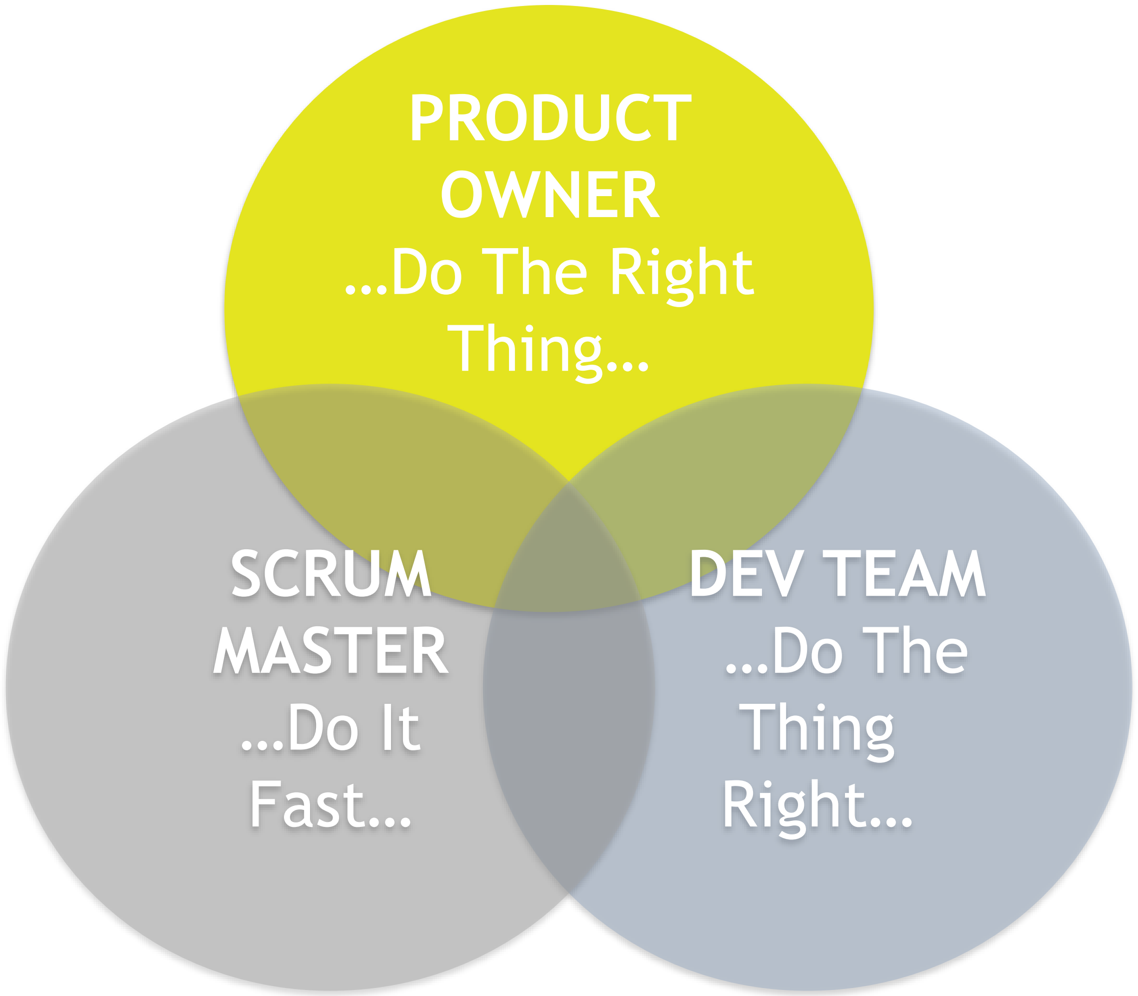
Scrum key components



Scrum key components

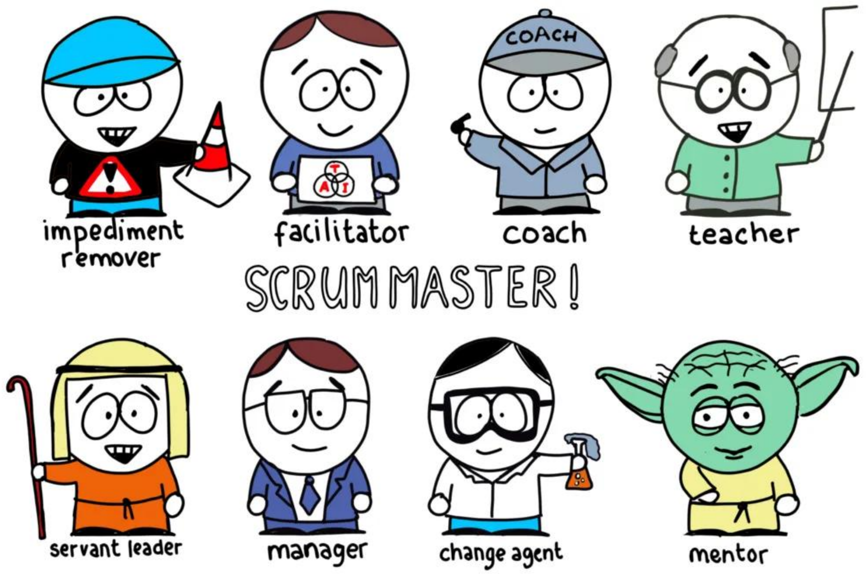


Roles


PRODUCT OWNER
…Do The Right Thing…
SCRUM MASTER
…Do It Fast…
DEV TEAM
…Do The Thing Right…

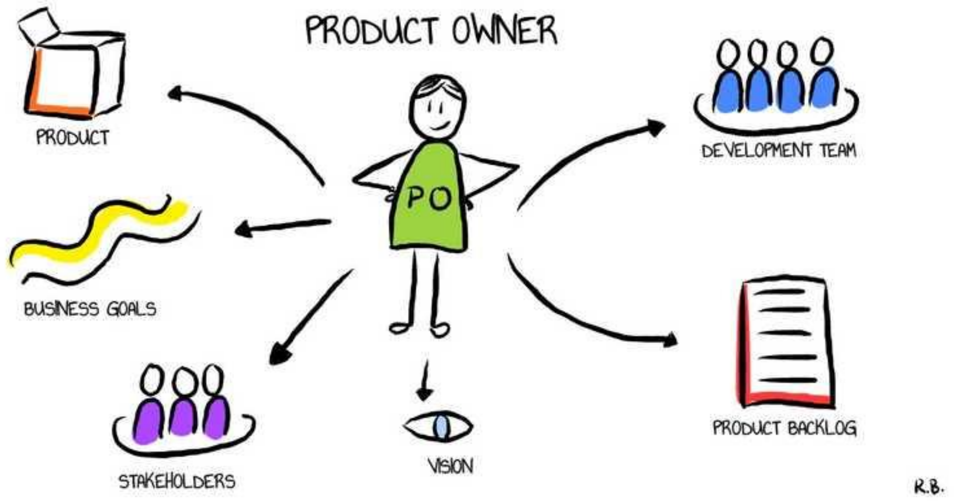
The ScrumMaster

- "Servant Leader" for the Team
- Responsible for the processes in the framework
- Plans and manages meetings (cerimonies)
- Ensures all principles and values are respected
- Impediment Remover for the dev team
- The ScrumMaster and Product Owner must be different people to avoid conflict of interest
- Often the ScrumMaster is the team leader or a senior team member. The Scrum Master requires a lot of effort; for a team of 9 people the Scrum Master is a full time job



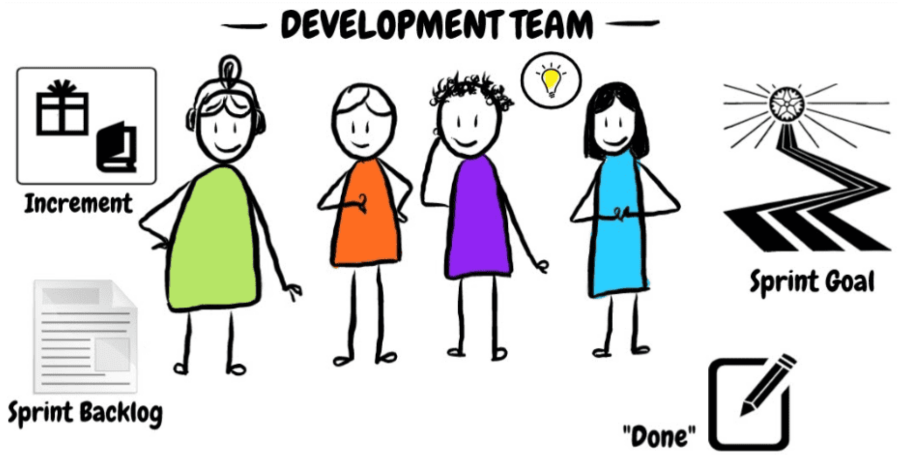
The Product Owner

- Is the owner of the output
- Defines release dates and the contents for each release
- Is the person in charge of quality, cost and time compliance
- Prioritizes software features according to requirements
- Modifies software features and priorities every 7-30 days
- Accepts or refuses the team’s deliverables

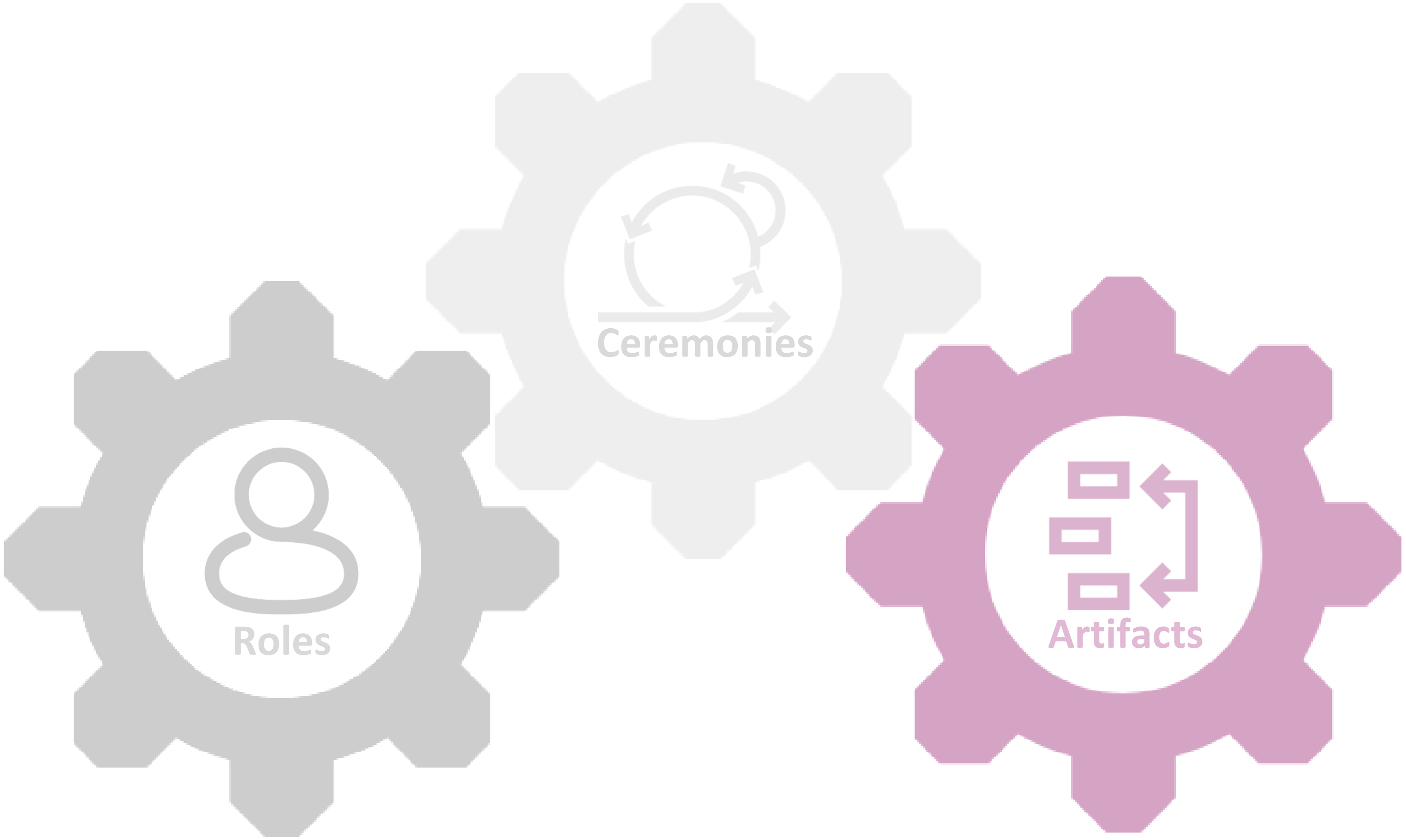


The Team

- Team has 7 (+/- 2) members
- Cross-skill competences: QA, Developers, UI Designers, Architects etc.
- Selects the activities to do during the Sprint from the Product Backlog
- Estimates and divides the activities into tasks
- Manages the situation if/when some team members are also allocated to other projects (ideally they are full-time only on 1 project)
- Shows the outputs at the end of each sprint (e.g. demo)
- According to business rules and processes, the team can manage itself. The team members will assume the responsibilities and risks only if they have space to manoeuvre



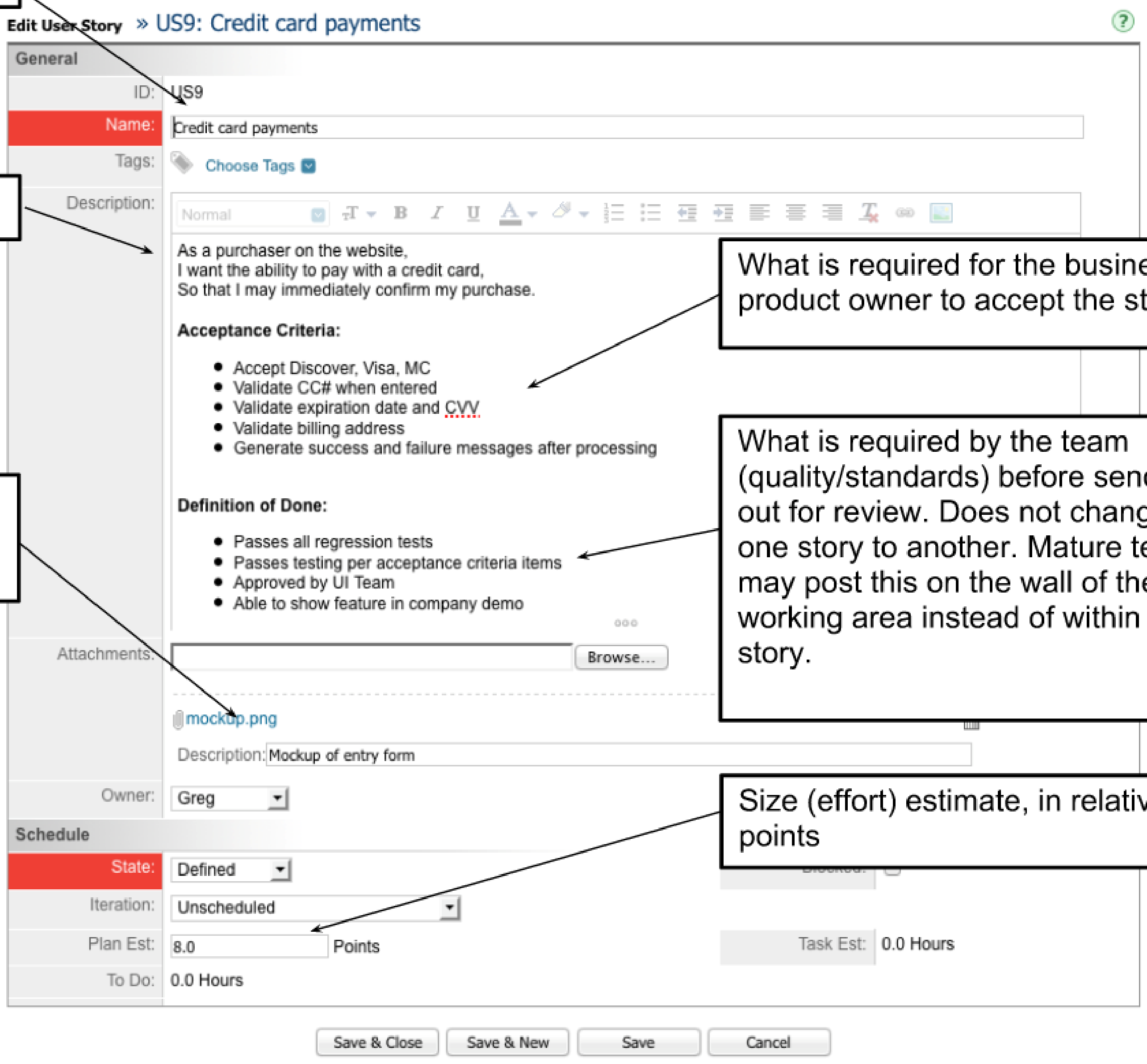
Scrum key components



Stories

- In SCRUM, the requirements are defined as Stories
- Features:
  - Are similar to UML use cases, but have less detail
  - Have a title, a short description and reference documents / screenshots
  - Can be divided in 3 categories:
    - **User Stories**: user-facing functional requirements
    - **Technical Stories**: non-functional requirements, usually written by team
    - **Defects or bug report**
- Stories are called strong when there is a good enough level of description to allow for division into different talks with related estimates

2
(



Story name

Value statement

Attach details and documents when necessary

What is required for the business and product owner to accept the story

What is required by the team (quality/standards) before sending out for review. Does not change from one story to another. Mature teams may post this on the wall of the team working area instead of within each story.

Size (effort) estimate, in relative points

Edit User Story » US9: Credit card payments

General

ID:	US9
Name:	Credit card payments
Tags:	Choose Tags
Description:	As a purchaser on the website, I want the ability to pay with a credit card, So that I may immediately confirm my purchase.

As a purchaser on the website,
I want the ability to pay with a credit card,
So that I may immediately confirm my purchase.

Acceptance Criteria:

- Accept Discover, Visa, MC
- Validate CC# when entered
- Validate expiration date and CVV
- Validate billing address
- Generate success and failure messages after processing

Definition of Done:

- Passes all regression tests
- Passes testing per acceptance criteria items
- Approved by UI Team
- Able to show feature in company demo

Attachments: mockup.png
Description: Mockup of entry form

Owner: Greg

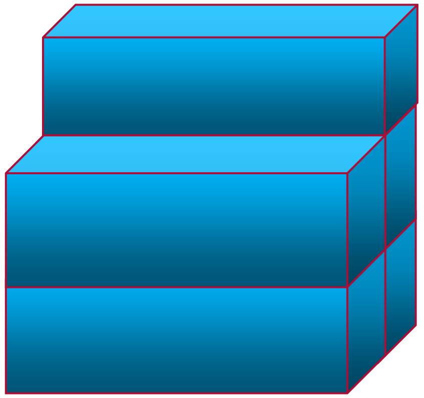
Schedule

State:	Defined
Iteration:	Unscheduled
Plan Est:	8.0 Points
To Do:	0.0 Hours
Task Est:	0.0 Hours

Product backlog

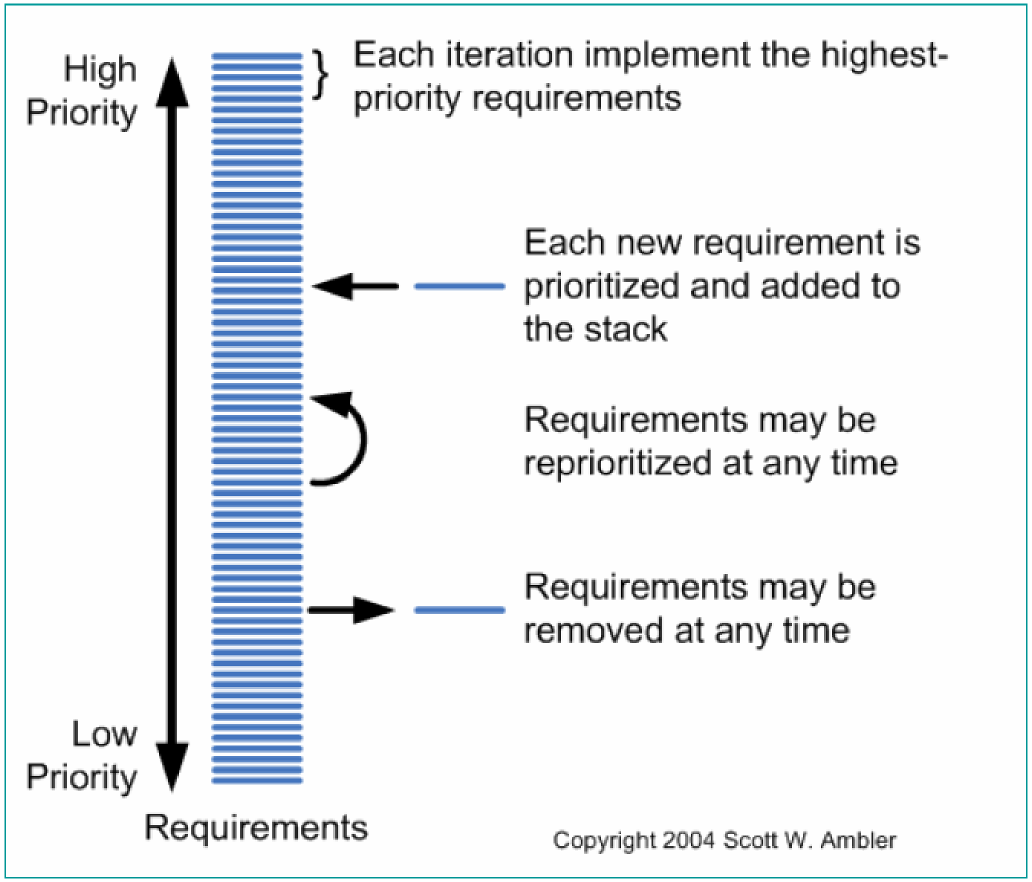
- Is a list of stories
- The Product Owner prioritizes the stories with the support of other stakeholders:
  - e.g. Marketing Manager, Internal Customer
  - Team can also help prioritize, e.g. the team can pinpoint a technological constraint that needs a specific position in the Product Backlog priority list
- The Product Owner remains the main person in charge of the Product Backlog contents and priotities

Product backlog



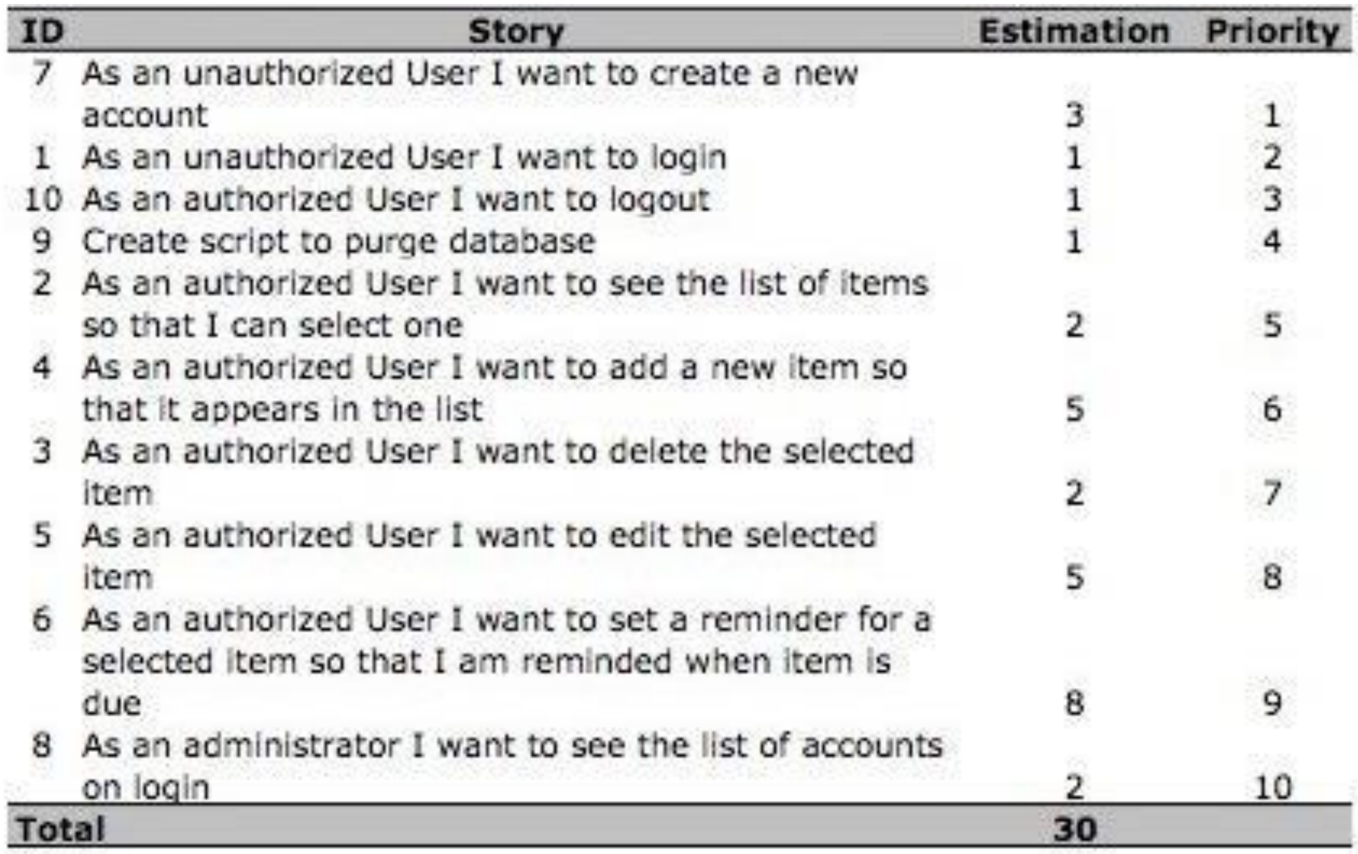
- A single prioritized list of features/tasks
    - Sort Criteria: business value, release presence, etc
- Each Product Backlog item has an implementation-effort estimate
- Includes each product feature and all necessary activities for implementation
    - The Team doesn't do anything that is not listed in the Backlog
    - When high priority items are selected for a Sprint, they are split by team into easy estimated tasks
    - It's hard to understand the exact split level
- Product Owner is the person in charge for the Product Backlog:
    - If the Product Backlog is not updated and kept "alive", then the project will fail
    - Only tasks in Product Backlog are implemented by the team

Product backlog


High Priority
Each iteration implement the highest-priority requirements
Each new requirement is prioritized and added to the stack
Requirements may be reprioritized at any time
Requirements may be removed at any time
Low Priority
Requirements
Copyright 2004 Scott W. Ambler

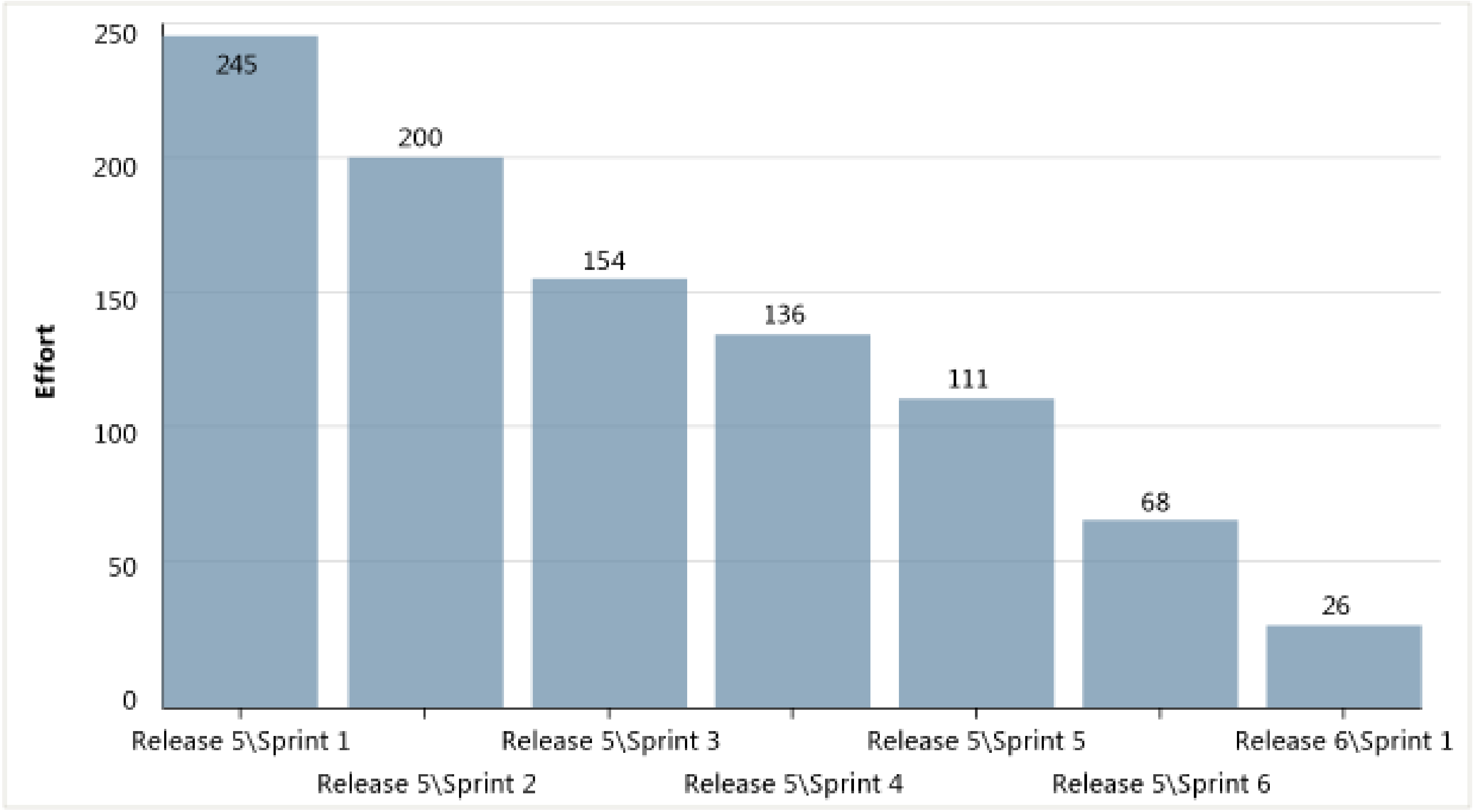
Iteration = Sprint
Requirement = Item/Feature

Product backlog example



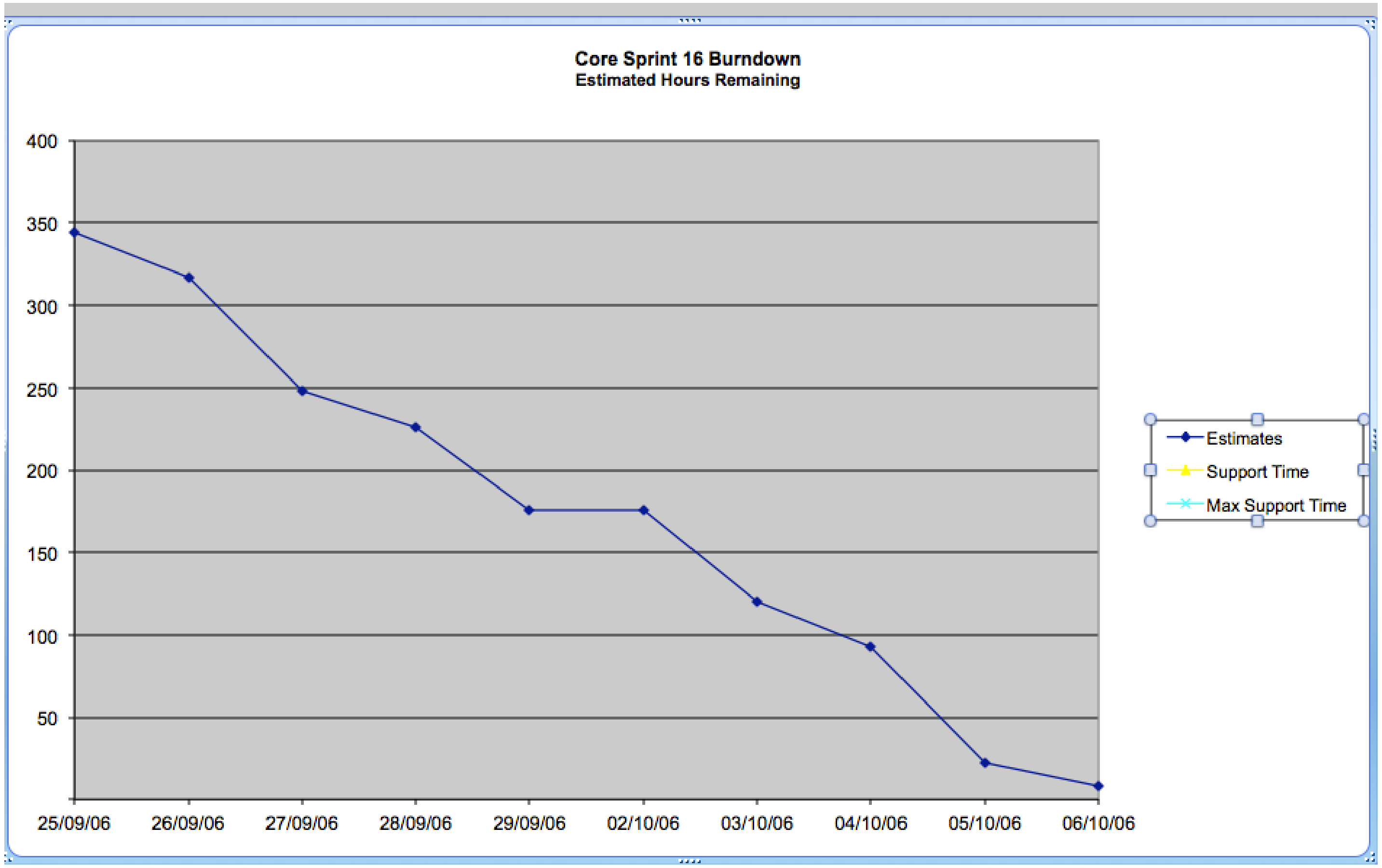
ID	Story	Estimation	Priority
7	As an unauthorized User I want to create a new account	3	1
1	As an unauthorized User I want to login	1	2
10	As an authorized User I want to logout	1	3
9	Create script to purge database	1	4
2	As an authorized User I want to see the list of items so that I can select one	2	5
4	As an authorized User I want to add a new item so that it appears in the list	5	6
3	As an authorized User I want to delete the selected item	2	7
5	As an authorized User I want to edit the selected item	5	8
6	As an authorized User I want to set a reminder for a selected item so that I am reminded when item is due	8	9
8	As an administrator I want to see the list of accounts on login	2	10
Total		**30**	

Product burndown chart example



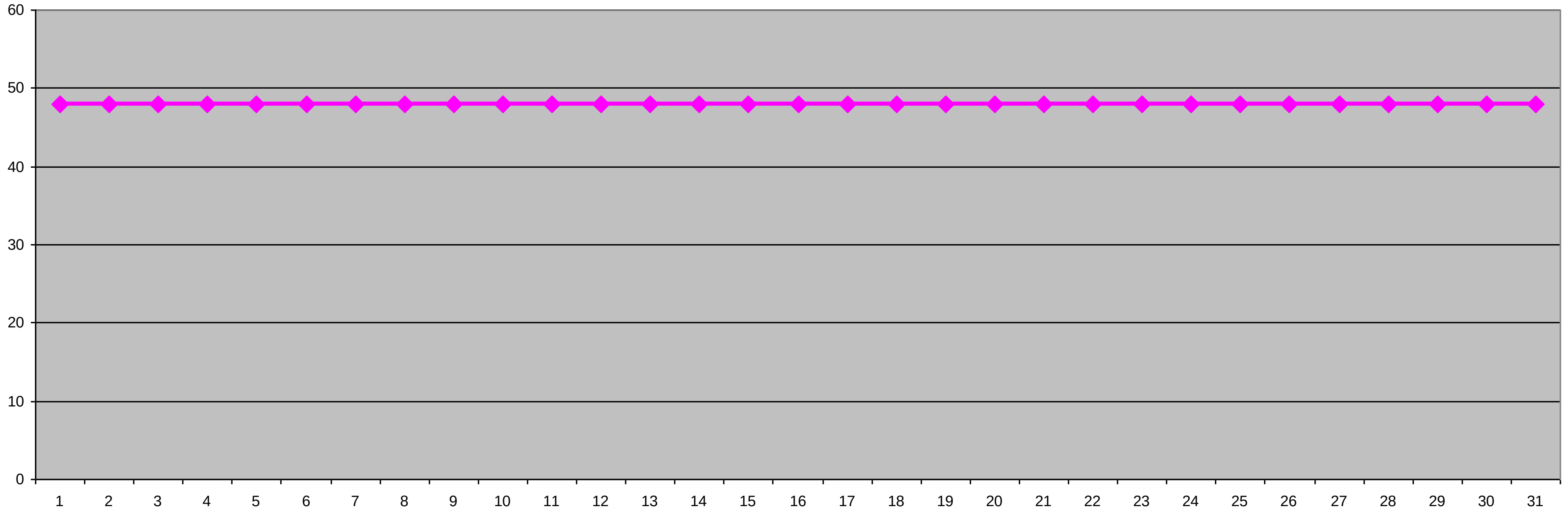
Product burndown chart example

Well done



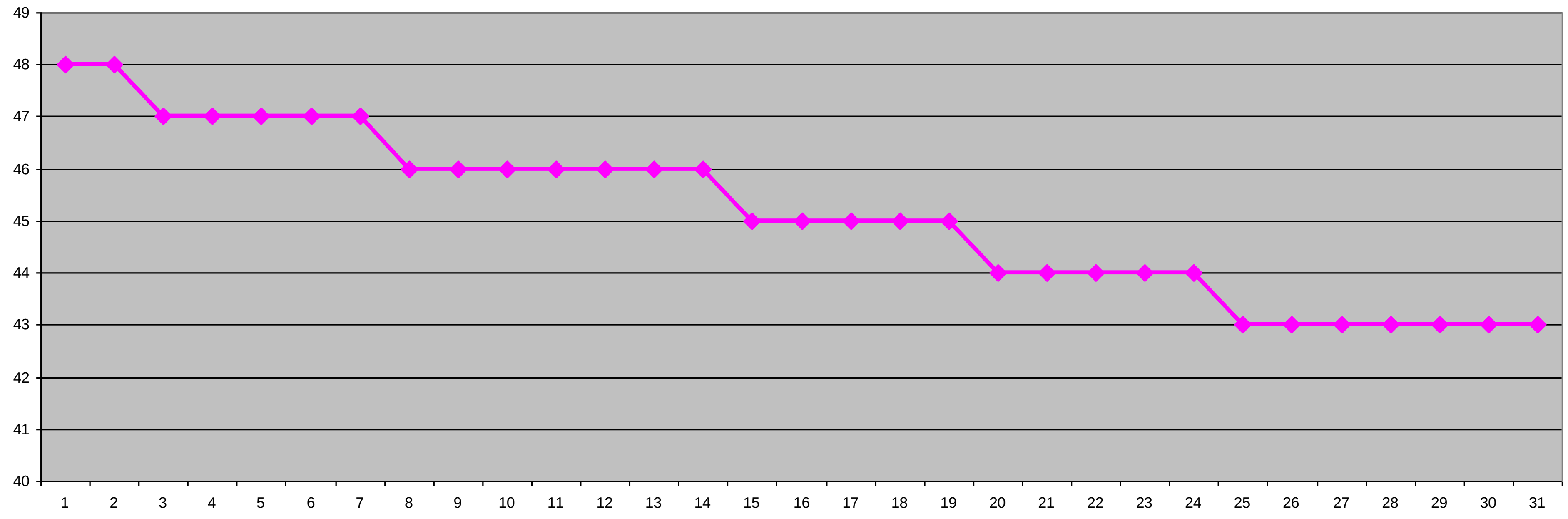
Burndown chart

No work being performed



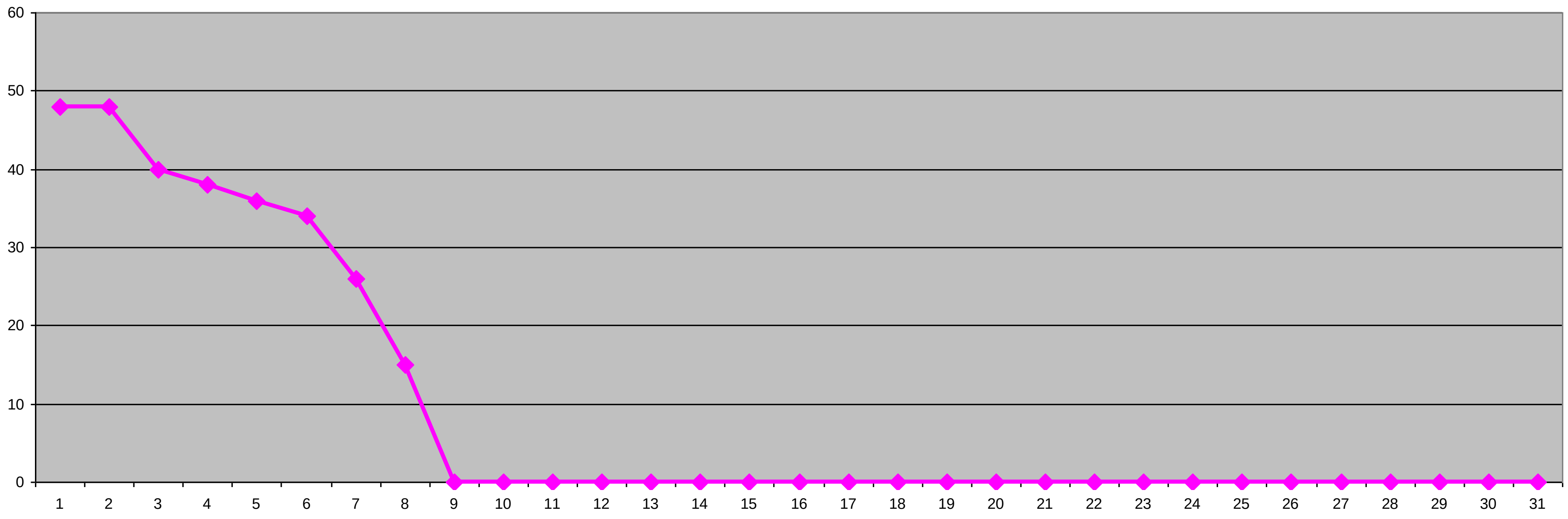
Burndown chart

Work being performed, but not fast enough



Burndown chart

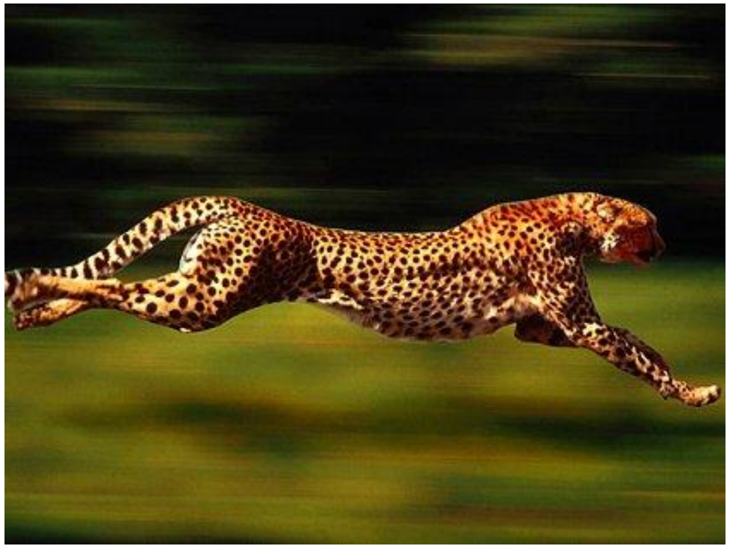
Work being performed, but too fast!



Velocity

- Measures team performance, the capacity to transform a requirement in a functional increase during a Sprint

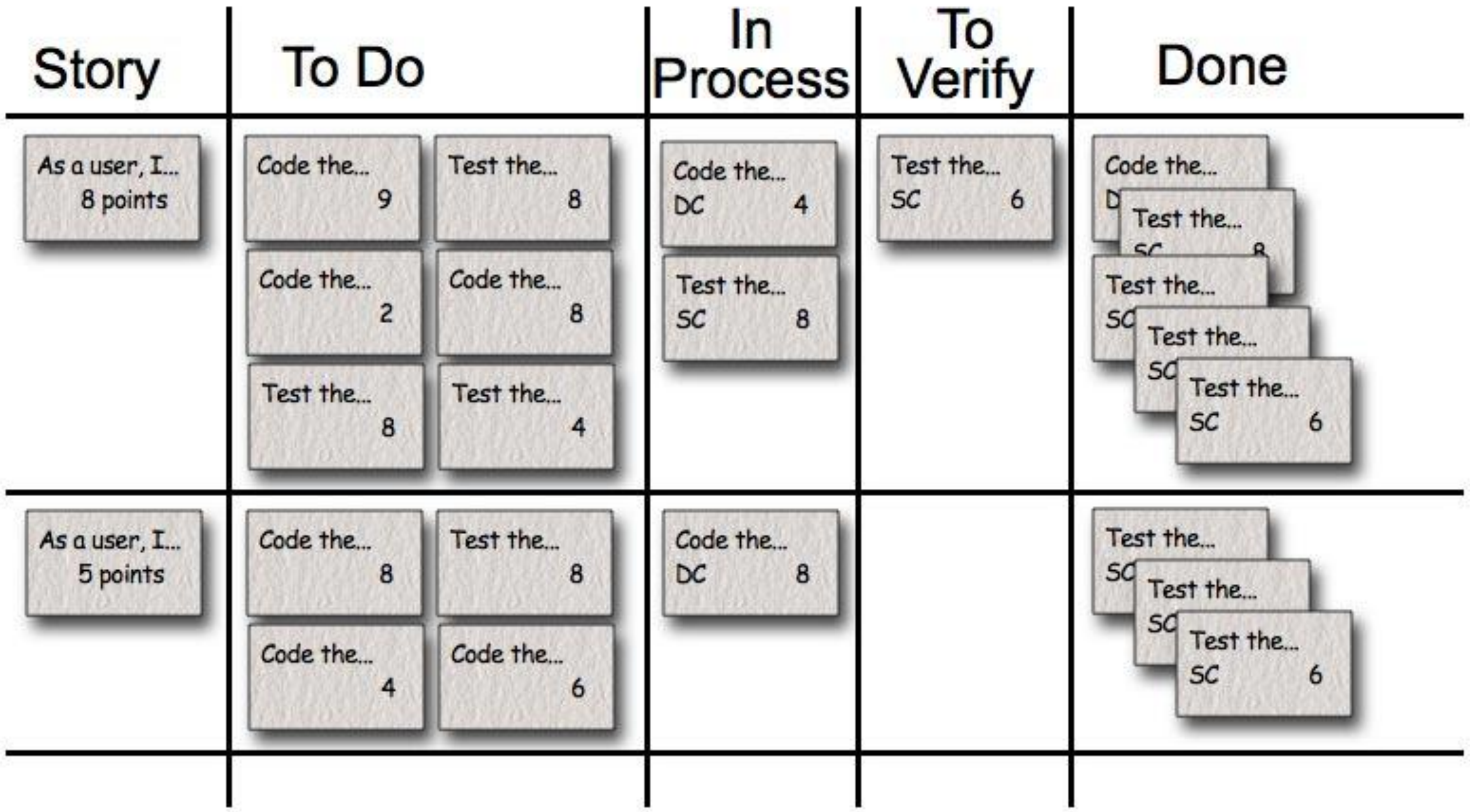
- The measurement could be:
    - Functional points
    - Ideal hours
    - Story points



Increment of Shippable Product

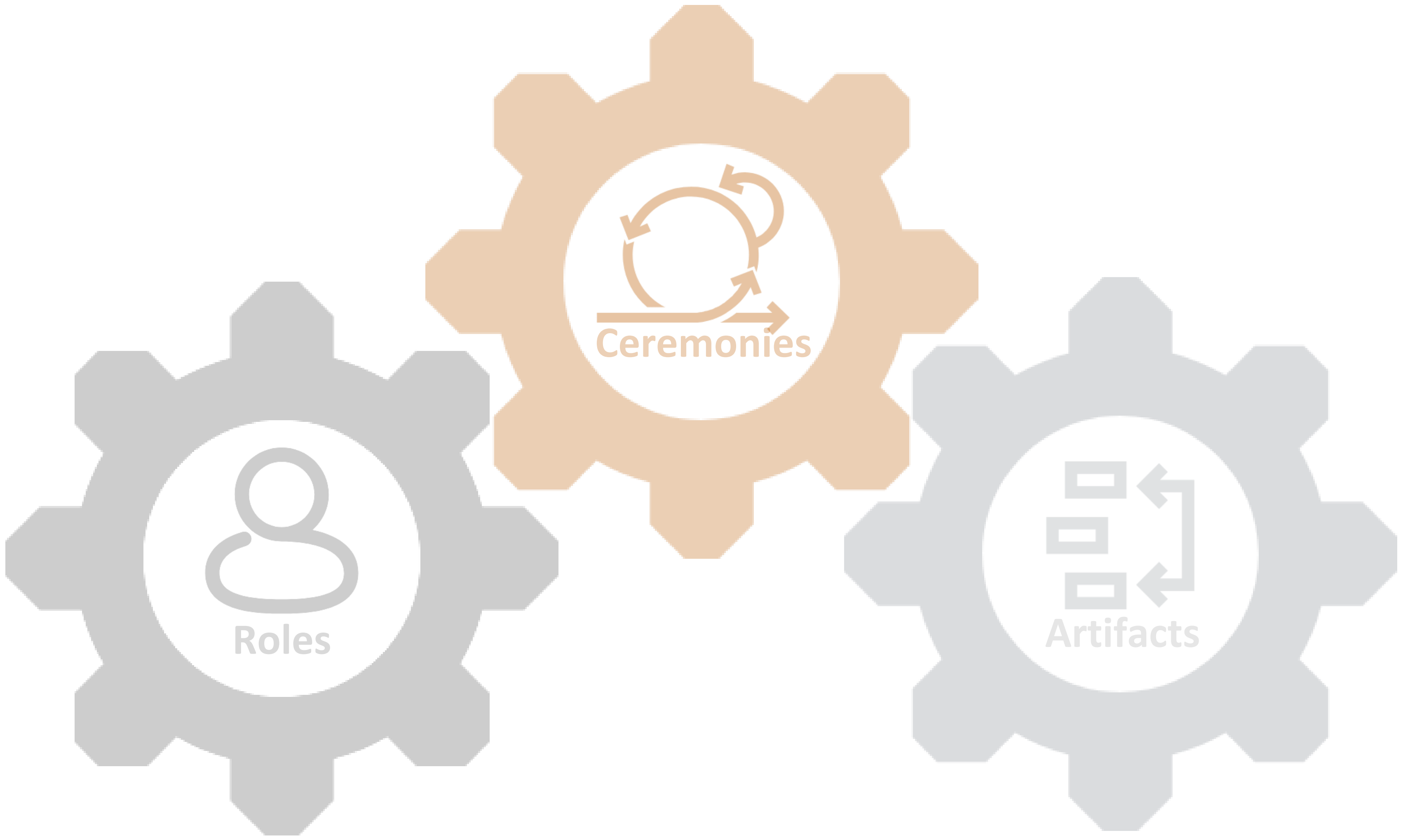
- Development teams should build an increment of product functionality every Sprint
- Continuous Delivering smaller functionalities using an iterative and incremental process
- This increment should be potentially shippable
- The increment consist of thoroughly tested code that has been built into an executable, and the user operation of the functionality is documented

Task boards example



Taskboard example

Scrum key components



Sprint Planning

The goal of Sprint Planning is to define which stories to select for the Product Backlog to implement in the next cycle.

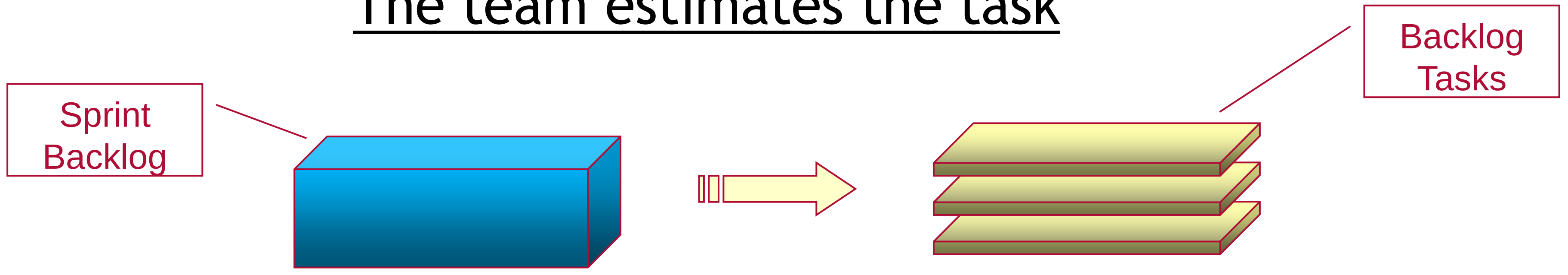
During this meeting:

- Show the Product Backlog
- Define a Sprint goal
- Select some stories for the Sprint
- Make the Sprint backlog
- Define an end date for the Sprint cycle

Sprint Planning Meeting

- The high priority stories are considered:
  - Based on estimates and team abilities. Usually only a small contingency is considered
- Team divides the stories in tasks and it estimates the tasks
- The Product Owner defines the final set of tasks for that iteration, negotiating it with the team

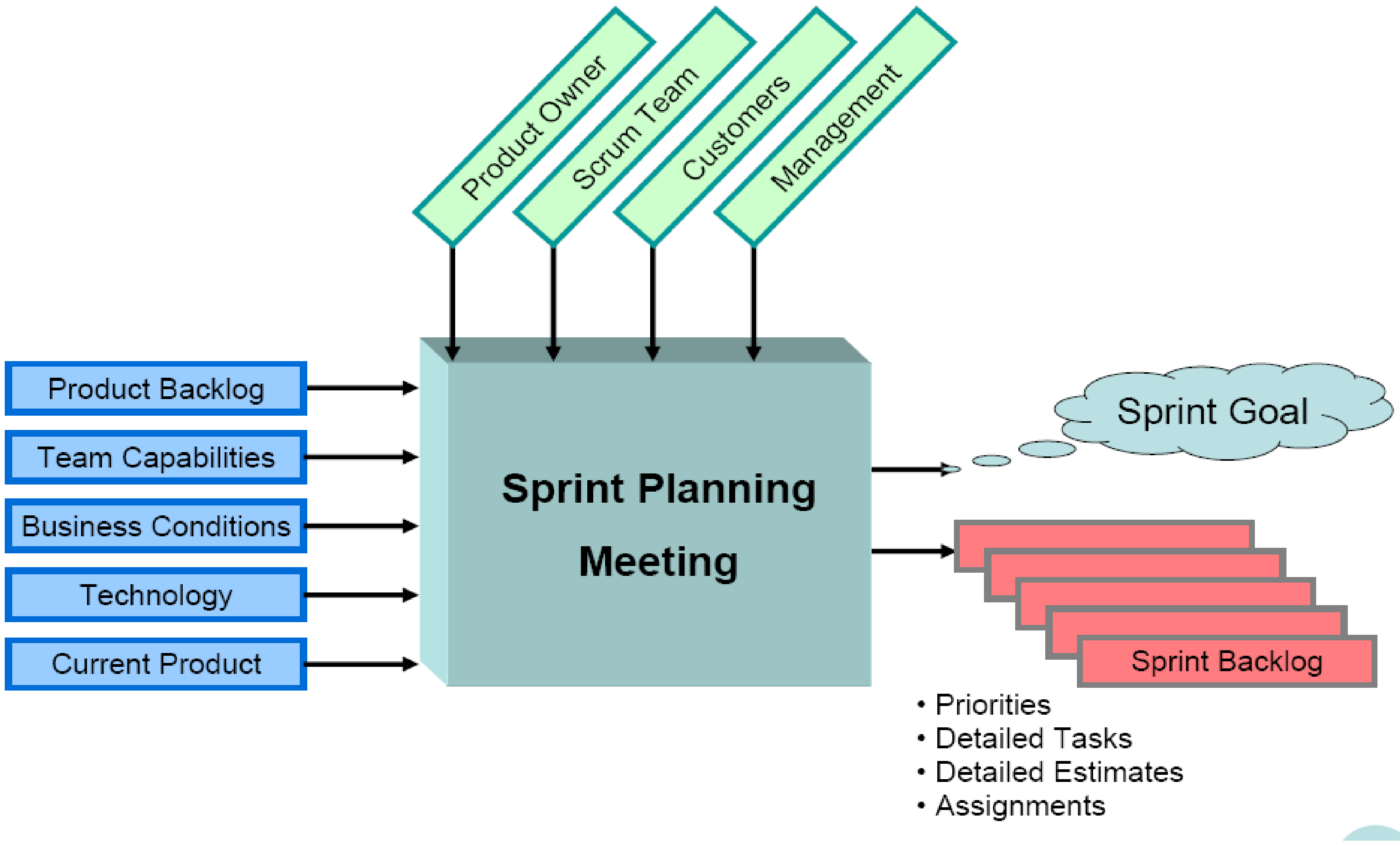
<u>The team estimates the task</u>



Product Backlog estimation

- The story estimation can be done with any scale of measurement. The main ones are:
  - Man-days: direct time measurement
  - Story Point: complexity measurement (easy and strong estimates)
    - Separation with time estimation
    - Understand how many story points the team can realistically execute by measuring velocity
- Teams often use poker cards for estimating

Summary: Sprint planning meeting

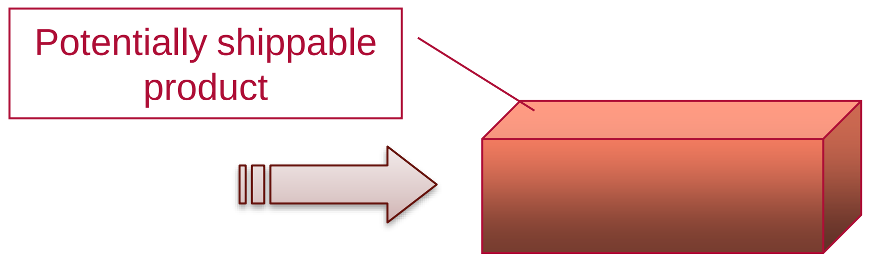


Daily Scrum Meeting

- It is a 15 minutes meeting managed by the Scrum master
- Each team member must answer the following 3 questions:
  - What did I do yesterday?
  - What will I do today?
  - Are there critical situations or obstacles?
- Everybody in the team can participate in this meeting, but only those who are active during the Sprint can talk
- The goals of the meetings are:
  - To have quick snapshot on sprint evolution
  - Share information about the sprint
  - Early risk/criticality management or personal problem resolution

Sprint Review Meeting and Sprint Retrospective

- It is at the end of a Sprint
- The meeting lasts 4 hours and it's time-boxed
- During the first half of the meeting:
  - Product Owner and vested stakeholders watch all planning demos and share feedback
  - Product Owner decides if the Sprint is complete and updates the Product Backlog
  - Product Owner, team and stakeholders discuss Backlog priorities for the next Sprint
  - Next Sprint goal is defined

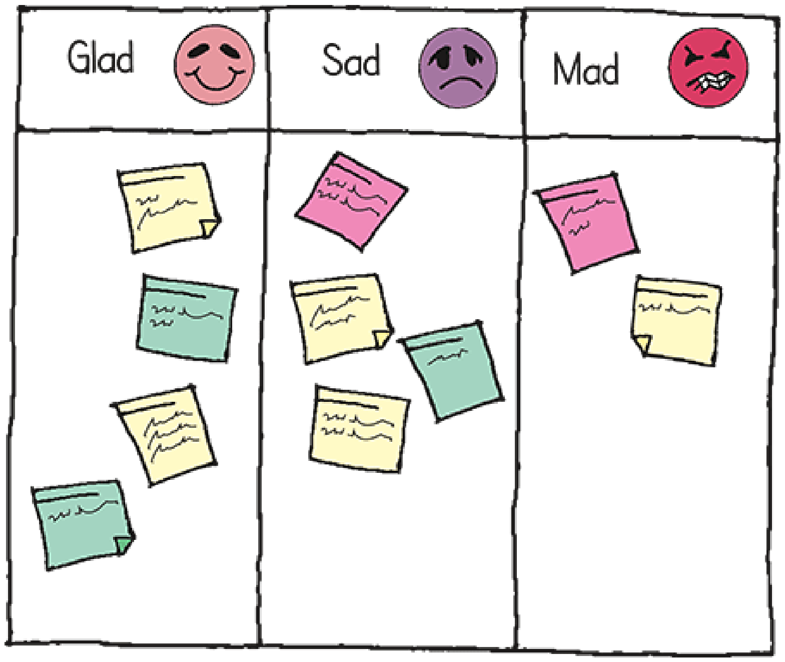


Sprint Retrospective

The second half of the Sprint Review Meeting is a team discussion called Retrospective, managed by the Scrum Master and very useful to enable the continuous improvement

- Analysis of work modalities, identification of pros to repeat in next Sprint
- Analysis of cons (what didn't work well), identification of countermeasures

One of the tools that can be used to manage the Sprint retrospective is the Glad-Sad-Mad board



GLAD

Pleased the team including
- things that went well
- things we learned
- novel ideas
- things to thank people for ("bouquets")

SAD

Disappointed the team specifically
- things to improve

MAD

Frustrated the team including
- things that didn't go so well or
- annoyed or
- wasted time