Agile Software Development

15 - 17 November 2022 | Guido Trensch (JSC, Simulation & Data Lab Neuroscience)

Member of the Helmholtz Association

Content

Motivation

Plan-Driven vs Agile Software Development

Introduction to Scrum

Principles and Practices

Kanban – Small Projects

Tools

References

Content

Motivation

Plan-Driven vs Agile Software Development

Introduction to Scrum

Principles and Practices

Kanban – Small Projects

Tools

References

Motivation

- Rapid software development became a success story and the standard methodology for developing sustaining complex software in industry, 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.

- Scientific software does not end its development cycle on publication of the paper.
- Reproducibility of scientific results requires sustainable software.

Content

Motivation

Plan-Driven vs Agile Software Development

Introduction to Scrum

Principles and Practices

Kanban – Small Projects

Tools

References

Plan-Driven vs Agile Software Development

Plan-driven

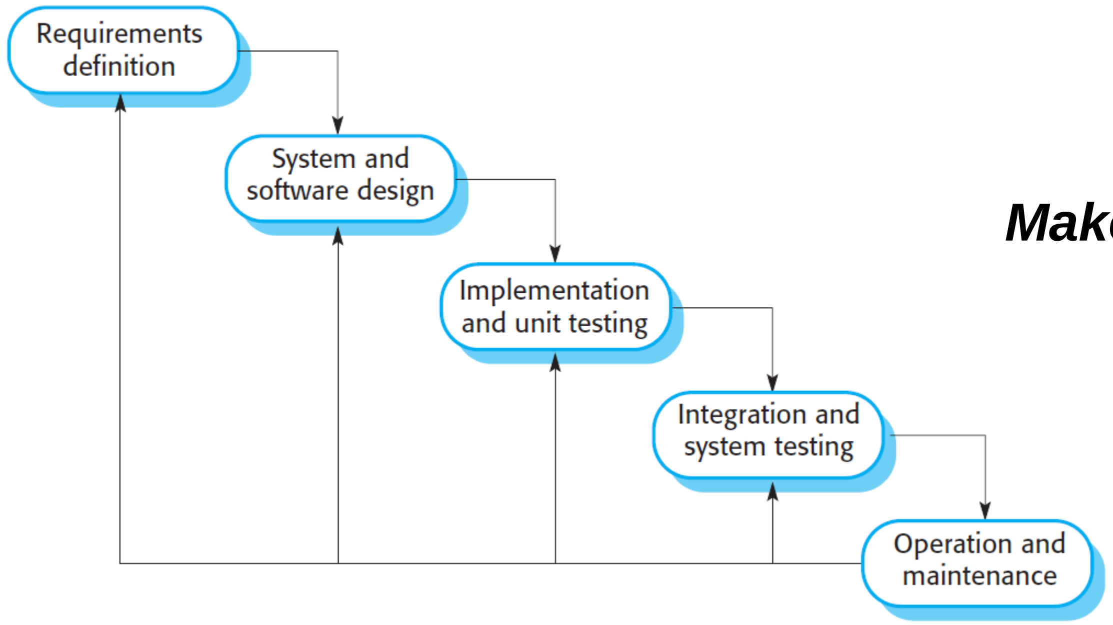
- Also known as “heavy-weight“ or “traditional” methodologies.
- Up-front system architecture and detailed plans.
- Completely specifies:
  - Requirements
  - Design
  - Build and test environments
- Uses a conventional waterfall or specification-based software development process.

Plan-Driven vs Agile Software Development

Plan-driven

- Waterfall model

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



Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

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.

[Ian Sommerville, Software Engineering]

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 with new lightweight methodologies like:

eXtreme Programming (XP) [1999 Kent Beck]

Created in response to the need to solve the problem of changing requirements.

The approach was developed by ***pushing recognized good practice to “extreme” levels****.*

Set of software development practices:

Pair Programming

Continuous Integration

Test-Driven Development

Refactoring

Collective Ownership

[Ian Sommerville, Software Engineering]
[www.agilealliance.org]
[Kent Beck, Extreme Programming Explained]

Plan-Driven vs Agile Software Development

What is Agile Software Development?

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

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**

Plan-Driven vs Agile Software Development

Methods and Processes

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

.. and more

… based on Practices like:

- Pair-programming
- Refactoring
- Following coding standards, clean code
- Test-driven development (TDD)
- User acceptance tests
- Continuous integration and delivery (CI/CD)
- Collective ownership

.. and more

Plan-Driven

Make a plan and do not change it!

Agile

Constantly revisit the plan!

Agile methods are designed to produce useful software quickly!

"Our highest priority is to satisfy the customer through early and continuous delivery of valuable software."

https://agilemanifesto.org

Content

Motivation

Plan-Driven vs Agile Software Development

Introduction to Scrum

Principles and Practices

Kanban – Small Projects

Tools

References

Introduction to Scrum

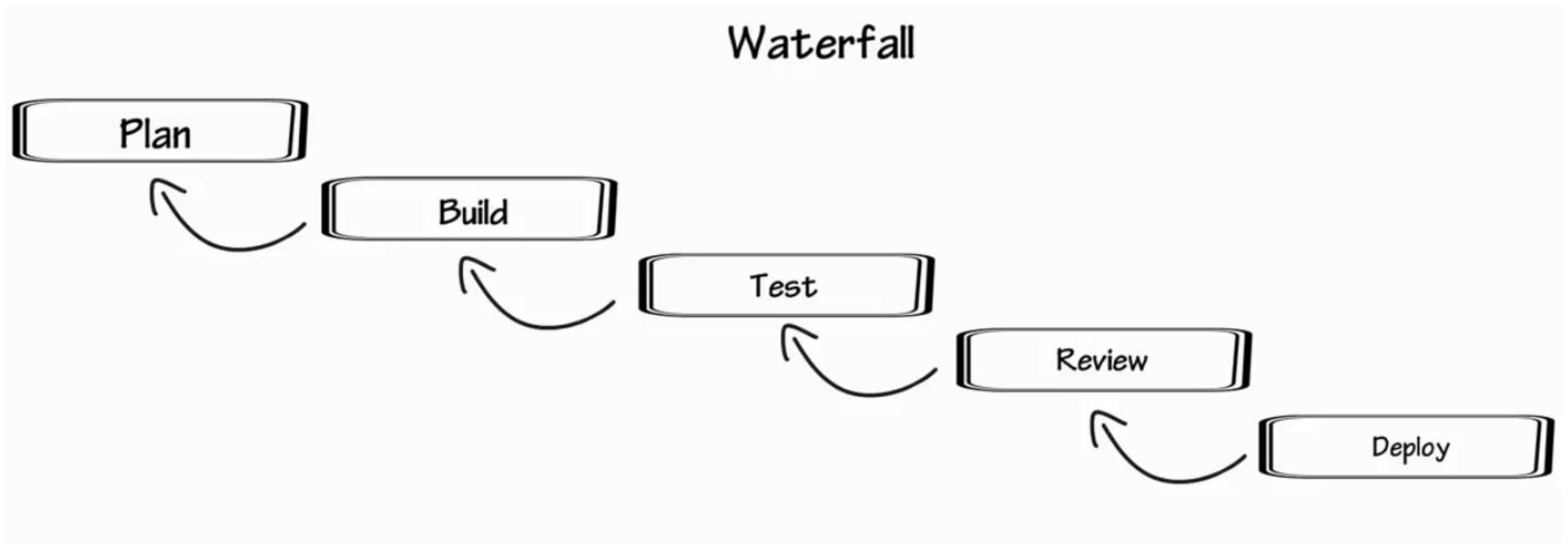
What is Scrum?

- 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.

Terminology:

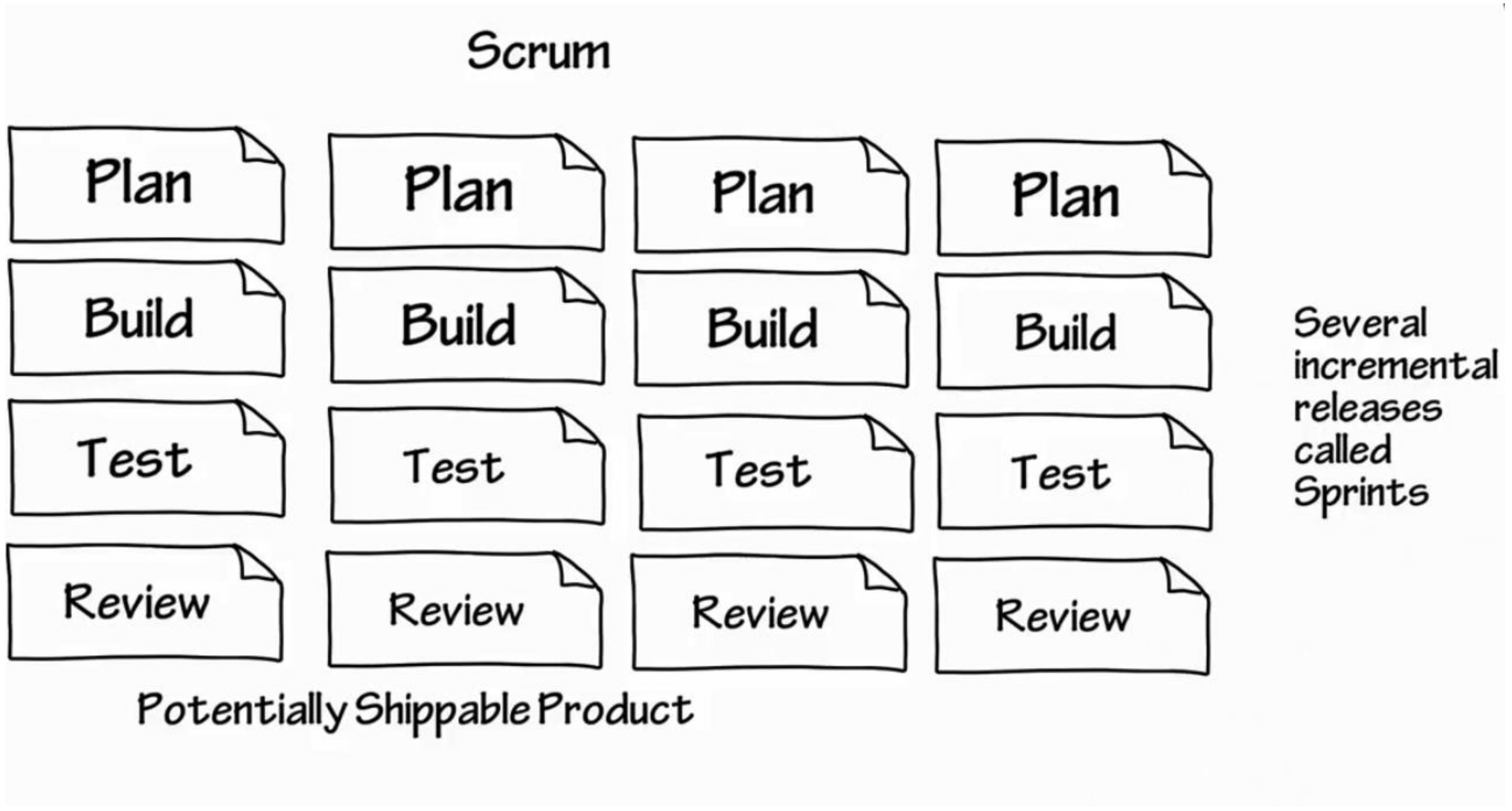
In the sport of rugby, a ***Scrum*** is a way of restarting the game, when the ball has gone out of play and 7-8 players work to move the ball forward.

Agile Development – Introduction to Scrum



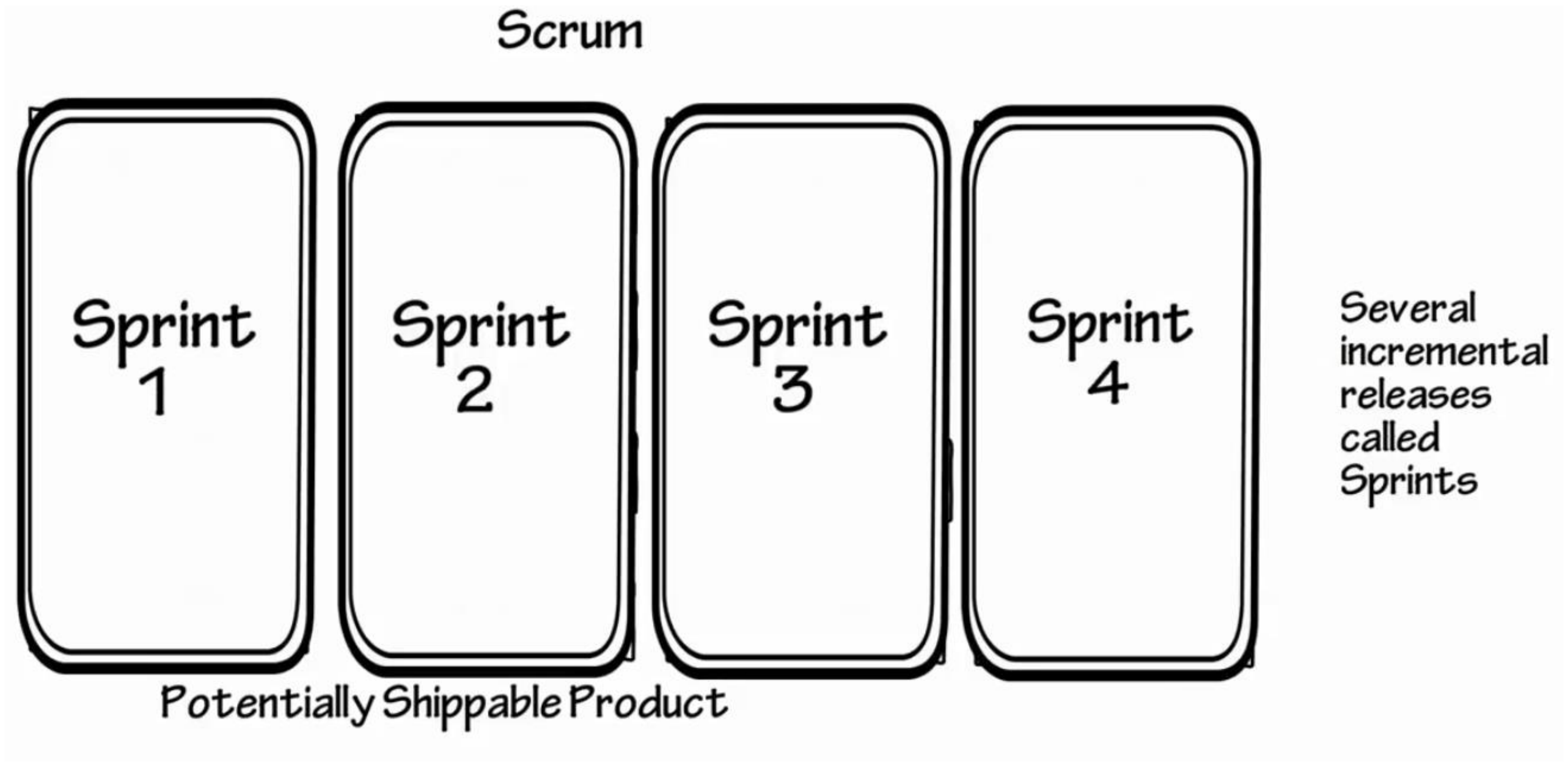
[Steve Stedman, https://www.youtube.com/watch?v=9TycLR0TqFA]

Agile Development – Introduction to Scrum



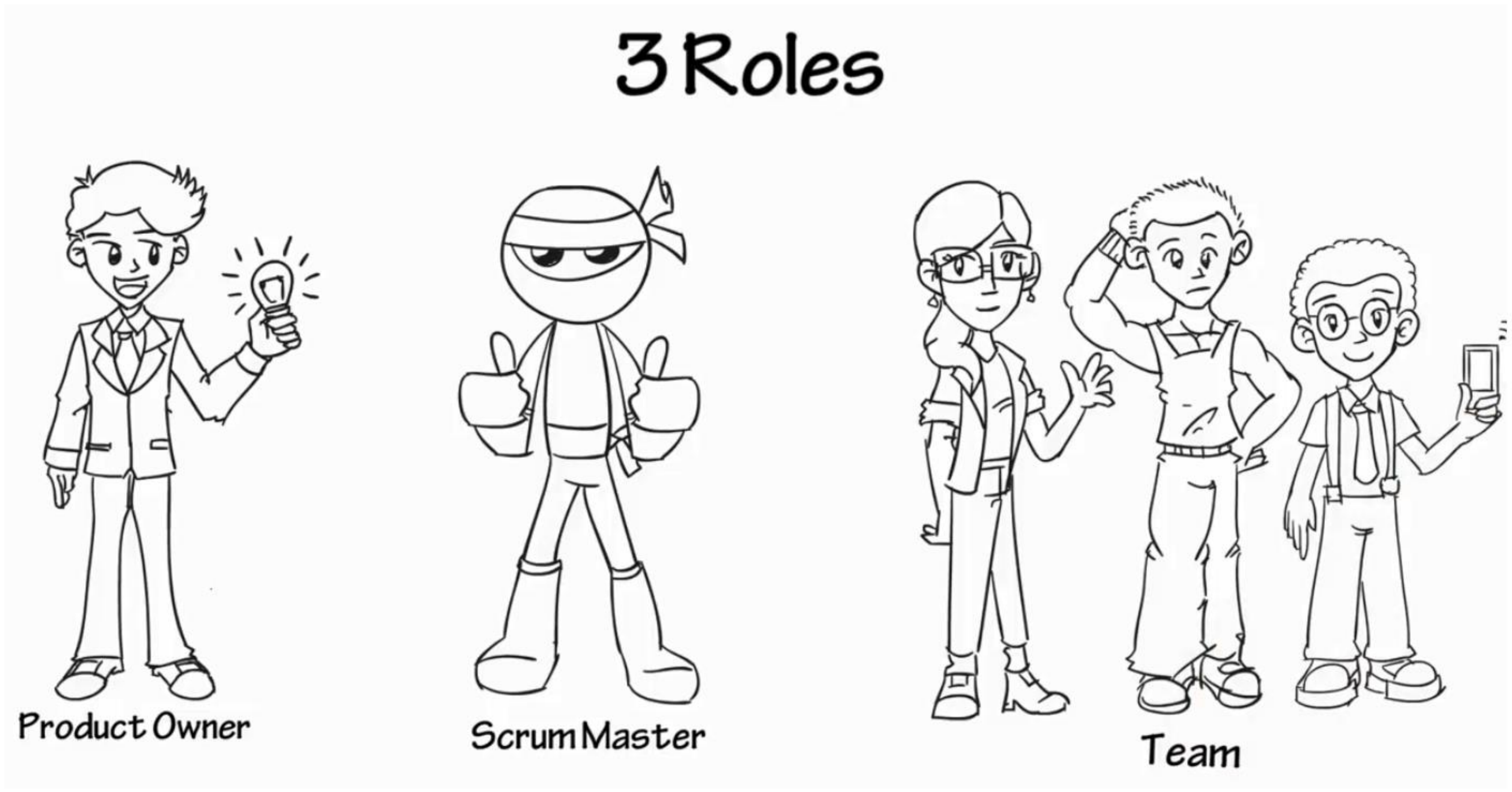
[Steve Stedman, https://www.youtube.com/watch?v=9TycLR0TqFA]

Agile Development – Introduction to Scrum


Scrum
Sprint 1
Sprint 2
Sprint 3
Sprint 4
Several incremental releases called Sprints
Potentially Shippable Product

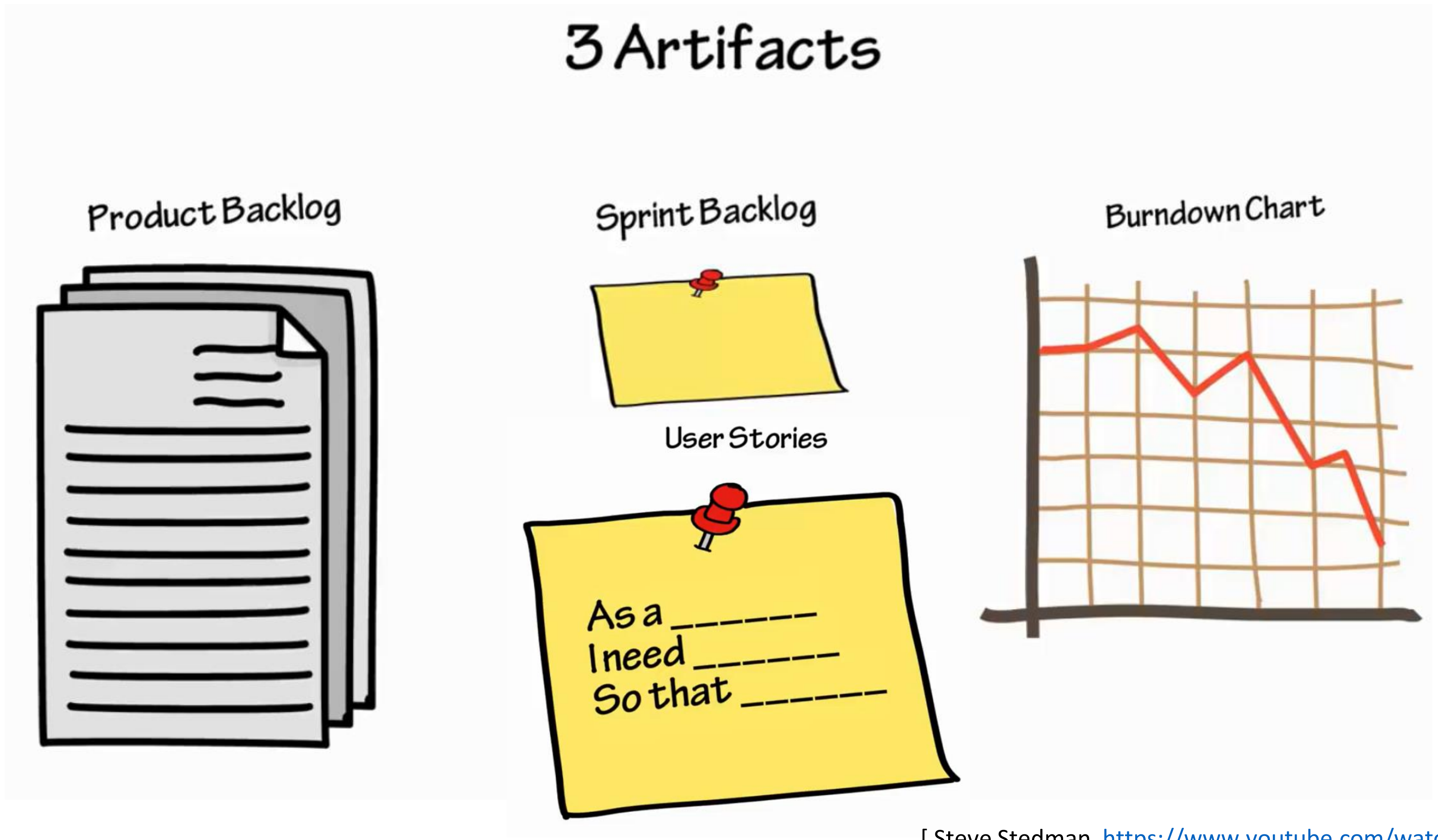
[Steve Stedman, https://www.youtube.com/watch?v=9TycLR0TqFA]

Agile Development – Introduction to Scrum



[Steve Stedman, https://www.youtube.com/watch?v=9TycLR0TqFA]

Agile Development – Introduction to Scrum



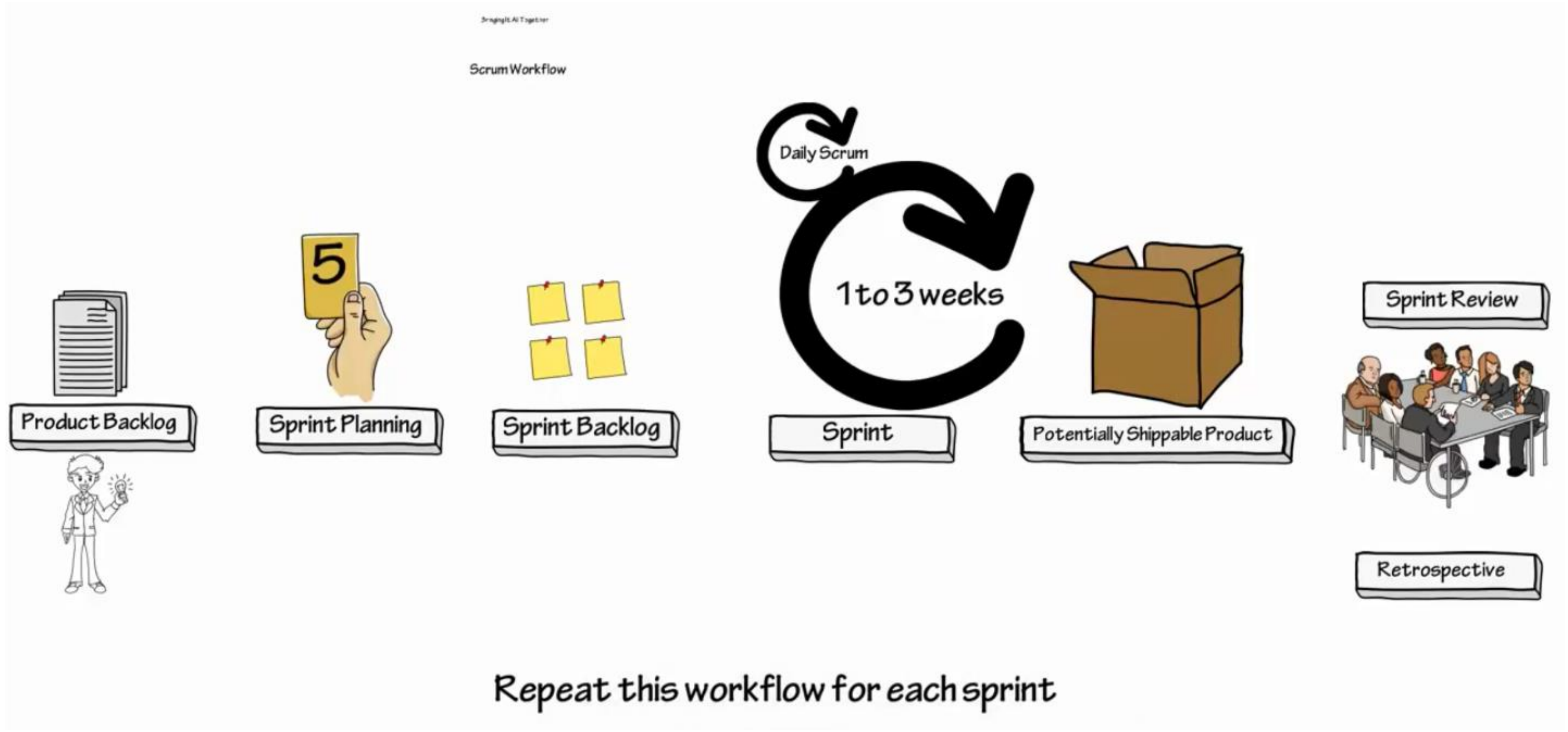
[Steve Stedman, https://www.youtube.com/watch?v=9TycLR0TqFA]

Agile Development – Introduction to Scrum

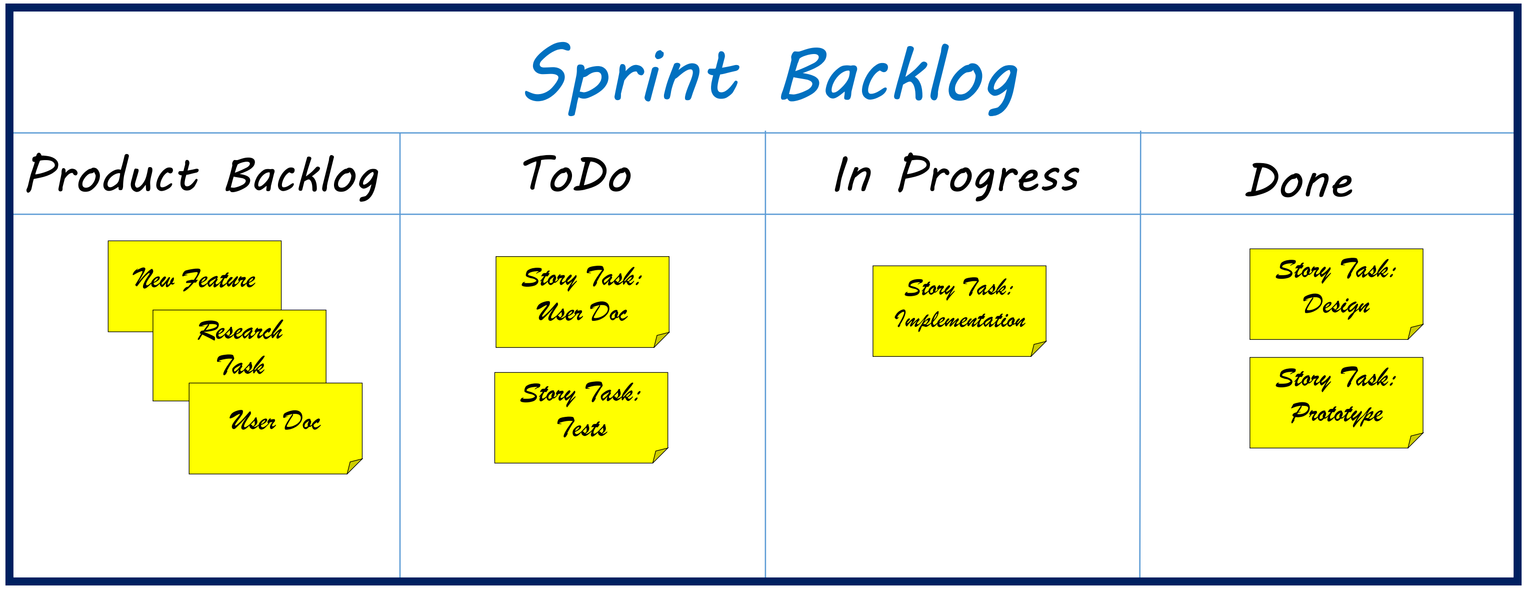
3 Ceremonies
Sprint Planning
5
Daily Scrum
Sprint Review

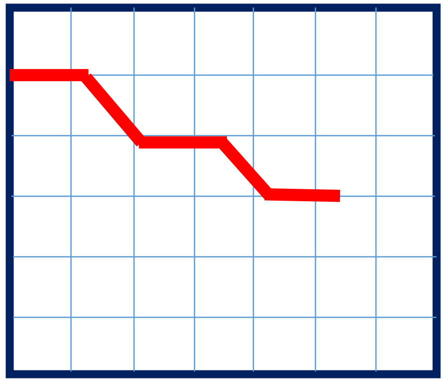
[Steve Stedman, https://www.youtube.com/watch?v=9TycLR0TqFA]

Agile Development – Introduction to Scrum



[Steve Stedman, https://www.youtube.com/watch?v=9TycLR0TqFA]


Sprint Backlog
Product Backlog
ToDo
In Progress
Done
New Feature
Research Task
User Doc
Story Task: User Doc
Story Task: Tests
Story Task: Implementation
Story Task: Design
Story Task: Prototype



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!

  He or she 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** both, 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 discuss **what was done in the sprint** and **adjust the product backlog** if necessary.

Introduction to Scrum

Scrum Team	Events	Artifacts
• Product Owner	• Sprint Planning	• Product Backlog
• Scrum Master	• Daily Scrum (Daily Stand Up)	• Sprint Backlog
• Development Team	• Sprint Review	• Sprint Progress
	• Sprint Retrospective	

- This is a **three-hour time-boxed** meeting which takes place 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**.
- **Eleminate waste !**
- From experience, this is the **most important event** !

Introduction to Scrum

Scrum Team	Events	Artifacts
• Product Owner	• Sprint Planning	• Product Backlog
• Scrum Master	• Daily Scrum (Daily Stand Up)	• Sprint Backlog
• Development Team	• Sprint Review	• Sprint Progress
	• Sprint Retrospective	

- The Product Backlog is a list of ToDo items, e.g.:
  - research tasks
  - feature definitions
  - architecture definitions
  - user stories (user requirements)
  - supplementary tasks
  - user documentation tasks
  - .. and more

Introduction to Scrum

Scrum Team

- Product Owner
- Scrum Master
- Development Team

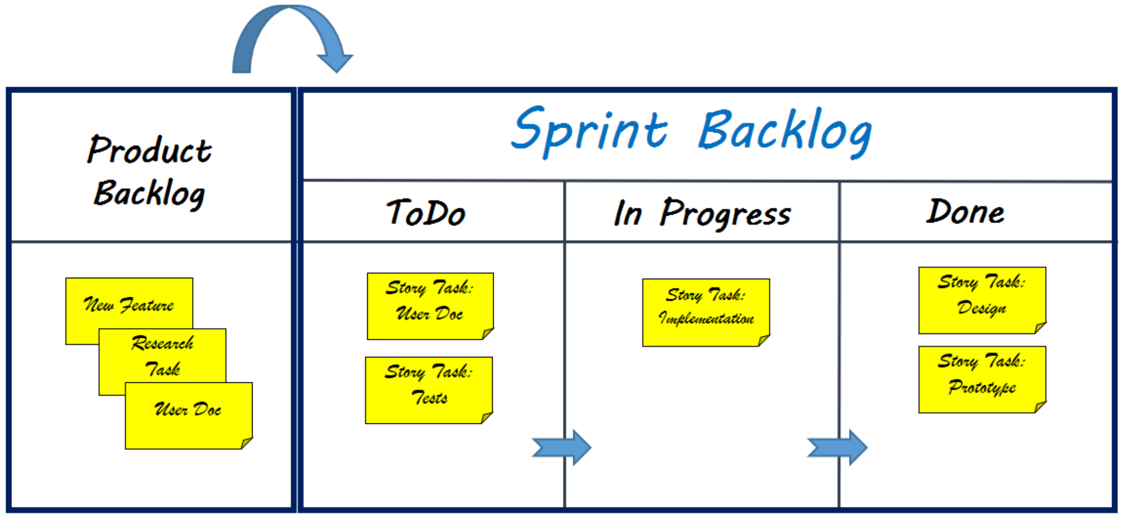
Events

- 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

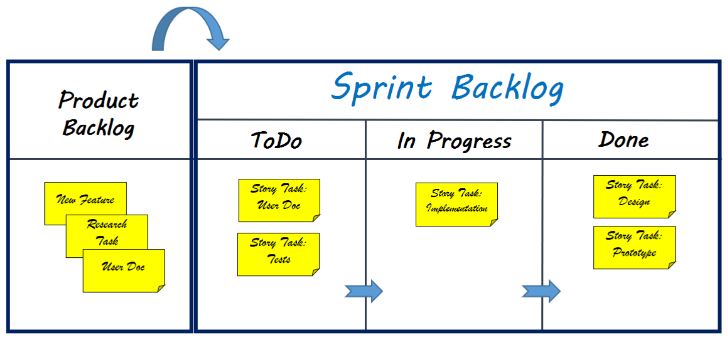
Events

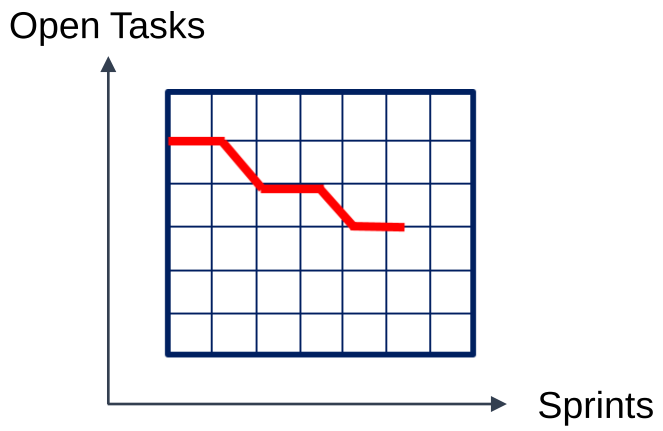
- Sprint Planning
- Daily Scrum (Daily Stand Up)
- Sprint Review
- Sprint Retrospective

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
  - all tests passed successfully
- 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.

Introduction to Scrum

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 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.mountaingoatsoftware.com/articles/toward-a-catalog-of-Scrum-smells]

Introduction to Scrum

Conclusions

- 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!

Content

Motivation

Plan-Driven vs Agile Software Development

Introduction to Scrum

Principles and Practices

Kanban – Small Projects

Tools

References

Principles and Practices

Scrum as a method works as container for agile development techniques, such as:

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

[Ian Sommerville, *"Software Engineering"*]

Principles and Practices

- Collective ownership
- Continuous integration
- Incremental planning
- Pair programming
- Refactoring
- Test-driven 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.

[Ian Sommerville, *"Software Engineering"*]

Principles and Practices

- Collective ownership
- Continuous integration
- Incremental planning
- Pair programming
- Refactoring
- Test-driven 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.

[Ian Sommerville, *"Software Engineering"*]

Principles and Practices

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

- Requirements are recorded on *“story cards”*.
- The stories to be included in a release are determined by:
  - the time available
  - their relative priority

[Ian Sommerville, *“Software Engineering”*]

Principles and Practices

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

- Developers work in pairs.
- Checking each other's work.
- Providing support.
- Knowledge transfer.

[Ian Sommerville, *"Software Engineering"*]

Principles and Practices

- Collective ownership
- Continuous integration
- Incremental planning
- Pair programming
- Refactoring
- Test-driven 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.

[Ian Sommerville, *"Software Engineering"*]

Principles and Practices

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

- An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.

[Ian Sommerville, *"Software Engineering"*]

Content

Motivation

Plan-Driven vs Agile Software Development

Introduction to Scrum

Principles and Practices

Kanban – Small Projects

Tools

References

Kanban

What is Kanban?

- Like Scrum, ***Kanban*** is also is an ***Agile Method*** offering **management framework** for software development.
- It's even more lightweight than Scrum and is preferably used in small projects (a few developers), and projects where:
    - tasks shift on a daily basis are unpredictable, and not plannable,
    - a fixed Scrum-sprint length planning is not possible.

How it works.

- Work items presented visually on a **Kanban board** (***ToDo, In Progress, Done***).
- Work is prioritized and pulled from backlog **when capacity becomes available**.
- Requires real-time communication of capacity and **full transparency** of the work.
- Only a **limited number** of “In Progress” items.

Content

Motivation

Plan-Driven vs Agile Software Development

Introduction to Scrum

Principles and Practices

Kanban – Small Projects

Tools

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.

GitHub

www.github.com

GitLab

www.gitlab.com

Atlassian Jira

www.atlassian.com/software/jira

asana

www.asana.com

icescrum

www.iceScrum.com

Tools

Agile Development Supporting Tools and Platforms

- Modern software development tools and platforms support agile methodologies and workflows:
  - Version control
  - Test-driven development (TDD)
  - Peer-review
  - Continuous integration, testing and delivery (CI/CD)
  - Basic agile managment

GitHub

www.github.com

GitLab

www.gitlab.com

Tools

GitLab

www.gitlab.com

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

- Git-repository
- Issue-tracker
- CI/CD pipeline
- Basic agile software development workflow support
- Basic project management functionality
- Milestones
- Configurable issue board
- Wiki
- Simple role management
- Community Edition is free of charge

Content

Motivation

Plan-Driven vs Agile Software Development

Introduction to Scrum

Principles and Practices

Kanban – Small Projects

Tools

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.

Roman Pichler · Stefan Roock (Hrsg.)

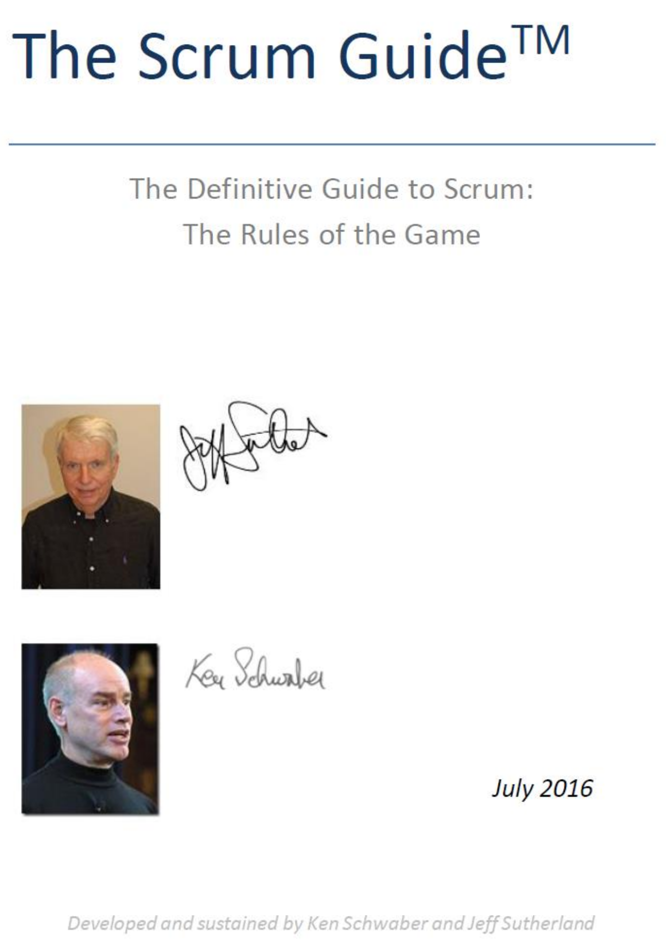
Agile Entwicklungspraktiken mit Scrum

→ Mit Beiträgen von:
Alex Bepple, Jens Coldewey, Jutta Eckstein, Peter Friese, Andreas Havenstein, Johannes Link, Martin Lippert, Roman Pichler, Stefan Roock, Bernd Schiffer, Henning Wolf

dpunkt.verlag

References

www.Scrum.org



The Scrum Guide^{TM}

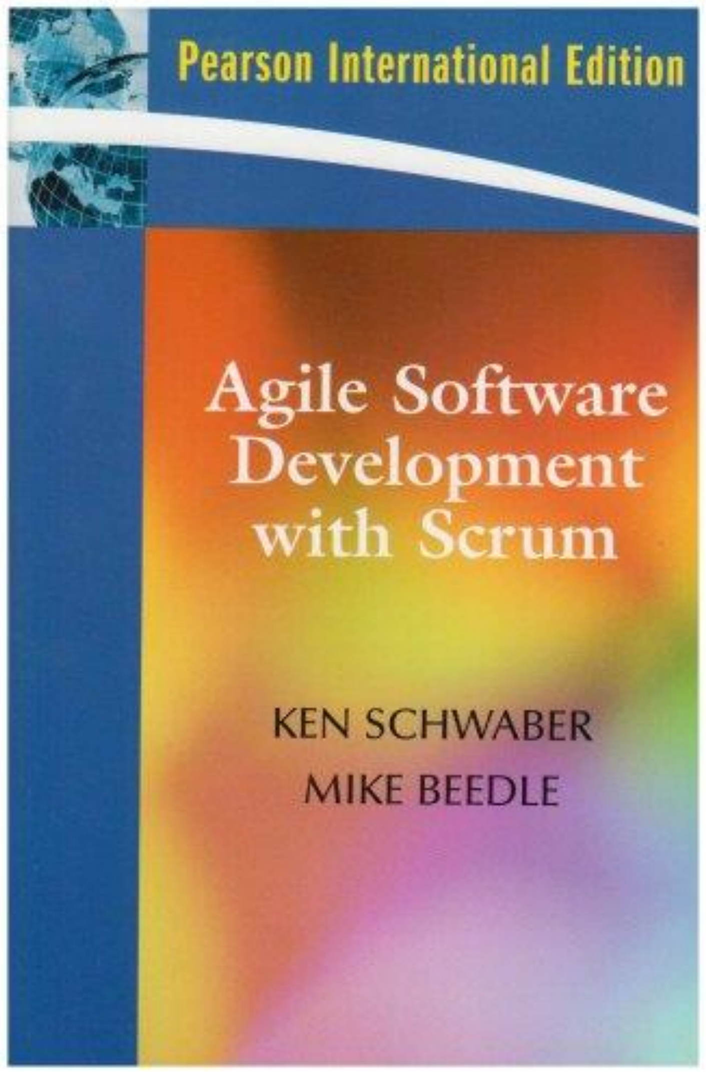
The Definitive Guide to Scrum:
The Rules of the Game

[illegible]

[illegible]

July 2016

Developed and sustained by Ken Schwaber and Jeff Sutherland



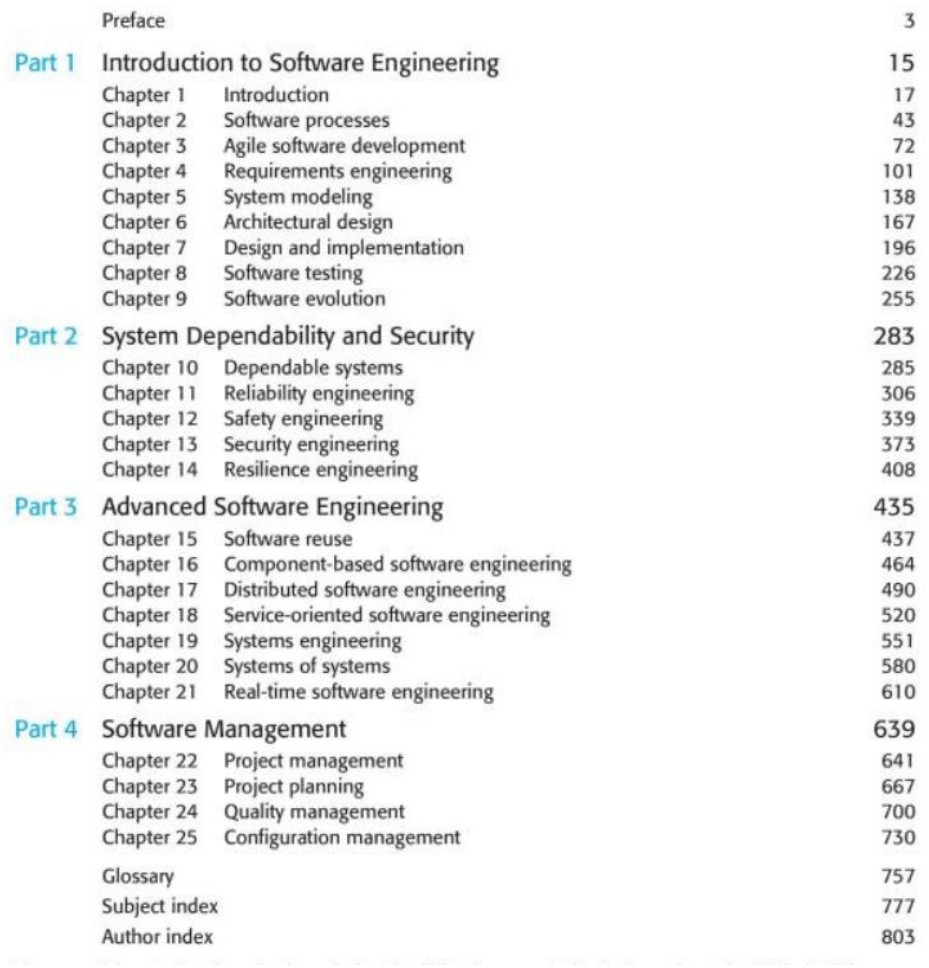
Pearson International Edition

Agile Software Development with Scrum

KEN SCHWABER
MIKE BEEDLE

References

The possibly most comprehensive book.



Preface 3

Part 1 Introduction to Software Engineering 15

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

Part 2 System Dependability and Security 283

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

Part 3 Advanced Software Engineering 435

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

Part 4 Software Management 639

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