Software Engineering

Block 2: Software management principles

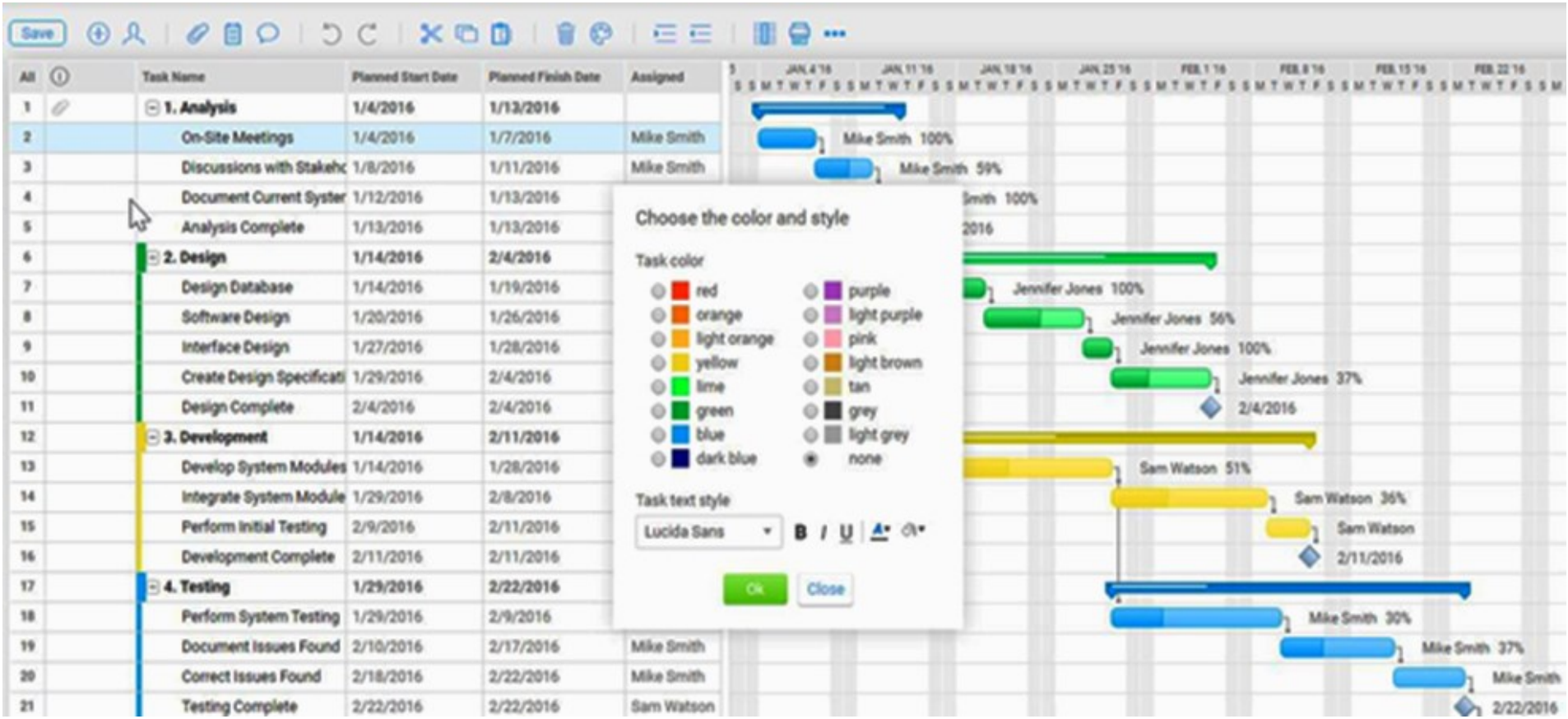
ESCI upf.
School of International Studies
BioInformatics-Software Engineering

Outline

In this session we will analyze two main topics: **Tools** & **SCRUM**

- Introduction to software engineering management
- Tools for Configuration Management Control
- Agile Software Development: SCRUM
  - Characteristics
  - Components of SCRUM :
    - Roles
    - Artifacts
    - Processes
  - The Task Board

Why do we need *management*?



Software engineering costs

- Project development needs to be monitored because the **risks of high costs** are real:

If a customer gets…

…then a lot of money is wasted and software satisfaction is low.

Therefore, the cheaper, faster and better quality, the more successful we will be.

*The objective is then to **reduce costs** through **good management**.*

Soft. Eng. Management Paradigm Comparison

	Advantages	Disadvantages
Waterfall	• Start-End of each defined stage → Progress of the measurable project. • Promote detailed documentation. • Signed agreement of system requirements.	• A real project is never so markedly sequential. It is necessary to contemplate the possibility of returning to previous stages of development. • Difficulty in explicitly establishing all the requirements at the beginning of the project. • The client may not have them clear or may change during the process. • **The client must wait to see the result at the end.**
Protyping	• Improvement of the first three drawbacks of the classical life cycle. • It is developed from a set of known requirements (general objectives)	• The client sees it as definitive software. • Prototypes done with inadequate resources
Incremental	• Iterative model that allows you to develop more and more complete versions, combining the advantages of waterfall and prototyping. • In each iteration, a new feature is added but always take into account the general overview.	• End of phases not defined. • Requirement list not closed until end of project.

Management versus Paradigms

Our development can follow several strategies (paradigms):

- Models of software development:
  - Lineal-sequential (Classic life cycle)
  - Evolutionary (Iterative or Incremental)
- Classics methods (lineal-sequential) have some inconveniences:
  - Planning phase requires a huge effort
  - In environments with fast changes, the specification of requirements can be hard
- So, **evolutionary methods** are currently recommended:
  - Leading model: *Agile Software Development*

But regardless of the strategy, control must be guaranteed.

Configuration Management Control → Repository

Why do we need a repository?

All developments need a process to establish and maintain the consistency of a product (in our case a software solution), during the whole development lifecycle and its deployment.

This process is known as **Configuration Management**.

Consistency involves two areas:

- The items (elements).
- And the control (the procedures).

Both areas are essential, so we target on each.

Configuration Management: Items

In the field of software engineering, **all** items/elements must be under control:

- Documents: Technical documents, diagrams, meeting notes, guides, …
- Code: Source code, test code, releases, …
- Reports: Issues, bugs, feedback, …
- Others: And any other material involved in the software development process.

Configuration Management: Control

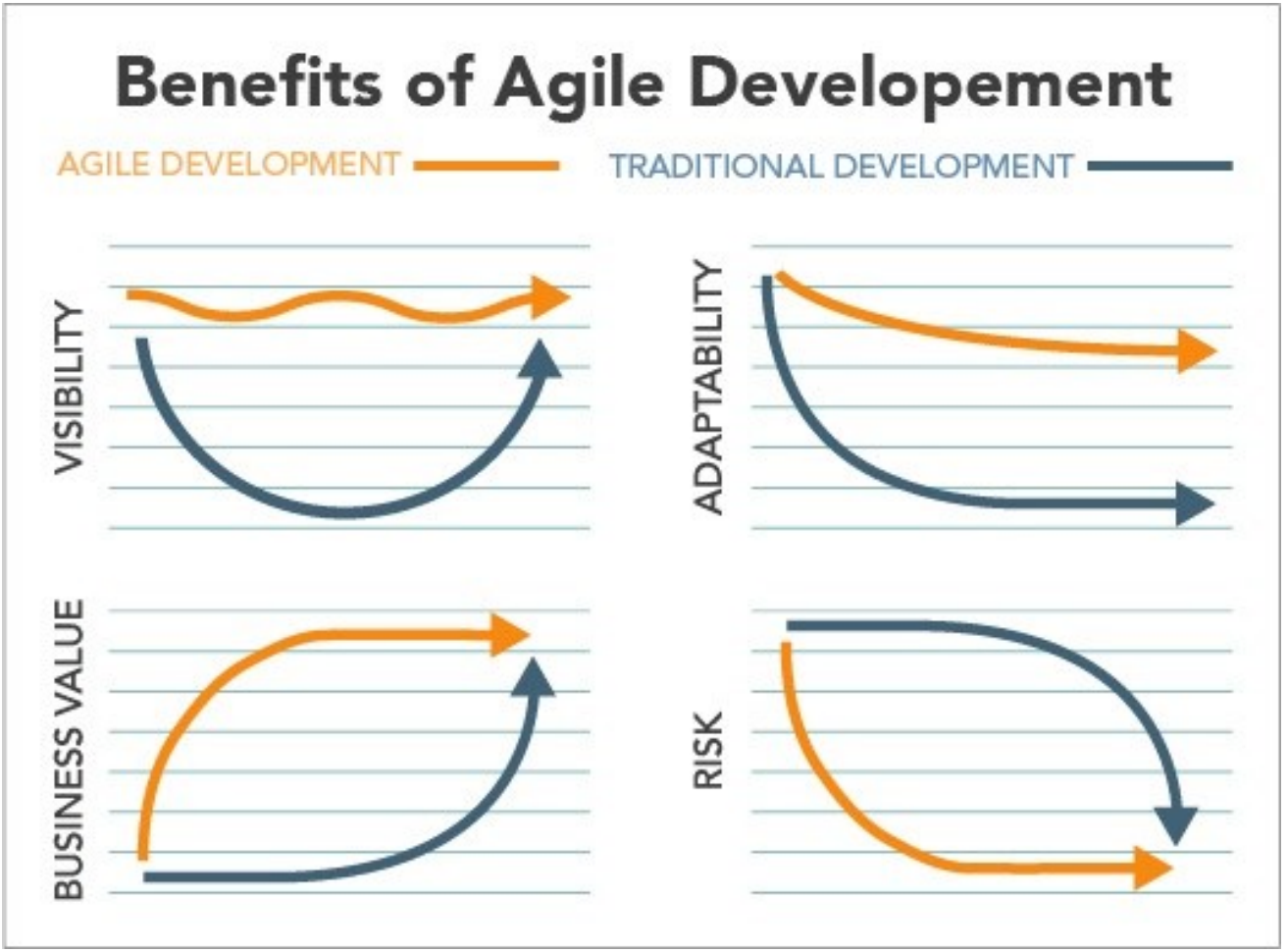
In the field of software engineering, all items/elements must be under control, and…
under control means in this context:

- Identify each element.
- Versioning the elements.
- Trace changes between versions.
- Enumerate the people involved.
- Add a timestamp to each item.
- Centralize the information.

Therefore, the solution to centralize the management is to use a **Repository**.

Do not mix several repositories, use only one!

Why are Agile methodologies successful?



Agile Management

The manifest (principles) of the Agile Software Development:

- People and iterations over processes and tools.
- Intuitive and functional software over extensive documentation.
- Collaboration with the client over contract negotiation.
- Response to the change over following the planification.

Note that Agile Management is a development model, not a methodology!

Agile Management

Implications of the Agile Software Development Manifest (I):

1. Our highest priority is customer satisfaction through early and **continuous delivery** of valuable software.
2. We embrace **changes in requirements**, even late in development. Agile processes exploit customer changes for a competitive advantage.
3. **Deliver** functional software **frequently**, from a couple of weeks to a couple of months, with a preference for the shorter timeframe.
4. Employers and developers should **work together** on a daily basis during the entire project.

Agile Management

Implications of the Agile Software Development Manifest (II):

5. Build projects around **motivated people**. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of transmitting information within and to a development team is the **face-to-face conversation**.
7. **Working software** is the primary measure of progress.
8. Agile processes promote sustainable development. Customers, developers, and users must be able to maintain a **constant rate** indefinitely.

Agile Management

Implications of the Agile Software Development Manifest (III):

9. Continued attention to **technical excellence** and **good design** increases agility.
10. **Simplicity** —"the art of maximizing the amount of work not done", in the sense of eliminating the unnecessary— is essential.
11. The best architectures, requirements and designs emerge from **self-organizing teams**.
12. At regular intervals, the team thinks about **how to be more effective**, and then refines and adjusts its behavior accordingly.

Be efficient by working hard in a short time!

Agile Management: SCRUM

The concept of SCRUM comes from rugby:

- *A scrum is a way to restart the game after an interruption, where the forwards of each side come together in a tight formation and struggle to gain possession of the ball when it is tossed in among them.*

The SCRUM methodology follows the Agile model with the regular benefits of:

- Minimization of risk → short iterations.
- Real time communication (face-to-face) → a few written documentation.
- Indicated for unpredictable requirements.

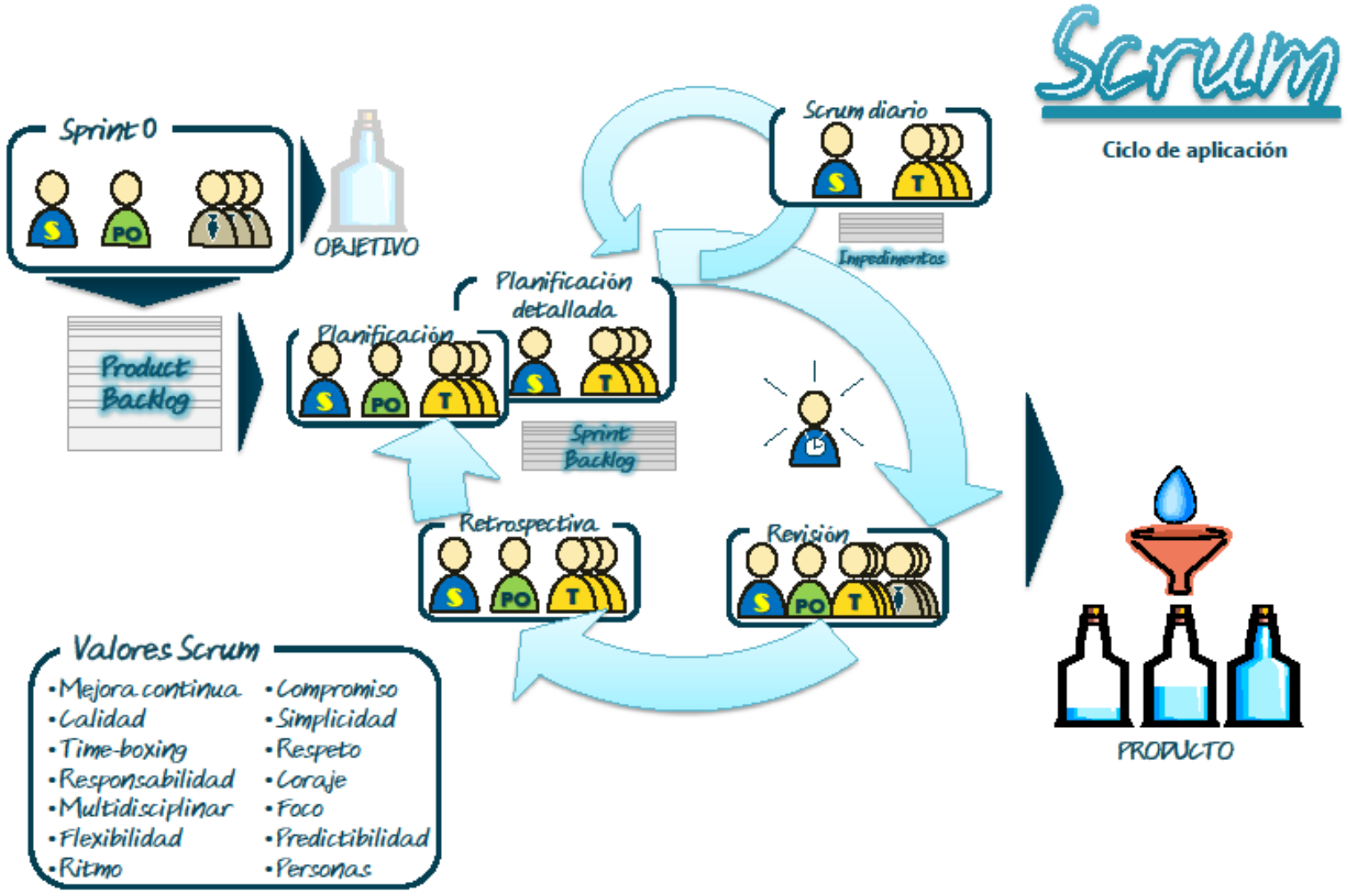
Scrum = agglomeration, "clumping"

Agile Management: SCRUM

General characteristics of the SCRUM methodology:

- **Agile environment** for the development and maintenance of complex products:
  - It allows to tackle complex problems, release products with the maximum value and the shortest period of time.
- **Incremental and iterative** process:
  - It allows to develop products where requirements change rapidly.
- Based on teamwork, so a **self organized team**:
  - Improves communication and maximizes cooperation.
  - Maximizes productivity.
  - Protects the team from interruptions due to impediments/difficulties.
- **Controls the chaos** due to conflicts of interest and needs.

Agile Management: SCRUM architecture

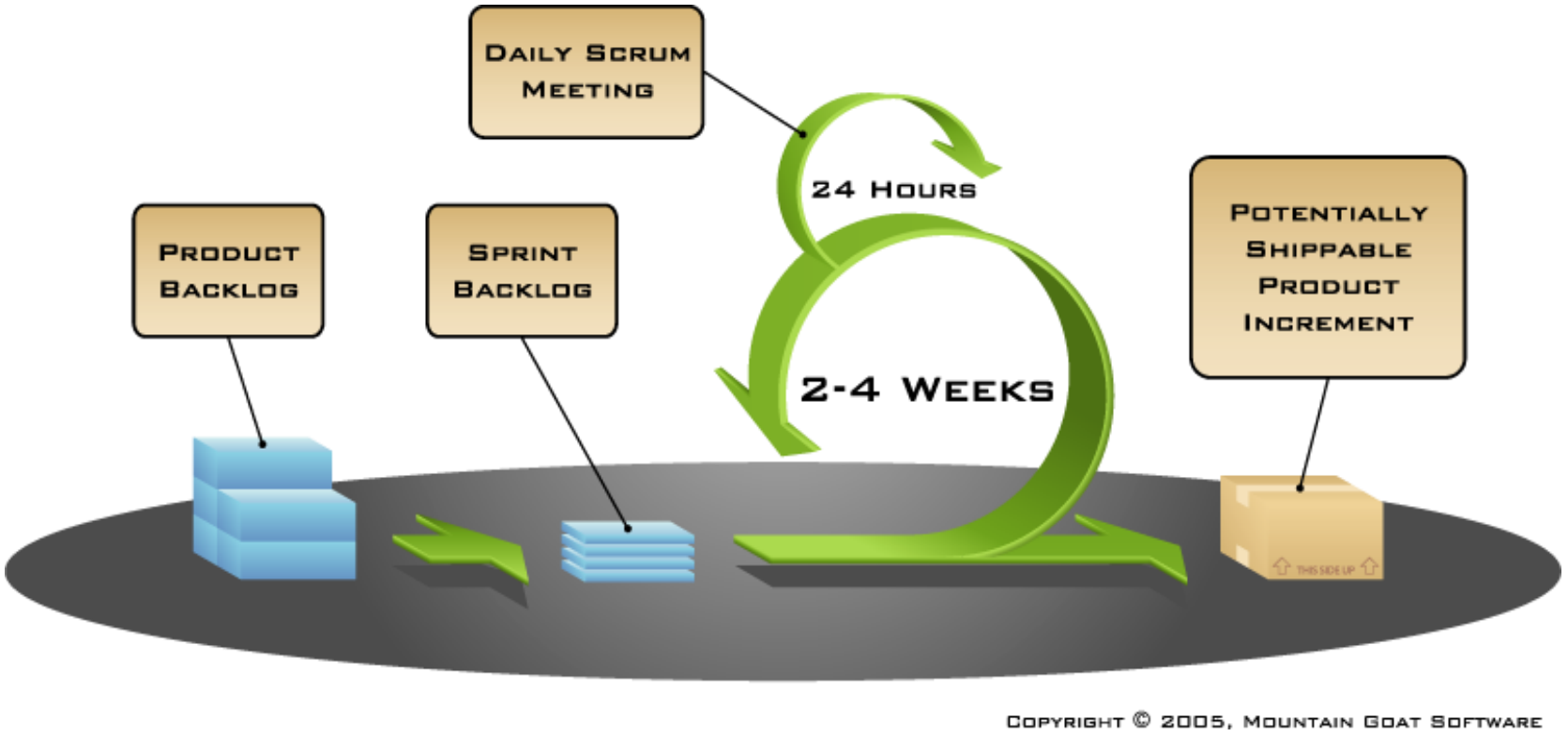


Búscanos en: facebook.com/metodosagiles y en Twitter: @metodosagiles

Agile Management: SCRUM foundations

- It is based on an **empirical process** control.
  - Knowledge comes from experience.
- Scrum uses an **iterative and incremental** approach to optimize the predictability and risk control.
- General fundamentals:
  - **Transparency** : The process must be visible to everybody. The same understanding of the product and the “facts” is shared.
  - **Review** : We must inspect the artifacts and progress towards the goal.
  - **Adaptation** : If a process is detected that deviates from the acceptable limits, it must be readjusted.
- Elements (components):
  - **Roles** : each person has one or more functions.
  - **Artifacts** : results of the sprint.
  - **Processes** : meetings in each sprint.

Agile Management: SCRUM functionalities



Agile Management: SCRUM characteristics

- The participants in the project have different **roles**:
  - Product owner: vision (connection) of the customer / user
  - Scrum master: coordinates the process
  - Scrum team: working team (executes tasks)
  - Stakeholders: customers, users.
- The requirements are saved as elements (items) in a list called **"product backlog"**:
  - the list is prioritized and the effort is estimated (duration)
  - These elements are called **"user stories"**.
- The product progresses in small increments of fixed time (<1 month), called **"sprints"**:
  - The functionalities to be implemented are selected → **Sprint backlog**
  - A short daily meeting is made (15min) → **Daily scrum meeting**
  - The product is designed, programmed and tested during the sprint

No planning changes are made during the sprint!

SCRUM: facts

Therefore, starting from the theory behind the SCRUM methodology:

- Scrum is not a process or technique; it is an environment where different processes and techniques can be used.
- The scrum rules establish relationships and iterations among the Scrum components.
- Scrum is light, easy to understand… **but very difficult to master!**

Try to understand the concept and apply it in a simple way.

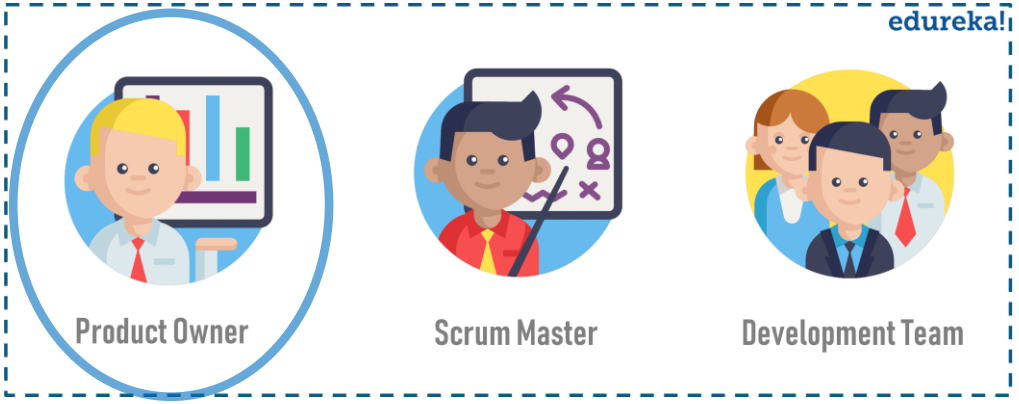
SCRUM: Components

- **Roles:**
  - Product Owner
  - Scrum Master
  - Scrum Team (development team)
  - Others: Stakeholders (parts with interest), users, customers
- Artifacts (elements):
  - Product Backlog (requirements list)
  - Sprint Backlog (selected requirements for the sprint)
  - Progress charts: burn-up, burn-down
- Processes:
  - Sprint planning meeting
  - Sprint (increment o iteration)
  - Daily scrum (sprint tracing)
  - Sprint review meeting
  - Sprint retrospective meeting

SCRUM: Roles

- "Pig" role:
  - Product Owner
  - Scrum Master
  - Development team
- "Chicken" role:
  - Stakeholders
  - Users
  - Customers

SCRUM: 'Product Owner' role



- Represents the **client**:
  - He acts as a single voice
  - He has the vision of the product
- Responsible for requirements:
  - Decide the Product Backlog
  - Change and re-prioritize the product backlog before of each sprint
- Accepts the software

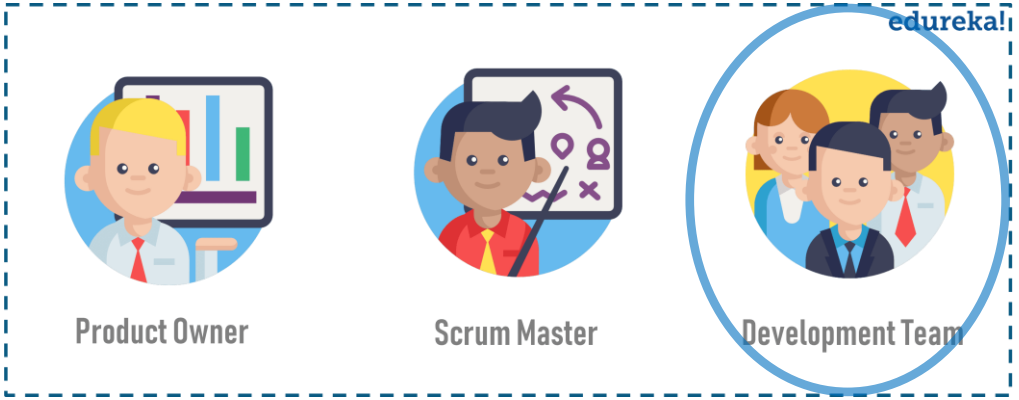
Development team implements what the client wants.

SCRUM: 'Scrum Master' role

- **Manager**, responsible of the process
- Makes easier to make meetings, and monitories the progress
- Helps the team:
  - Remove drawbacks to the team
  - Ensures the productivity and isolates the team from distractions
- He is the connection between the product owner and the scrum team
- Interacts with the rest of the organization

He is the manager, not the boss.

SCRUM: 'Development Team' role



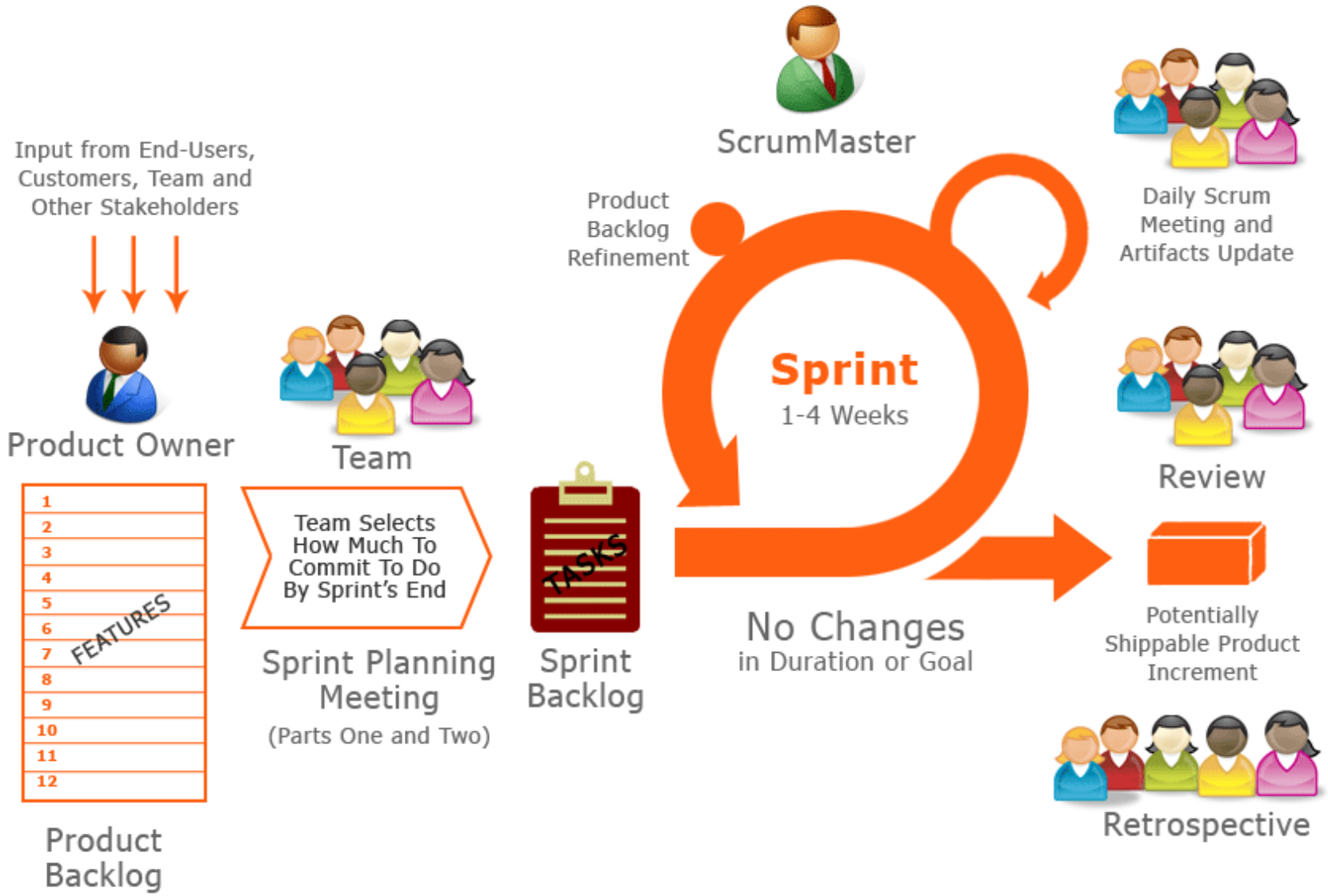
- Scrum team: **3-9** members
- Develops the product
- It is an autonomous team
- Responsible for the commitments
- Multi-functional:
  - each member has expertise
- Self-organized:
  - There is no default role, tasks are distributed taking into account the pending tasks and experience of each person.

Assign a task to someone who can perform it.

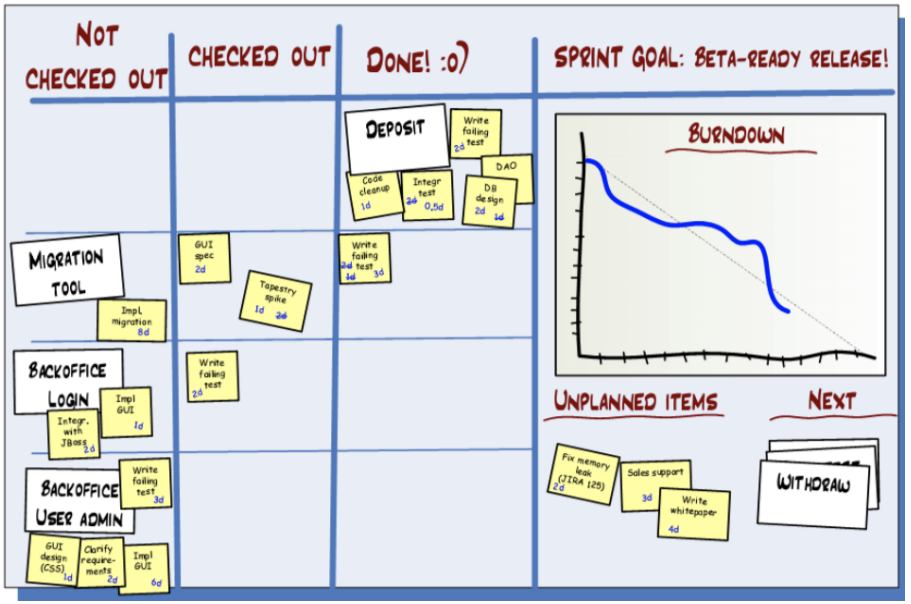
SCRUM: Components

- Roles:
  - Product Owner
  - Scrum Master
  - Scrum Team (development team)
  - Others: Stakeholders (parts with interest), users, management
- **Artifacts (elements):**
  - Product Backlog (requirements list)
  - Sprint Backlog (selected requirements for the sprint)
  - Progress charts: burn-up, burn-down
- Processes:
  - Sprint planning meeting
  - Sprint (increment o iteration)
  - Daily scrum (sprint tracing)
  - Sprint review meeting
  - Sprint retrospective meeting

SCRUM: Artifacts (elements)



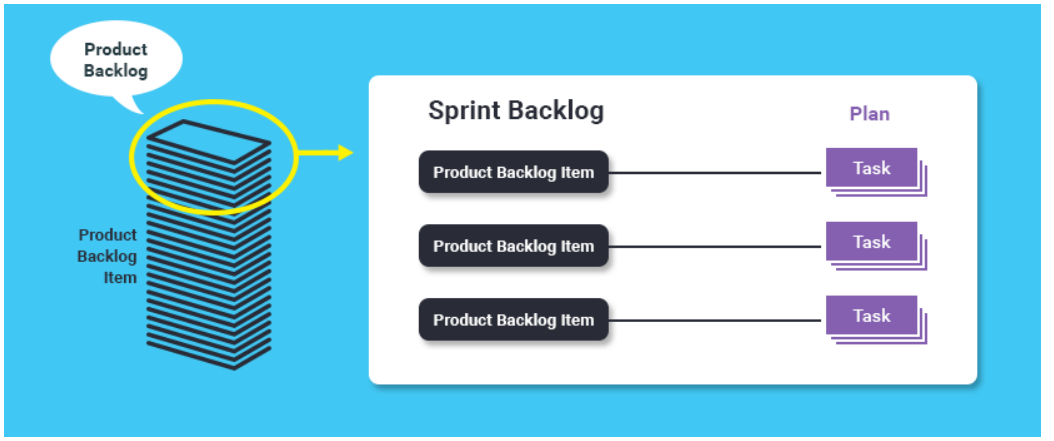
SCRUM: Product Backlog



- It is a **Requirements list**
  - It's an approximation → So, it's not accurate and can change
  - The requirements are prioritized
- Belongs to the Product Owner → s/he is responsible for managing it
  - S/He can change (add/remove, change priorities) before each sprint
- Estimation of the elements of the product backlog:
  - The speed of the team development is estimated.
  - Effort is estimated: hours/days

SCRUM: Sprint Backlog

- **Subset** of the Product Backlog
  - Defines the work that will be done in a sprint
  - If a task is very long, it must split
- It is created only by the Scrum Team
  - The team can add/remove elements to the list
  - Product Owner is not allowed to do anything with this list

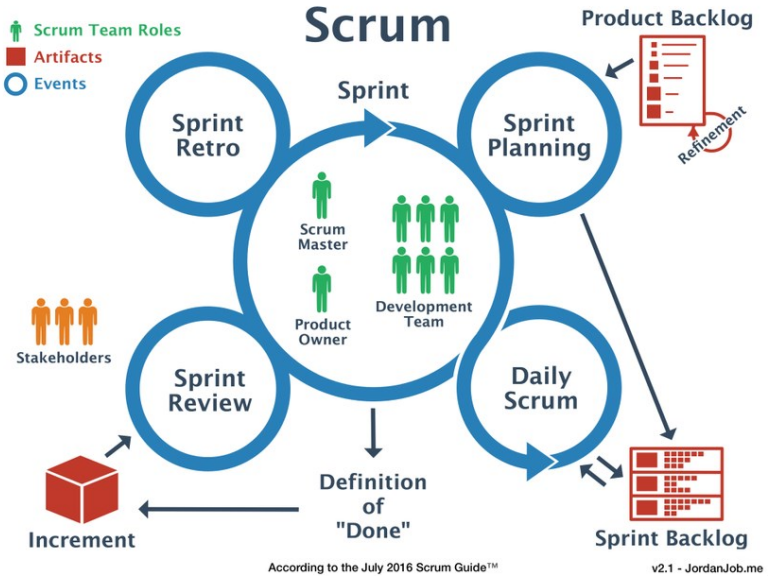


- The elements of the Sprint Backlog have 3 dimensions:
  - Priority (more or less importance)
  - Detail (breakdown of tasks)
  - State (pending, doing, paused, done). It should be updated every day

SCRUM: Components

- Roles:
  - Product Owner
  - Scrum Master
  - Scrum Team (development team)
  - Others: Stakeholders (parts with interest), users, management
- Artifacts (elements):
  - Product Backlog (requirements list)
  - Sprint Backlog (selected requirements for the sprint)
  - Progress charts: burn-up, burn-down
- **Processes:**
  - Sprint planning meeting
  - Sprint (increment o iteration)
  - Daily scrum (sprint tracing)
  - Sprint review meeting
  - Sprint retrospective meeting

SCRUM: Phases



- Project **kick-off meeting**:
  - It is a meeting before starting the project
  - The requirements list is created, some of the requirements are selected and prioritized → the result is the product backlog
- **Sprint planning meeting**:
  - The requirements (from the backlog list) to be developed in this sprint are selected
  - The tasks are also determined in a list: sprint backlog
- Sprint:
  - **Daily sprint meeting**
- At the end of the sprint → **sprint review**:
  - The following sprint is defined
  - The unmade tasks pass to the product backlog
  - It is decided whether to do a product increment or a release
  - At the end of the sprint → **sprint retrospective**

SCRUM: Sprint Planning meeting

- Collaborative Meeting at the start of **each** sprint:
  - Participants: Product Owner, the Scrum Master and Scrum Team
  - It can last up to 8 hours and consists of two parts
  - The requirements and priorities are defined
  - The tasks to be developed in the sprint are defined
- 1st Part:
  - Create the Product Backlog (requirements list)
  - Determine the Sprint Goal (objectives)
  - Participants: Product Owner, Scrum Master, Scrum Team
- 2nd Part:
  - Create Sprint Backlog (subset of requirements to do in the sprint)
  - Participants: Scrum Master, Scrum Team

SCRUM: Sprint

- An iteration or increment, of less than a month, where the functionality of the product is increased
- **No external influence** can interfere with the team during the sprint
- Each sprint starts with the Daily Scrum Meeting

Here (in the sprint) the software development work is done.

SCRUM: Daily Scrum meeting

- It is done at the beginning of the day (maximum 15 minutes)
- Each team member answers 3 questions:
  - What he has been done?
  - What drawbacks he had?
  - What is he going to do today?
- This allows the scrum master to **monitor progress**, and in case of problems:
  - Plan work not identified in previous days
  - Re-assign tasks
  - Eliminate problems
- Everyone is invited → avoid other unnecessary meetings:
  - But only the product owner, scrum master, scrum team can talk
- It is a meeting where team members **make commitments**:
  - It is not a session to solve doubts
  - It is not a way of knowing who does the tasks on time

SCRUM: Sprint Review meeting

- It is done at the end of the sprint
- Sprint review → **Objectives achieved?**
  - The team presents what has been achieved during the sprint
  - The new functionalities of the product are usually presented to the product owner with a demo
- It is informal (less than <2h)
- Participants:
  - Clients
  - Managers
  - Product Owner
  - Other engineers

SCRUM: Sprint Retrospective meeting

- Sprint retrospective meeting:
  - Sprint summary
  - Only participate the Scrum Team
- The estimated speed and the actual **speed are checked** : burn-down
- Feedback from the meeting:
  - What it has been done well?
  - What it should be improved?
  - What it has been improved?

SCRUM: The Task Board

The **Task Board** is a tool to organize and manage the tasks of the backlogs:

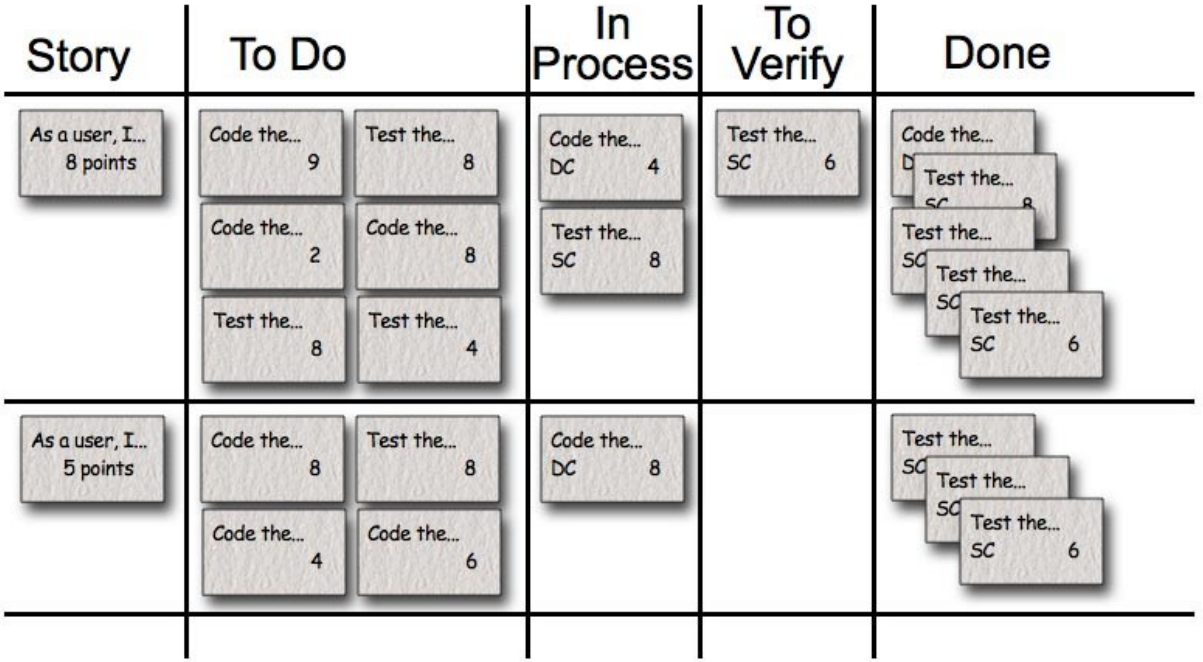
- It is a 2-dimensional matrix with rows representing *User Stories* and columns representing various *Status* values:
  - Tasks to do
  - Tasks in progress
  - Tasks done
- The objective is to provide immediate visibility of development status.

The Task Board is an abstraction used in several different methodologies.

Task Board: Level of detail

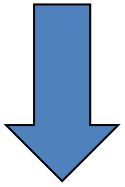
Simple

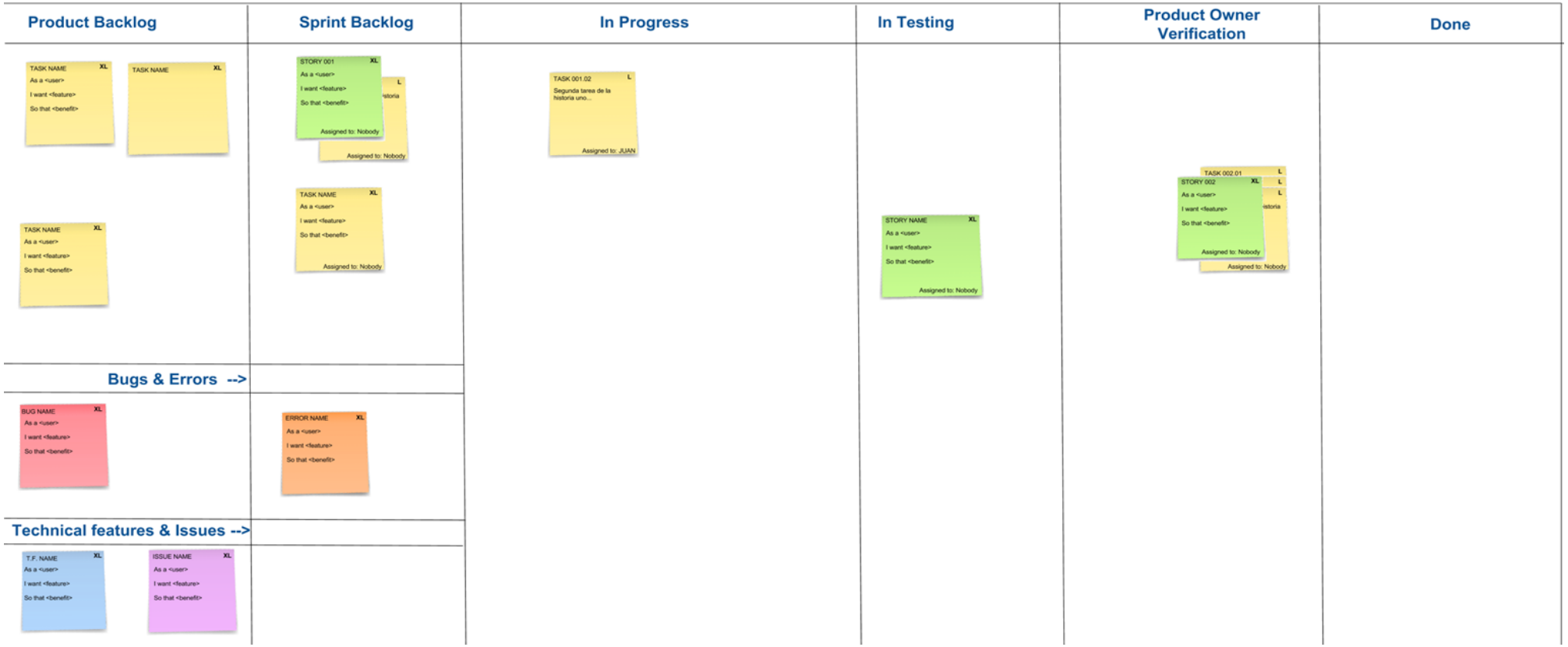
Regular



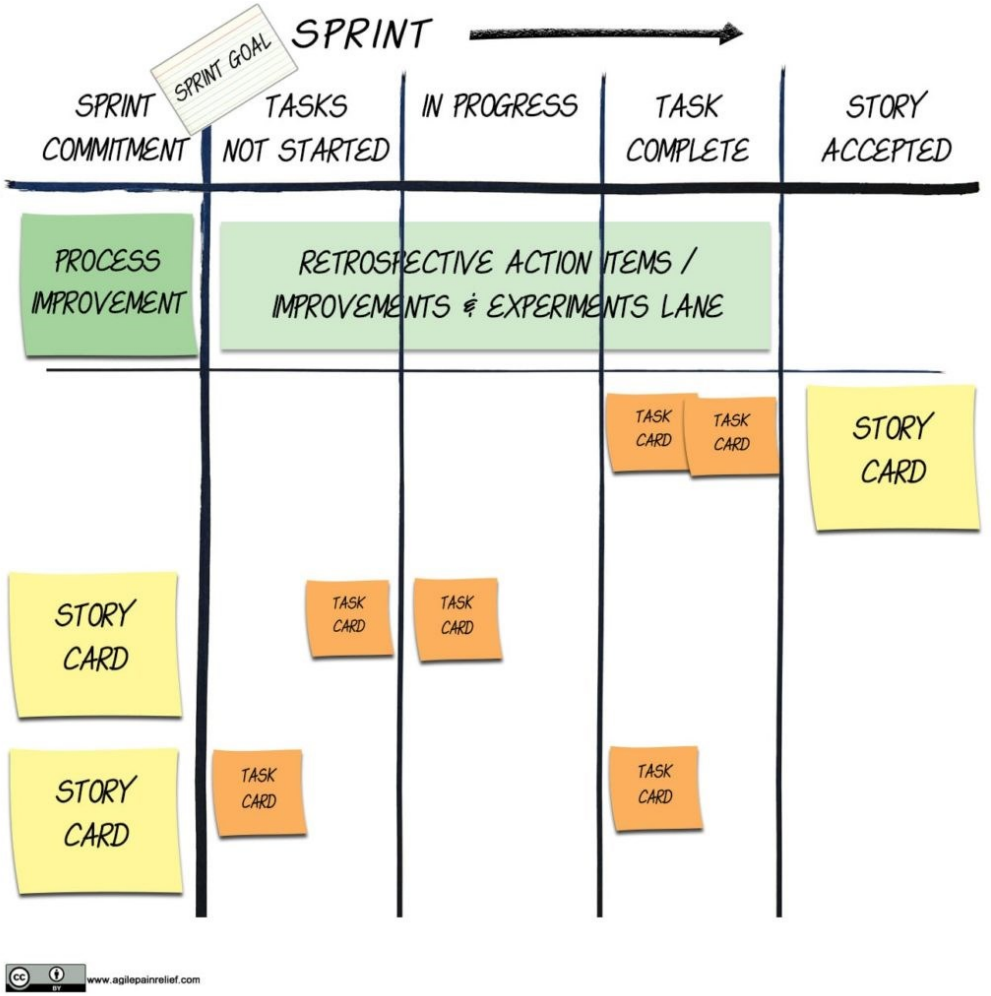
Task Board: Level of detail

Complex





Task Board: as a Tool

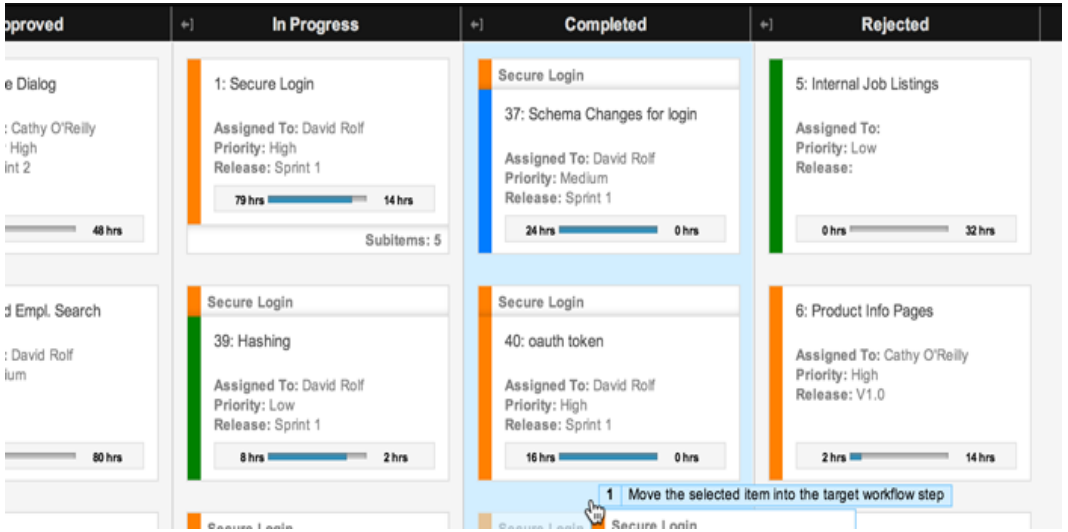


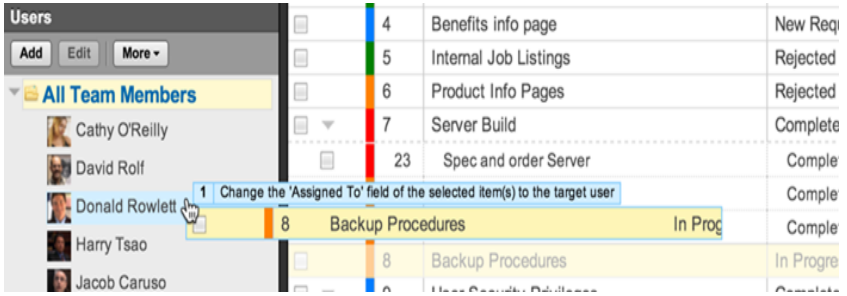
- Useful for **managing** tasks:
  - In addition to the status can store other information: priority, etc.
- Useful for **assigning** tasks:
  - Users have to take responsibility for specific tasks.
  - One person → One task
- Useful for **controlling** tasks:
  - The necessary effort must be allocated.
  - Progress must be monitored.
  - In case of problems, tasks must be reallocated.

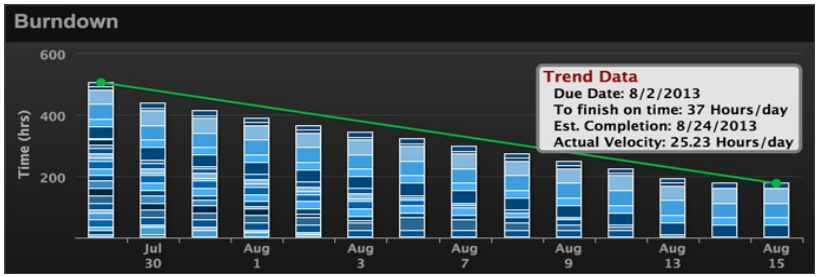
SCRUM: Software tools

To complete the management there are multiple tools that complement the simple Task Board with enhanced functionality to achieve a complete management of resources. However, the Backlog remains the central point in all of them.

- Open source: Kunagi, ScrumDo, SprintoMeter, IceScrum, Trello, …
- Commercial: JIRA Agile, Eylean, OnTime, Azure DevOps, …







CONCLUSIONS

- The repository is a central point for management.
- SCRUM simplifies management with the Task Board and productive meetings.
- The Scrum Team coordinates, communicates, solves problems and improves continuously.

Recommended reading to find out more:
https://www.openproject.org/collaboration-software-features/agile-project-management/