



# Agile Software Development

19/20 November 2019 | Guido Trens (JSC, SimLab Neuroscience)

Motivation

Plan-Driven vs Agile Software Development

Introduction to Scrum

Principles and Practices

Tools

Example Project for Agile Management with GitLab

Small Projects

References

## Motivation

Plan-Driven vs Agile Software Development

Introduction to Scrum

Principles and Practices

Tools

Example Project for Agile Management with GitLab

Small Projects

References

# Motivation

---

- Scientific software does not end its development cycle on publication of the paper.
- Reproducibility of scientific results requires sustainable software.
- Learn from the industry where rapid software development became the standard methodology for developing sustaining complex software, also known as:

## ***“Agile Development” or “Agile Methods”***

- Why is Agile Development such a success story?
  - Agile development accelerates the delivery.  
In contrast: plan-driven software development is a lengthy process
  - Agile methods can handle changing requirements.

# Content

---

Motivation

## **Plan-Driven vs Agile Software Development**

Introduction to Scrum

Principles and Practices

Tools

Example Project for Agile Management with GitLab

Small Projects

References

# Plan-Driven vs Agile Software Development

---

## Plan-driven

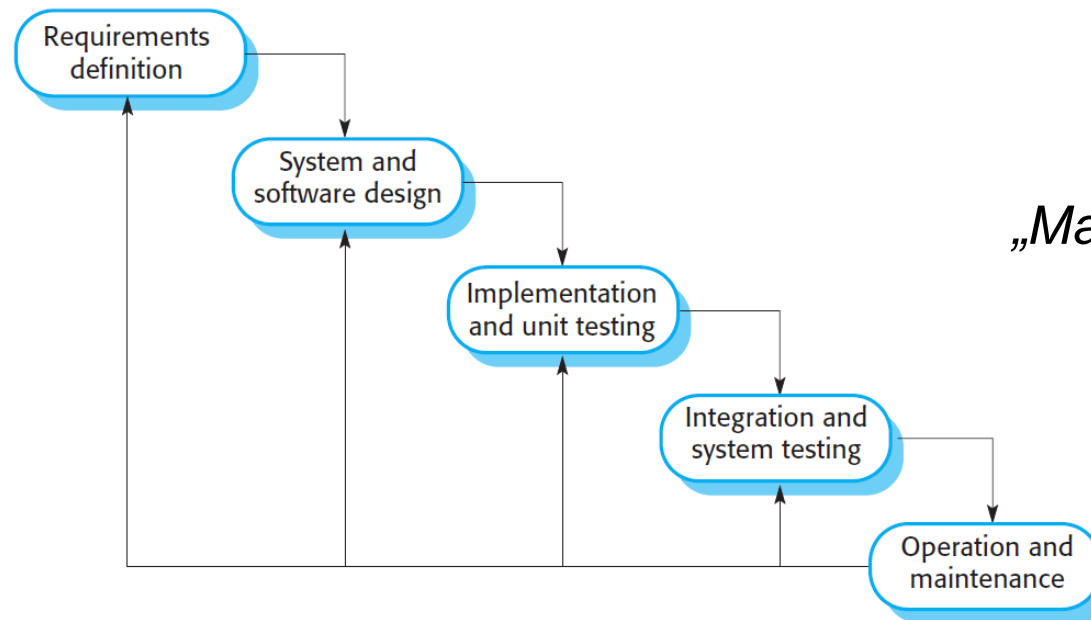
- Also known as “heavy-weight“ or “traditional” methodologies
- Up-front system architecture and detailed plans
- Completely specifies:
  - Requirements
  - Design
  - Build and test environments
- Uses a conventional waterfall or specification-based software development process

# Plan-Driven vs Agile Software Development

## Plan-driven

- Waterfall model

*One stage must be completed before progress to the next stage is possible!*



*„Make a plan and do not change it!“*

- Plan-driven software development is still applicable for some types of software, e.g. safety-critical systems.

# Plan-Driven vs Agile Software Development

---

## Agile Development

- The need for rapid software development and processes has been recognized for many years.
- The idea of “**Agile Methods**“ took off in the late 90’s.

- eXtreme Programming (XP) [1999 Kent Beck]

*The approach was developed by **pushing recognized good practice**, such as iterative development, to “**extreme**” levels.*

*For example: In XP, several new versions of a system may be developed by different programmers, integrated, and tested in a day.*

- Dynamic System Development Method (DSDM)

*Is a generic approach to **project management and solution delivery** rather than being focused on software development.*



# Plan-Driven vs Agile Software Development

---

## What is Agile Software Development?

It is ..

- a methodology, a set of methods and practices, a way of executing software development management
- iterative
  - *Iteration is the main concept in agile. (All agile methods are iterative!)*
  - *It is the total opposite of the waterfall-model!*
  - *The work is done in tight cycles, so called “sprints“.*
  - *The “plan“ is **constantly revisited**.*
- streamlined
  - *It favors for getting the work done.*
- time-boxed
  - *The work is planed by time instead of by feature.*
- very collaborative

[ Dave Hecker, <https://www.youtube.com/watch?v=-zDct5d2smY> ]

# Plan-Driven vs Agile Software Development

---

## Agile Methods and Processes

- eXtreme Programming (XP)
- Scrum
- Large-scale Scrum (LS Scrum)
- Kanban
- ...

.... based on practices like:

- Test-driven development (TD)
- User acceptance tests
- Pair-programming
- Refactoring
- Continuous integration and delivery (CI/CD)
- Following coding standards
- ...

# Plan-Driven vs Agile Software Development

---

Plan-Driven

Agile

Make a plan and do  
not change it!



Constantly revisit the plan!

**Agile methods are designed to  
produce useful software quickly!**

# Content

---

Motivation

Plan-Driven vs Agile Software Development

**Introduction to Scrum**

Principles and Practices

Tools

Example Project for Agile Management with GitLab

Small Projects

References

# Introduction to Scrum

---

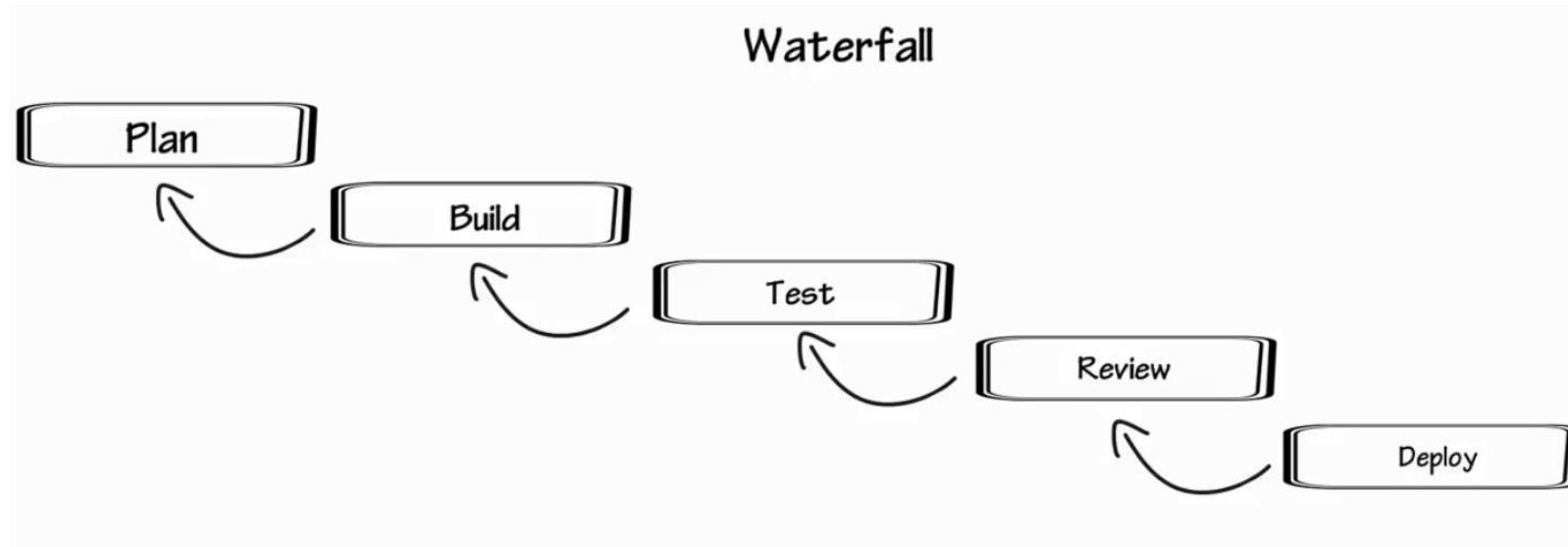
## What is Scrum?

- Scrum is an **agile method** offering a **lightweight project management framework** for effective team collaboration.
- The *Scrum methodology* was first public presented in 1995 by *Jeff Sutherland* and *Ken Schwaber* at the OOPSLA conference.

In the sport of rugby, a *Scrum* is a way of restarting the game, when the ball has gone out of play and 7-8 players work to move the ball forward.

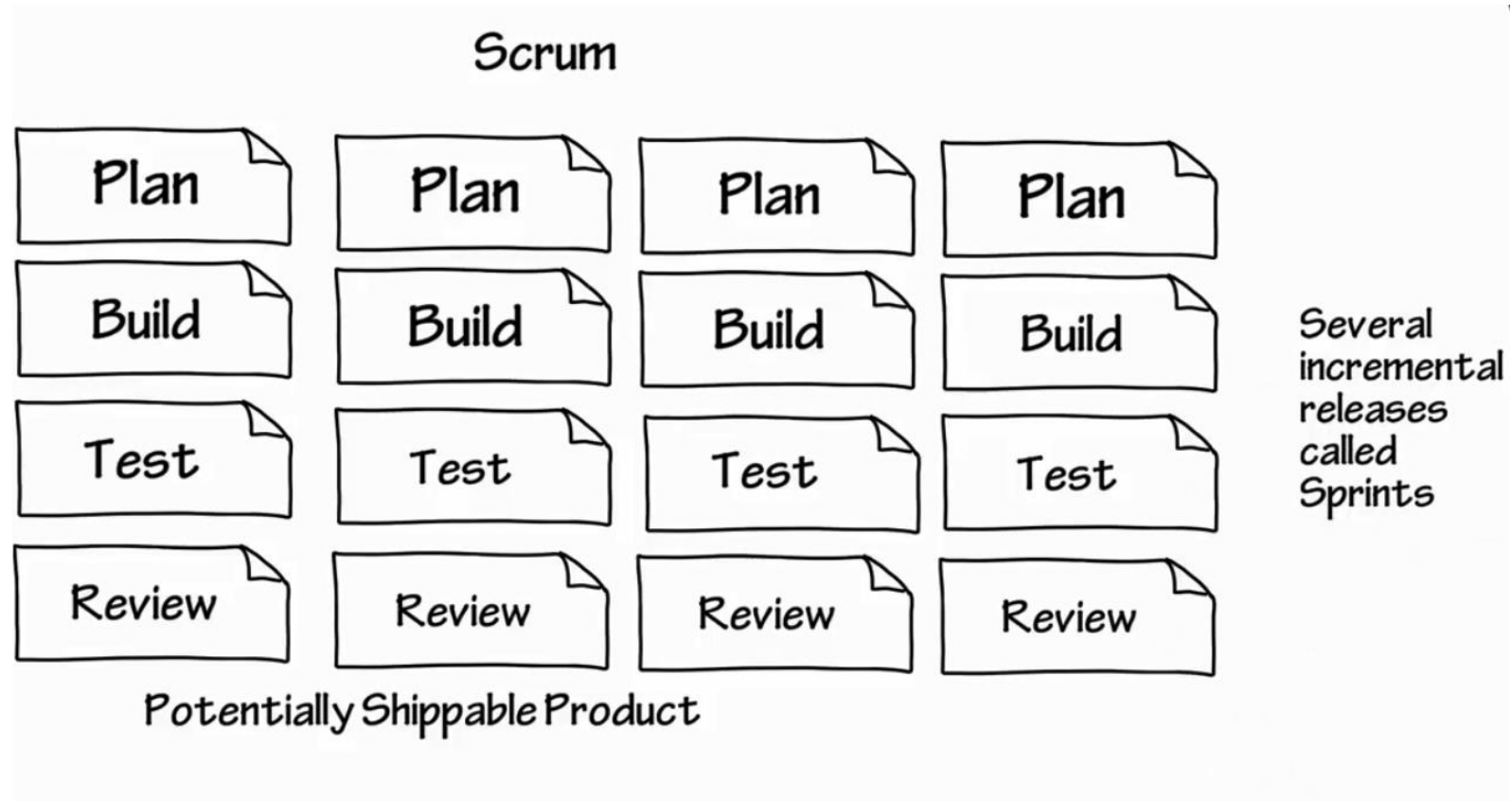


# Agile Development – Introduction to Scrum



[ Steve Stedman, <https://www.youtube.com/watch?v=9TyclROTqFA> ]

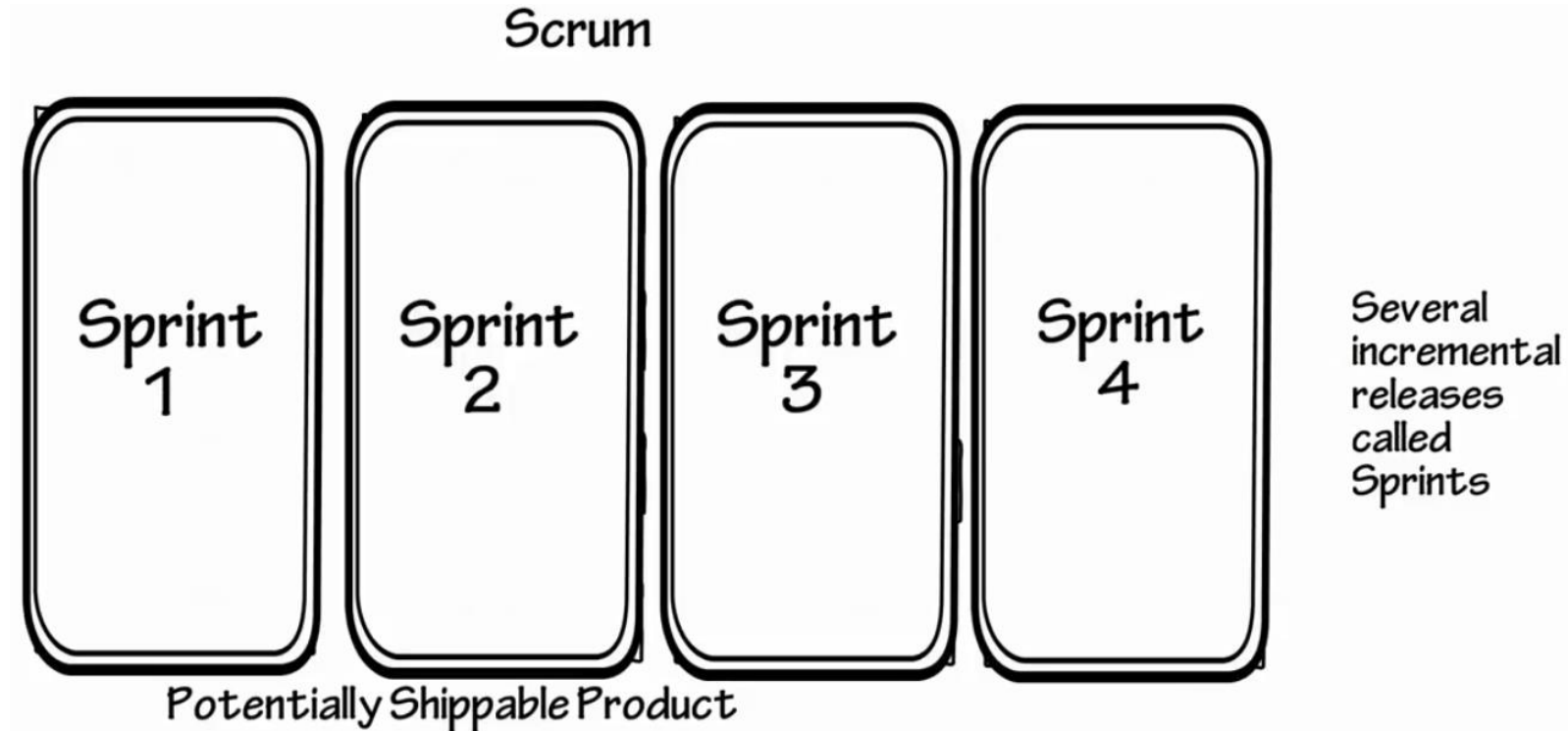
# Agile Development – Introduction to Scrum



[ Steve Stedman, <https://www.youtube.com/watch?v=9TyclROTqFA> ]

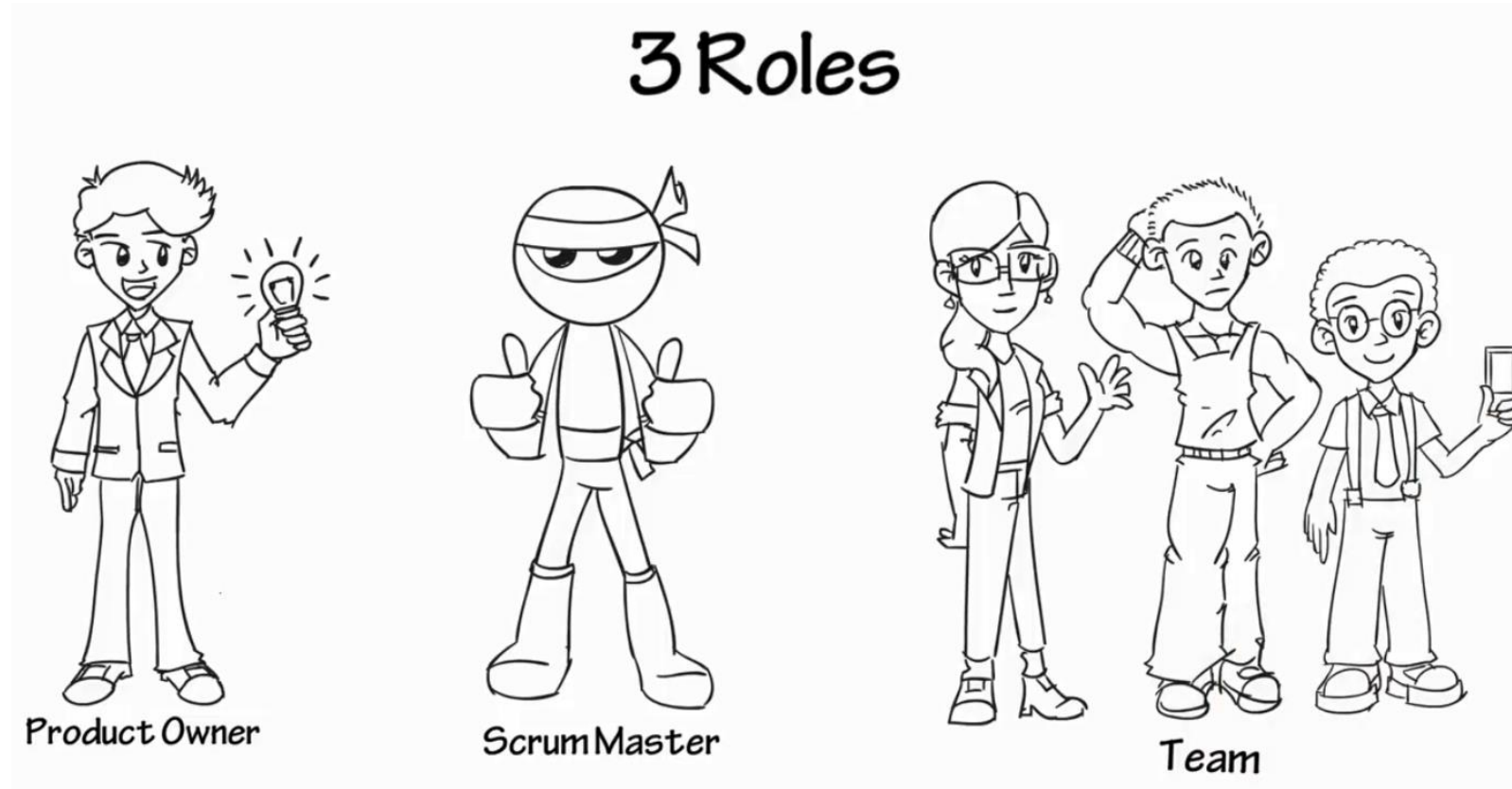


# Agile Development – Introduction to Scrum



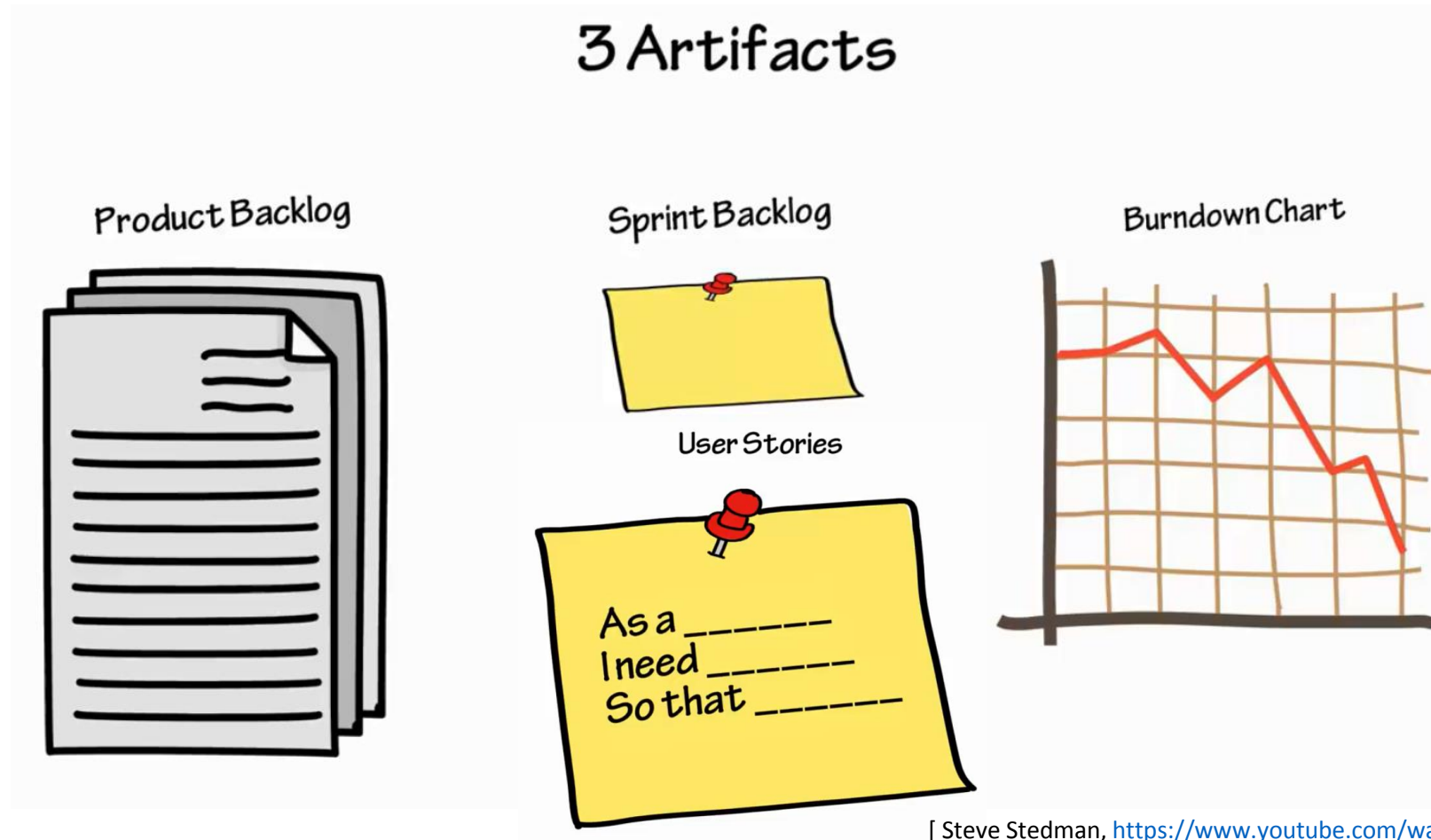
[ Steve Stedman, <https://www.youtube.com/watch?v=9TyclROTqFA> ]

# Agile Development – Introduction to Scrum



[ Steve Stedman, <https://www.youtube.com/watch?v=9TyclR0TqFA> ]

# Agile Development – Introduction to Scrum



# Agile Development – Introduction to Scrum

## 3 Ceremonies

Sprint Planning



Daily Scrum

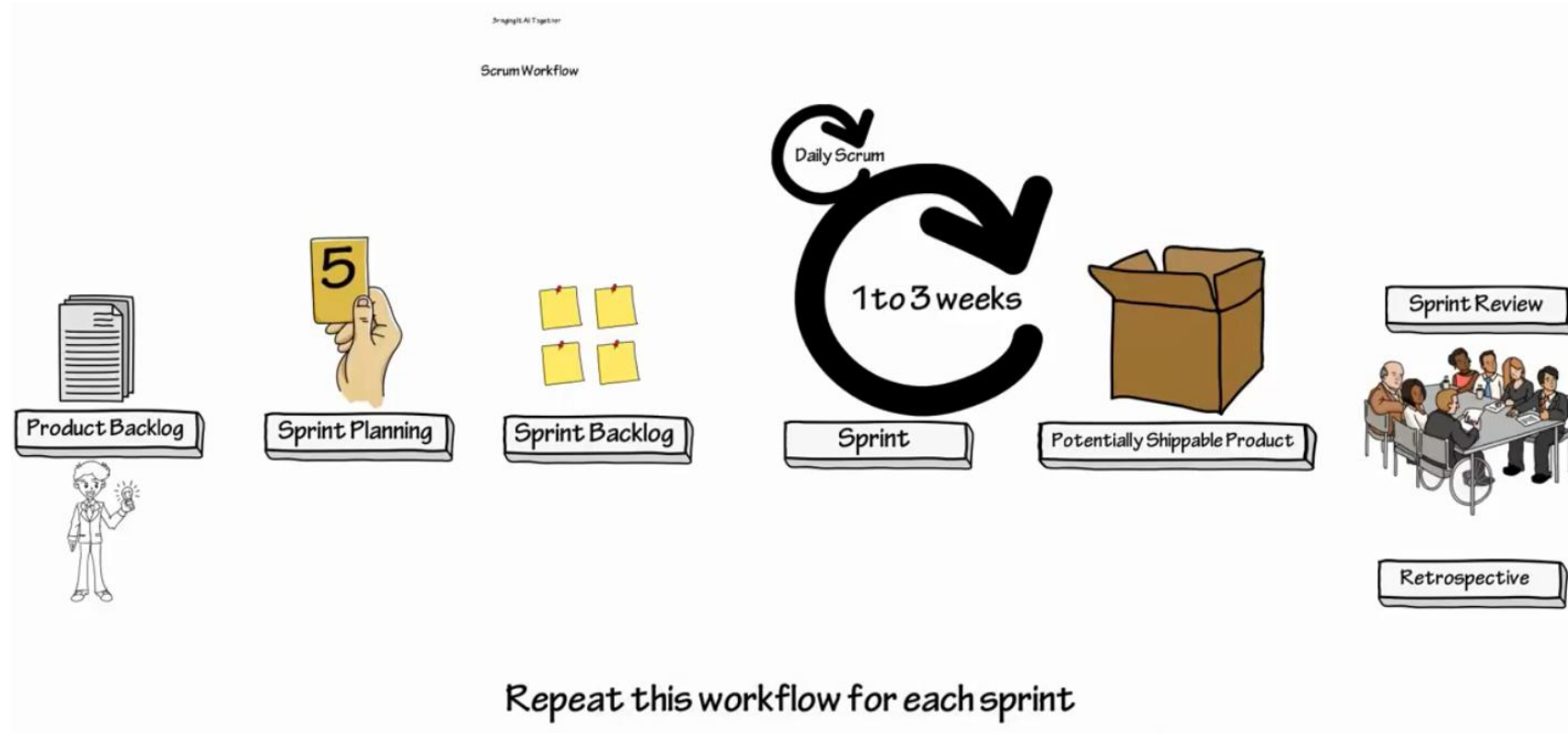


Sprint Review



[ Steve Stedman, <https://www.youtube.com/watch?v=9TyclROTqFA> ]

# Agile Development – Introduction to Scrum



[ Steve Stedman, <https://www.youtube.com/watch?v=9TyclR0TqFA> ]

# Introduction to Scrum

---

## Scrum Team

- Product Owner
- Scrum Master
- Development Team

## Events

- Sprint Planning
- Daily Scrum (Daily Stand Up)
- Sprint Review
- Sprint Retrospective

## Artifacts

- Product Backlog
- Sprint Backlog
- Sprint Progress

# Introduction to Scrum

---

## Scrum Team

- Product Owner
- Scrum Master
- Development Team

## Events

- Sprint Planning
- Daily Scrum (Daily Stand Up)
- Sprint Review
- Sprint Retrospective

## Artifacts

- Product Backlog
- Sprint Backlog
- Sprint Progress



- **One person**, not a committee!
- The Product Owner is responsible for **managing the backlog** to achieve the desired outcome.
  - Clearly identifies and describes product backlog items.
  - Makes decisions regarding the priority of product backlog items.
  - Ensures transparency.

# Introduction to Scrum

---

## Scrum Team

- Product Owner
- **Scrum Master**
- Development Team



## Events

- Sprint Planning
- Daily Scrum (Daily Stand Up)
- Sprint Review
- Sprint Retrospective

## Artifacts

- Product Backlog
- Sprint Backlog
- Sprint Progress

- He or she **guides the team** in the effective use of Scrum and **protects the team** from outside interruptions and distractions.
- The Scrum master is responsible for ensuring the team follows the processes and practices that the team agreed they would use.
- The Scrum master serves the Product Owner and the development team, facilitates Scrum events as requested or needed and **moderates the daily stand up.**



# Introduction to Scrum

---

## Scrum Team

- Product Owner
- Scrum Master
- Development Team

## Events

- Sprint Planning
- Daily Scrum (Daily Stand Up)
- Sprint Review
- Sprint Retrospective

## Artifacts

- Product Backlog
- Sprint Backlog
- Sprint Progress



- They are a **self-organizing team** and manage their own work.
  - No one, not even the Scrum master, tells the development team how to turn the backlog into increments of potentially releasable functionality.
- Development **team size ~ 3 - 9**: Small enough to remain nimble, large enough to complete significant work within a sprint.

# Introduction to Scrum

---

## Scrum Team

- Product Owner
- Scrum Master
- Development Team

## Events

- Sprint Planning
- Daily Scrum (Daily Stand Up)
- Sprint Review
- Sprint Retrospective

## Artifacts

- Product Backlog
- Sprint Backlog
- Sprint Progress

- In this meeting, the entire Scrum **team plans** the work for the **next sprint**.
- The meeting is **time-boxed** to a **maximum of eight hours** for a four-week sprint.
- The work is selected from the backlog.

# Introduction to Scrum

---

## Scrum Team

- Product Owner
- Scrum Master
- Development Team

## Events

- Sprint Planning
- **Daily Scrum (Daily Stand Up)**
- Sprint Review
- Sprint Retrospective

## Artifacts

- Product Backlog
- Sprint Backlog
- Sprint Progress

- It is **time-boxed** meeting, **max. 15 minutes**, for the development team to synchronize.
- What did I do yesterday?
- What will I do today?
- Do I see any impediment that prevents me or the team from reaching the sprint goal.
- Moderated by the Scrum master.

# Introduction to Scrum

---

## Scrum Team

- Product Owner
- Scrum Master
- Development Team

## Events

- Sprint Planning
- Daily Scrum (Daily Stand Up)
- **Sprint Review**
- Sprint Retrospective

## Artifacts

- Product Backlog
- Sprint Backlog
- Sprint Progress

- This is an informal **four-hour time-boxed** meeting (for a four-week sprint) at the end of a sprint.
- The Scrum team and the stake holders collaborate about **what was done in the sprint** and **adapt the product backlog** if needed.

# Introduction to Scrum

---

## Scrum Team

- Product Owner
- Scrum Master
- Development Team

## Events

- Sprint Planning
- Daily Scrum (Daily Stand Up)
- Sprint Review
- Sprint Retrospective

## Artifacts

- Product Backlog
- Sprint Backlog
- Sprint Progress

- From experience, this is the most important event !
- This is a **three-hour time-boxed** meeting which occurs after the sprint review and prior the next sprint planning.
- During the retrospective, the Scrum team inspects how the last sprint went with regards to processes, tools, etc.
- The team creates a **plan for improvements**.
- **Eliminate waste !**

# Introduction to Scrum

---

## Scrum Team

- Product Owner
- Scrum Master
- Development Team

## Events

- Sprint Planning
- Daily Scrum (Daily Stand Up)
- Sprint Review
- Sprint Retrospective

## Artifacts

- Product Backlog
- Sprint Backlog
- Sprint Progress

- The Product Backlog is a list of ToDo items, e.g.:
  - features definitions
  - architecture definitions
  - user stories
  - supplementary tasks
  - user documentation tasks
  - .. etc.

# Introduction to Scrum

## Scrum Team

- Product Owner
- Scrum Master
- Development Team

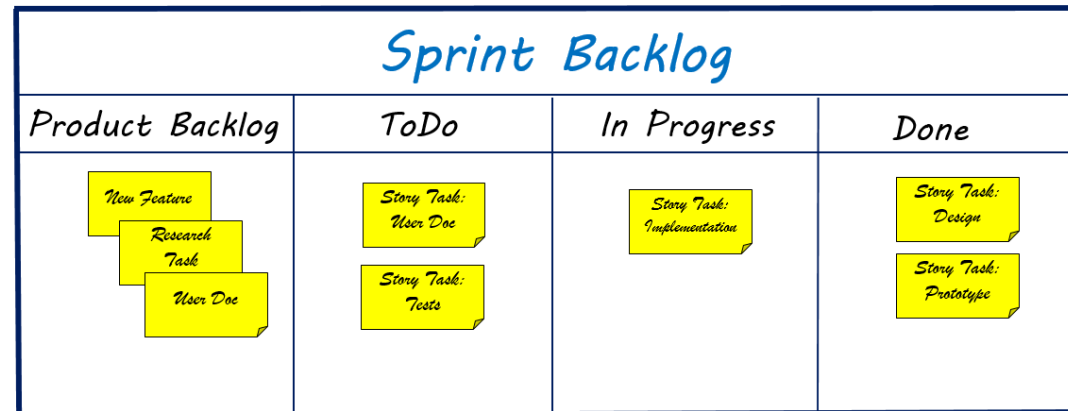
## Events

- Sprint Planning
- Daily Scrum (Daily Stand Up)
- Sprint Review
- Sprint Retrospective

## Artifacts

- Product Backlog
- **Sprint Backlog**
- Sprint Progress

- The Sprint Backlog is a set of backlog items, selected for the sprint.



# Introduction to Scrum

## Scrum Team

- Product Owner
- Scrum Master
- Development Team

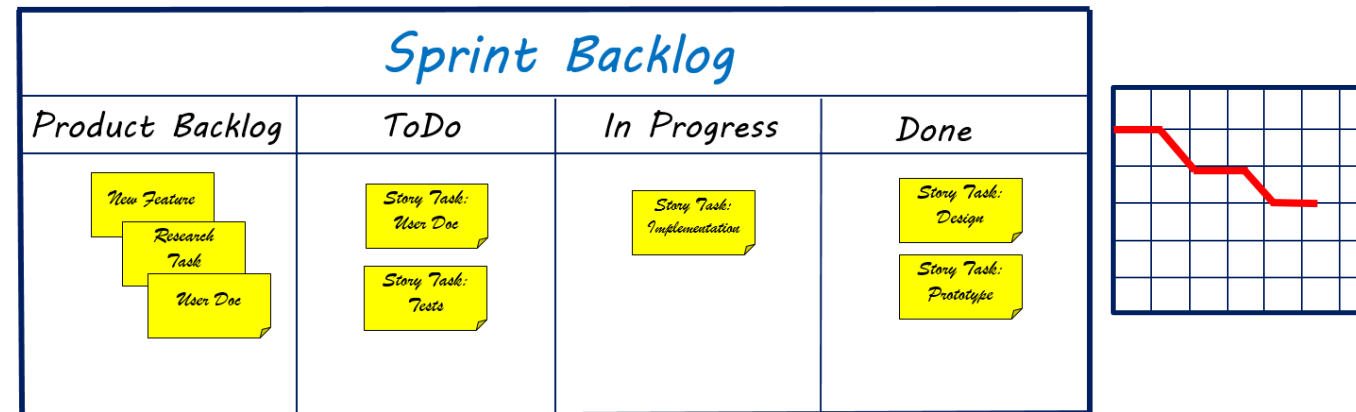
## Events

- Sprint Planning
- Daily Scrum (Daily Stand Up)
- Sprint Review
- Sprint Retrospective

## Artifacts

- Product Backlog
- Sprint Backlog
- Sprint Progress

- Usually a burn-down-chart.





# Introduction to Scrum

---

## “Definition of Done“

- To ensure transparency, Scrum team members must have a **shared understanding of what it means for a task to be completed**, e.g.:
  - source code peer-reviewed
  - documentation adapted
  - test case provided
- As Scrum teams mature, the “Definition of Done“ will expand to include **more stringent criteria for higher quality**.
- This guides the team in knowing how many product backlog items can be selected during sprint planning.

**Any product should have a “Definition of Done“.**

# Introduction to Scrum

---

## Scrum Myths: There is no planning

- In reality there is a lot of planning in Scrum.
- In Scrum, we emphasize the activity of planning over the plan itself.
- Planning is collaborative.
- Planning is part of every event.
- The people doing the work own the plan.
- The way planning is done is to **eliminate waste** !

[ <https://www.Scrum.org/resources/blog/Scrum-myths-there-no-planning-Scrum> ]

# Introduction to Scrum

---

## **Scrum Smells: Signs that something may be amiss on a Scrum project**

- Not all Scrum team members attend the daily Scrum meeting.
- Too much discussion in the Scrum meeting.
- Scrum master assigns work.
- The daily Scrum is for the Scrum master.
- The project team has highly specialized job roles.
- Wild fluctuations shown on a team's initial sprint burndown charts continue to be seen in much later sprints.

[\[https://www.mountangoatsoftware.com/articles/toward-a-catalog-of-Scrum-smells\]](https://www.mountangoatsoftware.com/articles/toward-a-catalog-of-Scrum-smells)

# Introduction to Scrum

---

## Summary

- Scrum is simple to understand but difficult to master.
- Scrum is not restricted to software development.
- Artifacts defined by Scrum are specifically designed to maximize transparency.
- Scrum functions well as a container for other techniques, methodologies and practices.

**Scrum does not solve problems but makes them visible!**

Motivation

Plan-Driven vs Agile Software Development

Introduction to Scrum

**Principles and Practices**

Tools

Example Project for Agile Management with GitLab

Small Projects

References

# Principles and Practices

---

**Scrum as a method works as container for agile development techniques.**

- Collective ownership
- Continuous integration
- Incremental planning
- Pair programming
- Refactoring
- Test first development



# Principles and Practices

---

- Collective ownership
  - Continuous integration
  - Incremental planning
  - Pair programming
  - Refactoring
  - Test first development
- Developers work on all areas of the system
  - No islands of expertise develop
  - All the developers take responsibility for all of the code
  - Anyone can change anything

# Principles and Practices

---

- Collective ownership
  - **Continuous integration**
  - Incremental planning
  - Pair programming
  - Refactoring
  - Test first development
- As soon as the work on a task is complete, it is integrated into the whole system.
  - After any such integration, all the unit tests in the system must pass.

# Principles and Practices

---

- Collective ownership
  - Continuous integration
  - **Incremental planning**
  - Pair programming
  - Refactoring
  - Test first development
- Requirements are recorded on “story cards”
  - The stories to be included in a release are determined by:
    - the time available
    - their relative priority

# Principles and Practices

---

- Collective ownership
  - Continuous integration
  - Incremental planning
  - Pair programming
  - Refactoring
  - Test first development
- Developers work in pairs
  - Checking each other's work
  - Providing support
  - Knowledge transfer

# Principles and Practices

---

- Collective ownership
  - Continuous integration
  - Incremental planning
  - Pair programming
  - Refactoring
  - Test first development
- All developers are expected to refactor the code continuously as soon as potential code improvements are found.
  - This keeps the code simple and maintainable.

# Principles and Practices

---

- Collective ownership
  - Continuous integration
  - Incremental planning
  - Pair programming
  - Refactoring
  - Test first development
- An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.

# Content

---

Motivation

Plan-Driven vs Agile Software Development

Introduction to Scrum

Principles and Practices

## **Tools**

Example Project for Agile Management with GitLab

Small Projects

References

# Tools

---

## Agile Management Tools

- There is a vast market of agile management tools.
- They are usually not free of charge for larger projects.
- The functionality differs in a wide range, from simple tracking or dashboard tools to complex workflow management and reporting for large teams and projects.



**Atlassian Jira**



[www.atlassian.com/software/jira](http://www.atlassian.com/software/jira)





## Agile Development Supporting Tools and Platforms

- Modern software development tools and platforms support agile methodologies and workflows:
  - Version control
  - Test-driven development
  - Peer-review
  - Continuous integration, testing and delivery
  - Basic agile management

**GitHub**



[www.github.com](http://www.github.com)



**Travis CI**

[www.travis-ci.com](http://www.travis-ci.com)



**GitLab**

[www.gitlab.com](http://www.gitlab.com)

# Content

---

Motivation

Plan-Driven vs Agile Software Development

Introduction to Scrum

Principles and Practices

Tools

**Example Project for Agile Management with GitLab**

Small Projects

References

# Example Project for Agile Management with GitLab

---

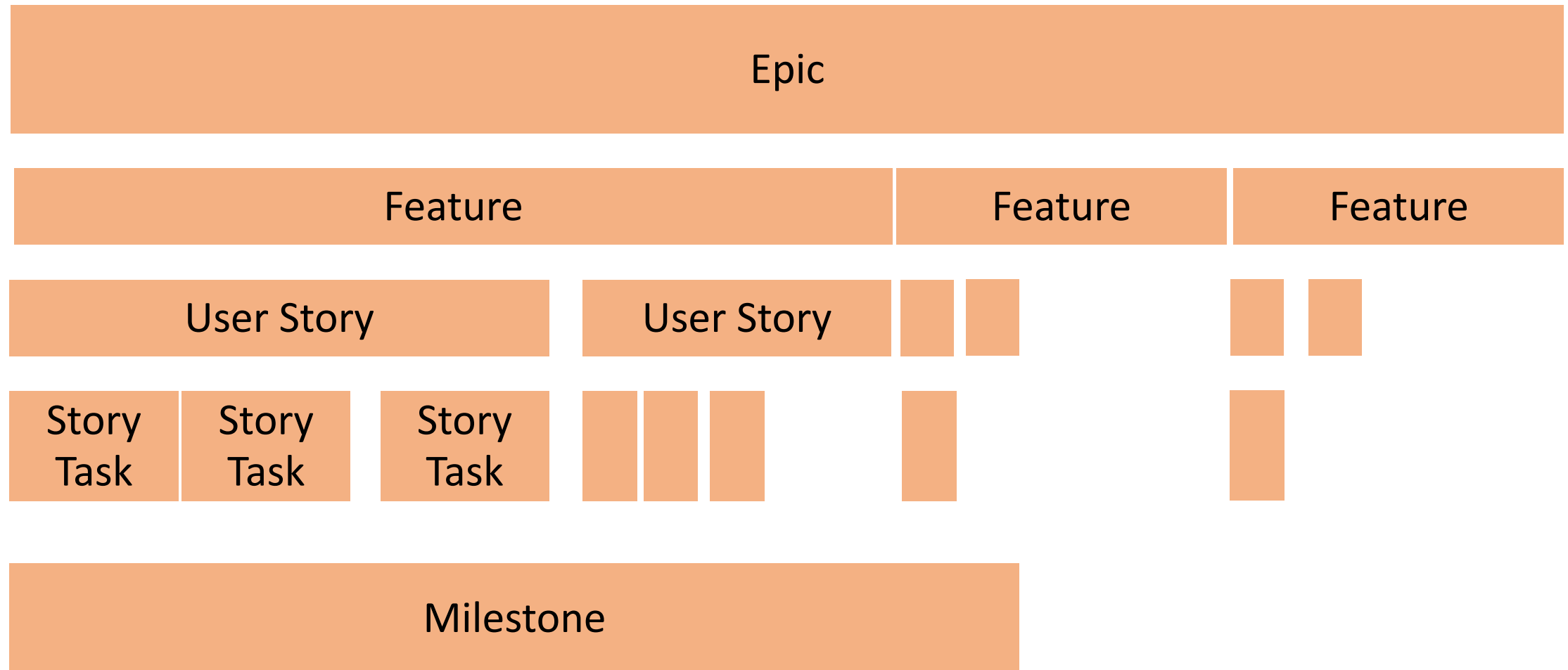
## What is GitLab?



Web-based DevOps (set of software development practices) lifecycle tool:

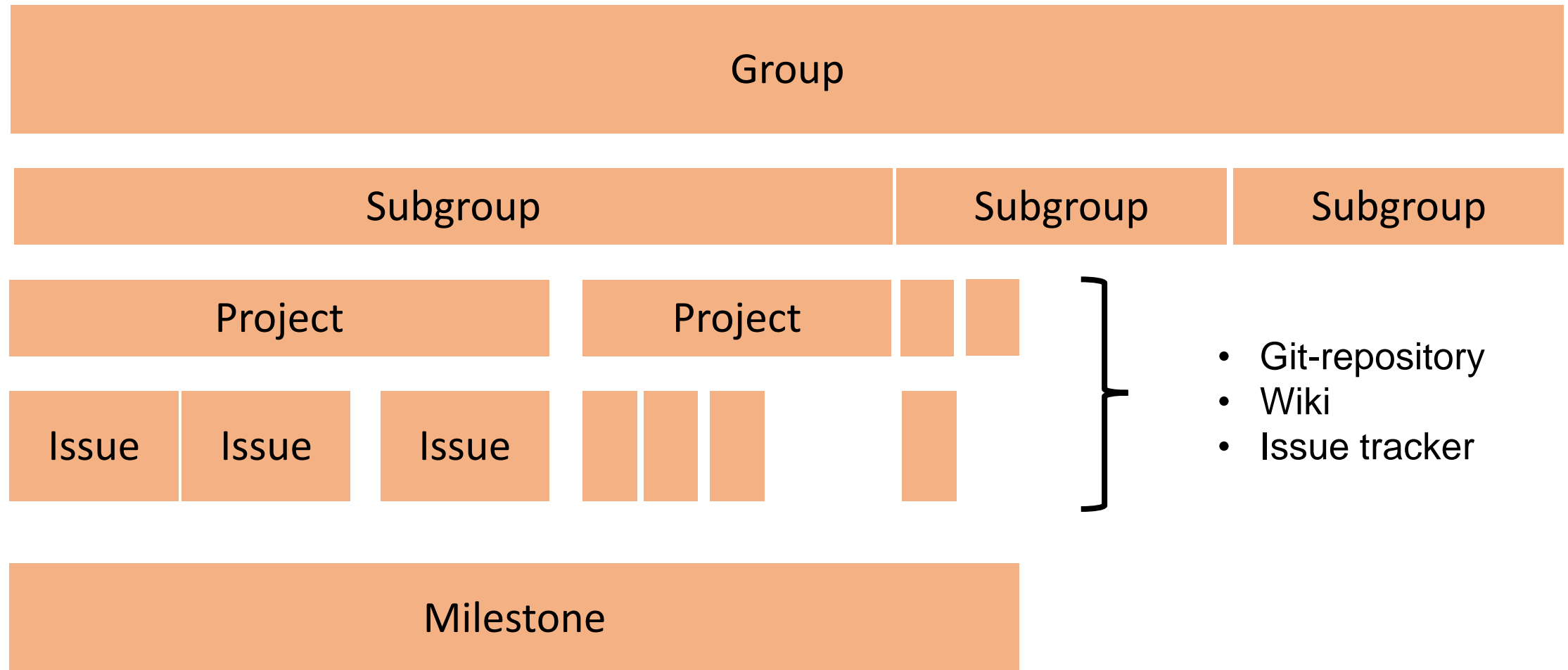
- Git-repository
- Wiki
- Issue-tracker
- CI/CD pipeline
- **Agile software development workflow support**
  - Very basic project management functionality
  - Configurable scrum board
  - Milestones
  - Simple role management
- Community Edition is free of charge

## A Common Project Management Object Hierarchy



# Example Project for Agile Management with GitLab

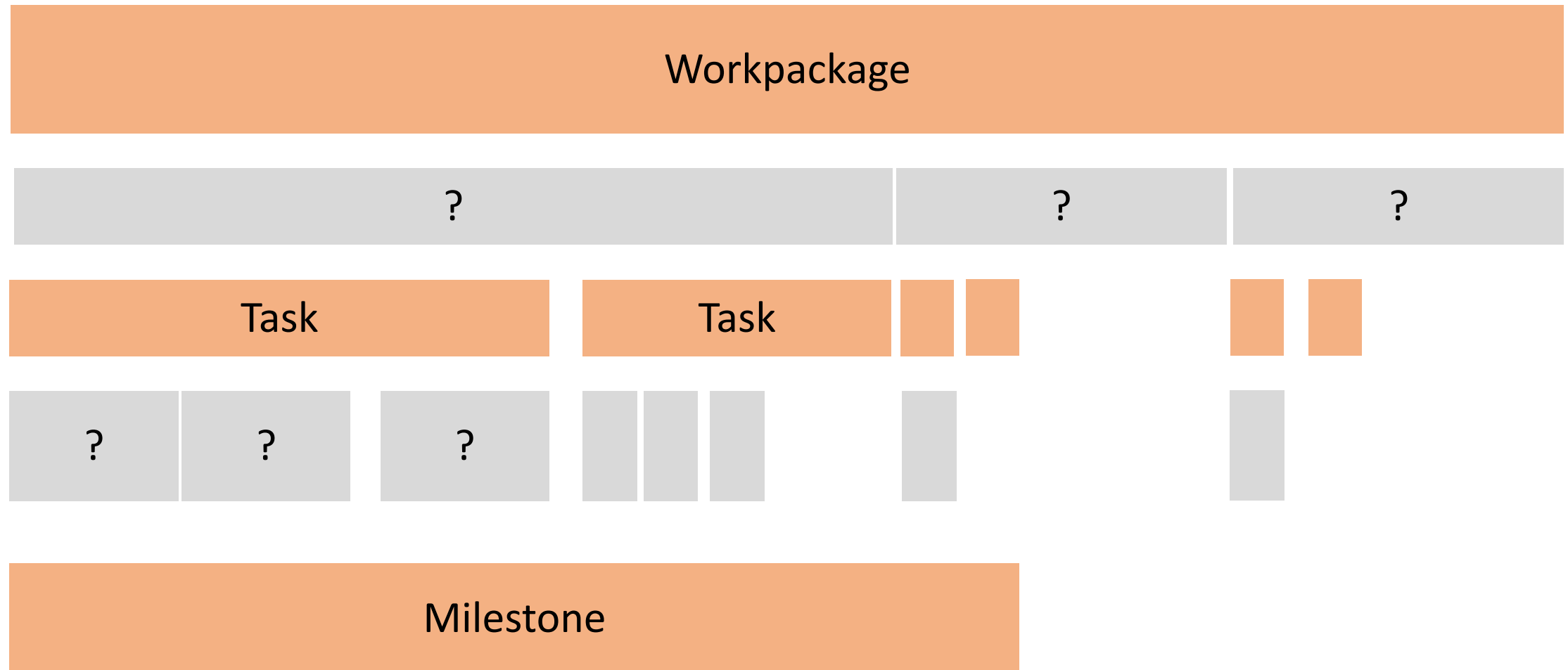
## GitLab provides



- Git-repository
- Wiki
- Issue tracker

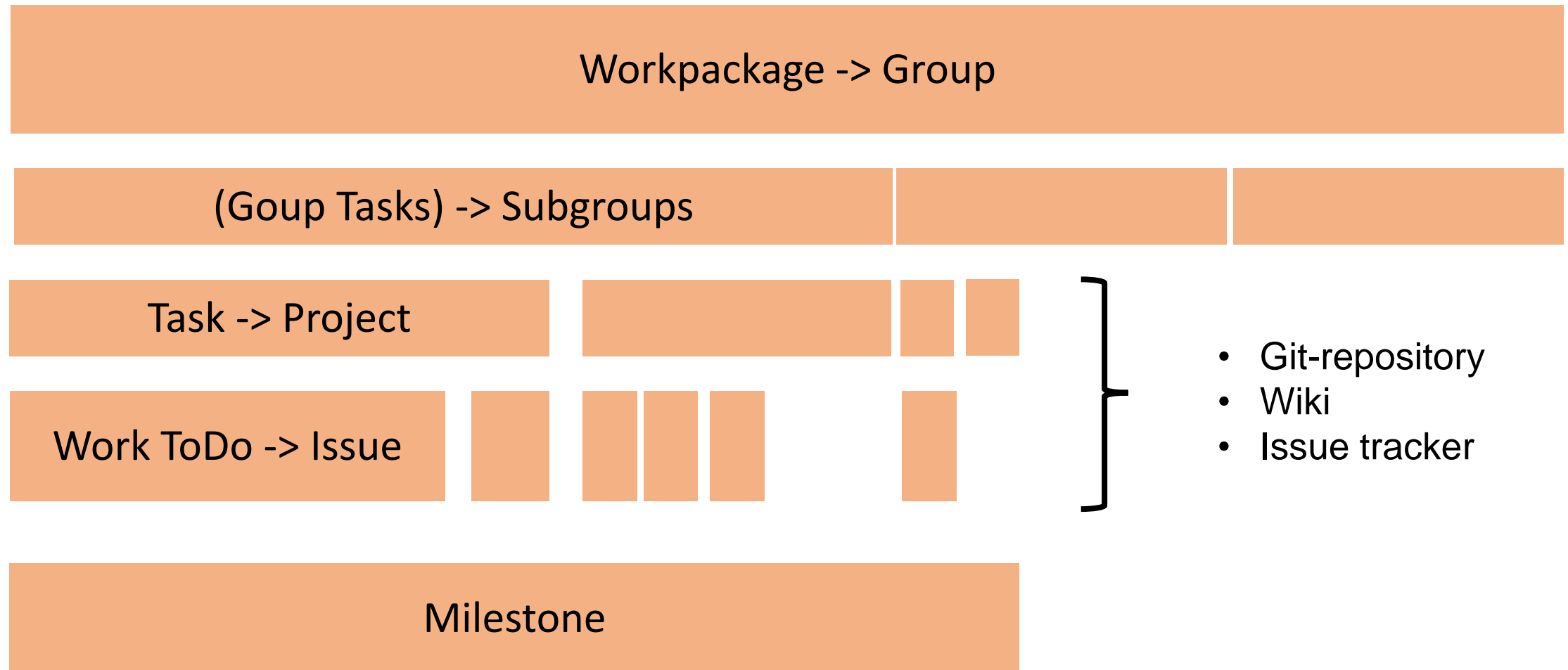
# Example Project for Agile Management with GitLab

„Our Project“ consists of



# Example Project for Agile Management with GitLab

## „Our Project“-objects mapping to GitLab



- Keep all information at one place in common repositories
- Track project (WP) status, milestones
- Simplify the reporting
- Assign work to project members
- Link task dependencies
- Work concurrently in collaboration across teams / organizations
- Maintain the work (software, hardware designs, publications etc.) and their revisions with Git!
  
- FZ Jülich provides two GitLab instances:
  - `gitlab.fz-juelich.de` for internal use
  - `jugit.fz-juelich.de` for collaboration with external partners



## **GitLab Demo**

### **(FZJ Advanced Computing Architectures (ACA) Project)**

# Content

---

Motivation

Plan-Driven vs Agile Software Development

Introduction to Scrum

Principles and Practices

Tools

Example Project for Agile Management with GitLab

**Small Projects**

References

# Small Projects

---

- Small project: about two developers
- A common understanding of software engineering methods and best practices is needed.
- Use a less complex agile approach such as **Kanban** !

It works even if:

- tasks shift on a daily basis, unpredictable, not plannable
- a fixed Scrum-sprint length planning is not possible

Kanban briefly explained:

- visualizes the workflow and uses a **Kanban board** (***ToDo, In Progress, Done***)
- work is prioritized and pulled from backlog when capacity becomes available
- limited number of “In Progress” items

Motivation

Plan-Driven vs Agile Software Development

Introduction to Scrum

Principles and Practices

Tools

Example Project for Agile Management with GitLab

Small Projects

**References**

# References

---

- This book provides a state of the art view of most current thinking about using Scrum.
- It is full of practical advices.



# References

- This book focuses on the technical aspects of agile development, e.g. continuous integration, test driven development, refactoring, pair programming and collective ownership.



# References

---

- [www.agilealliance.org](http://www.agilealliance.org)
- [www.Scrum.org](http://www.Scrum.org)

## The Scrum Guide™

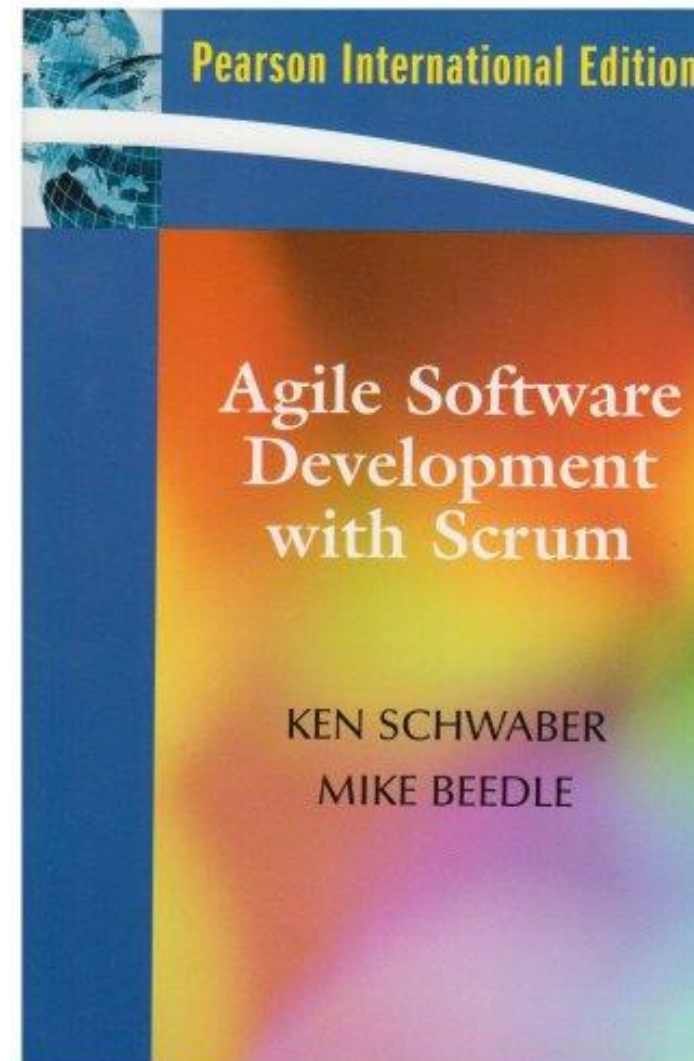
---

The Definitive Guide to Scrum:  
The Rules of the Game



July 2016

*Developed and sustained by Ken Schwaber and Jeff Sutherland*



# References

The possibly most comprehensive book.

<http://iansommerville.com/software-engineering-book/>

	Preface	3
<b>Part 1</b>	<b>Introduction to Software Engineering</b>	<b>15</b>
	Chapter 1 Introduction	17
	Chapter 2 Software processes	43
	Chapter 3 Agile software development	72
	Chapter 4 Requirements engineering	101
	Chapter 5 System modeling	138
	Chapter 6 Architectural design	167
	Chapter 7 Design and implementation	196
	Chapter 8 Software testing	226
	Chapter 9 Software evolution	255
<b>Part 2</b>	<b>System Dependability and Security</b>	<b>283</b>
	Chapter 10 Dependable systems	285
	Chapter 11 Reliability engineering	306
	Chapter 12 Safety engineering	339
	Chapter 13 Security engineering	373
	Chapter 14 Resilience engineering	408
<b>Part 3</b>	<b>Advanced Software Engineering</b>	<b>435</b>
	Chapter 15 Software reuse	437
	Chapter 16 Component-based software engineering	464
	Chapter 17 Distributed software engineering	490
	Chapter 18 Service-oriented software engineering	520
	Chapter 19 Systems engineering	551
	Chapter 20 Systems of systems	580
	Chapter 21 Real-time software engineering	610
<b>Part 4</b>	<b>Software Management</b>	<b>639</b>
	Chapter 22 Project management	641
	Chapter 23 Project planning	667
	Chapter 24 Quality management	700
	Chapter 25 Configuration management	730
	Glossary	757
	Subject index	777
	Author index	803

