Software Engineering

SE Processes I + II

Software & Systems Engineering | Prof. Dr. Andreas Vogelsang | 08.01.2024 und 10.01.2024

@andivogelsang

vogelsang@cs.uni-koeln.de

Learning Goals

- Know what development processes are
- Know the basic types of processes: sequential, iterative, incremental
- Know sequential processes: Waterfall, V-Modell
- Know iterative processes: Spiral model
- Know agile development and Scrum

Software Development Processes

Software Development Process

Motivation

- How to **structure** a project?
- What are **activities** and phases? analysis (requirements elicitation + system modeling), design (architectural + software design), implementation, test, deployment
- How to organize **communication**?
- Who has which **responsibilities**?
- Did we **forget** anything?
- Can we **predict** the project result?
- How to **manage and control** progress?
- How to share and elicit **experience**?
- How to synchronize **hardware** and software development?

Software Development Process [Sommerville]

A **software process** is a set of related activities that leads to the production of a software system. [...] **Products** or deliverables are the outcomes of a process activity. [...] **Roles** reflect the responsibilities of the people involved in the process. [...] Pre- and postconditions are **conditions** that must hold before and after a process activity."

Why different processes? [Sommerville]

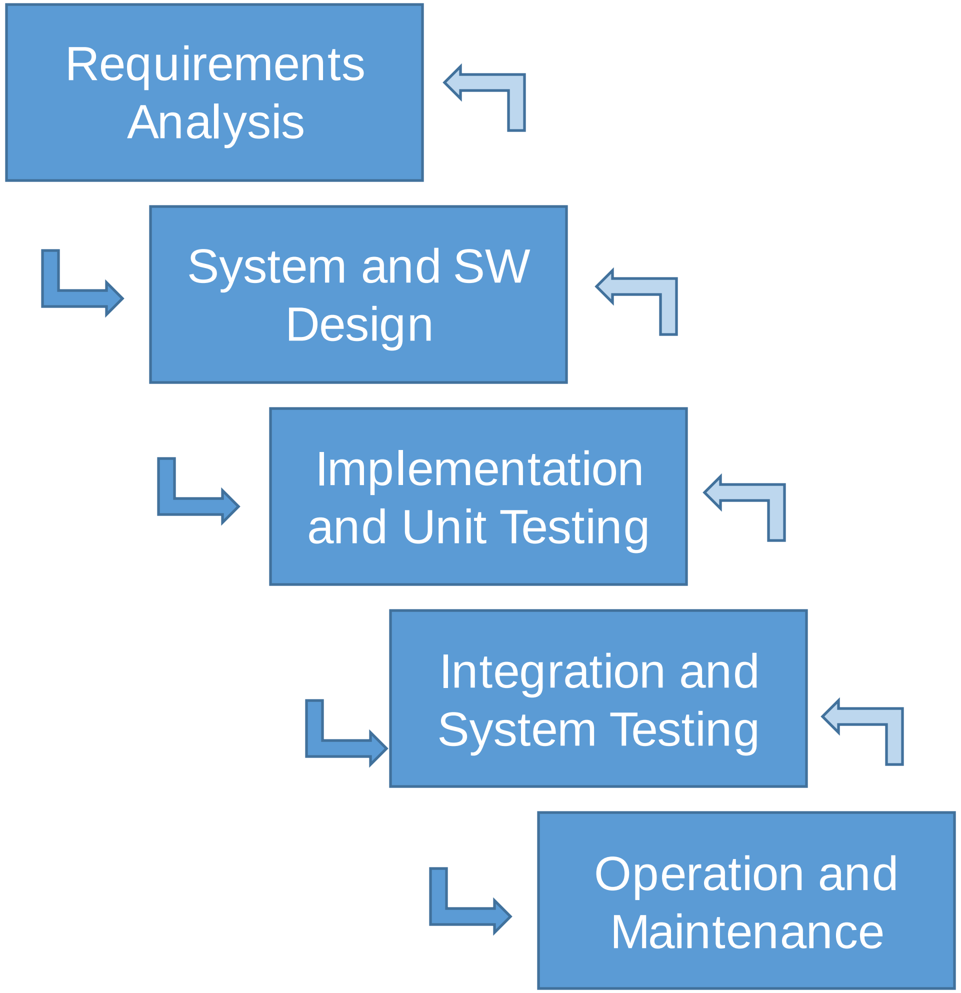
The process used in different companies depends on the type of software being developed, the requirements of the software customer, and the skills of the people writing the software.

The Waterfall Model

The Waterfall Model

Waterfall Model

- first process model, motivated by practice
- by Winston W. Royce 1970
- each development phase ends by the approval of one or more documents (document-driven process model)
- phases do not overlap
- numerous variants with varying number of phases: 5–7
- here: simplified variant by Sommerville


Requirements Analysis
System and SW Design
Implementation and Unit Testing
Integration and System Testing
Operation and Maintenance

The Waterfall Model – Phases [Sommerville]

1. Requirements Analysis

The system's services, constraints, and goals are established by consultation with system users. They are then defined in detail and serve as a **system specification**.

2. System and Software Design

The systems design process allocates the requirements to either hardware or software systems. It establishes an overall **system architecture**. Software design involves identifying and describing the fundamental software system abstractions and their relationships.

3. Implementation and Unit Testing

The software design is realized as a set of **programs or program units**. Unit testing involves verifying that each unit meets its specification.

4. Integration and System Testing

The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the **software system is delivered** to the customer.

5. Operation and Maintenance

Normally, this is the longest life-cycle phase. The system is installed and put into practical use. Maintenance involves **correcting errors** that were not discovered in earlier stages of the life cycle [...].

The Waterfall Model – Discussion

Advantages

- easy to understand, manage, and control
- good for systems development (i.e., with high manufacturing costs for hardware)
- easier to use the same model as for hardware
- combination with formal system development feasible (e.g., B method)

Example Domains

embedded systems where software must interface with hardware systems

critical systems with extensive safety and security analysis of specification and design

large software systems that are typically developed by several companies

Disadvantages

- for software development: stages should feed information to each other
- changes in previous stages are hard to achieve
- problems from previous stages left for later resolution
- freezing of requirements may lead to software not wanted by the user
- freezing of design may lead to bad structure and implementation tricks
- requires clear and stable requirements and good design upfront

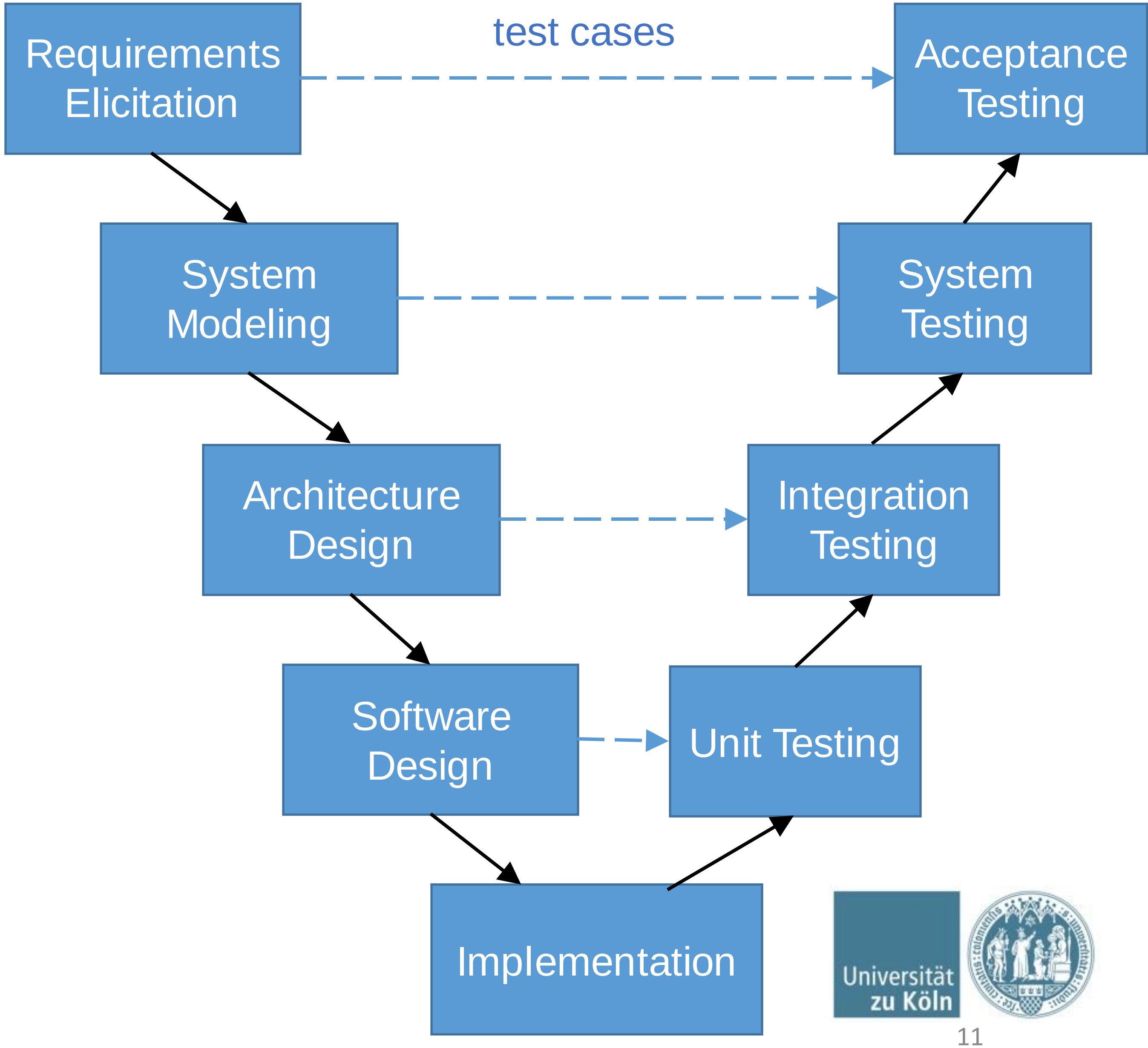
1:32pm
July 16th 1969

The V-Model

The V-Model

V-Model

- developed by the German Ministry of Defense (Verteidigungsministerium) and required since 1992
- extension of the waterfall model: project-aligned activities such as quality assurance, configuration management, project management
- 1997: V-model 97: incremental development, inclusion of hardware, object-oriented development
- 2004: V-model XT (for extreme tailoring): adaptability, application beyond software
- integration of four testing stages



Stages of Testing

1. Unit Testing

- each component is tested independently
- unit may stand for a component or smaller entities (package, class, method)
- tests created by the developers
- automation is common (e.g., JUnit)

2. Integration Testing

- some components are integrated (e.g., into subsystems) and tested together
- detects inconsistencies in interfaces and communication between components
- top-down vs. bottom-up integration

3. System Testing

- all components are integrated to the complete system
- detects further inconsistencies and unanticipated interactions
- system is tested against system requirements

4. Acceptance Testing

- final stage in the testing process before accepted for operational use
- system is tested against user requirements and with real data
- performed by (potential) customer

The V-Model – Discussion

Advantages

- quality assurance in several testing stages
- completeness helps to not miss activities
- V-model 97/XT are widely applicable (e.g., hardware, incremental)
- Over the waterfall model
  - **Standardization**: Terminology, roles, work products
  - **Effort estimation**: easier due to system breakdown

Disadvantages

- complex and extensive process
- adaptations often required (cf. XT for extreme tailoring)
- overhead useful only for large software systems
- changes in requirements are problematic

Example Domains

- since 1992: V-model required by German government (e.g., Bundeswehr), since 2004 V-model XT
- embedded, critical, and large software systems as for the waterfall model

When does the V-Model work?

When does the V-Model work well?

- Requirements are more or less clear
- Requirements are more or less stable
- The system solution may be complicated but can easily be decomposed into smaller units
- The single units can be developed more or less independently

Major challenge: Change!

- Software seems to be easily changeable
- Developers like to change/improve code
- Customers usually do not understand what is easy and what is hard to change
- Plus: In systems (mechanics, electronics, software), software is often the "glue" that is developed and tested last.

Solution: Iterations

If requirements are unclear and volatile and/or if technology is challenging, you need **iterations** and **feedback** to approach a solution

Universität zu Köln

Iterative Processes

Process Iterations

Iterative Processes

- Build the product in iterations
- Interleave and repeat
  - Requirements Engineering, Risk Assessment
  - Architecture and Design
  - Implementation
  - Quality Assurance
  - Deployment

Incremental Processes

- Build/extend the product by increments
- Start with a core product
- Extend the core product with additional features

How to decide?

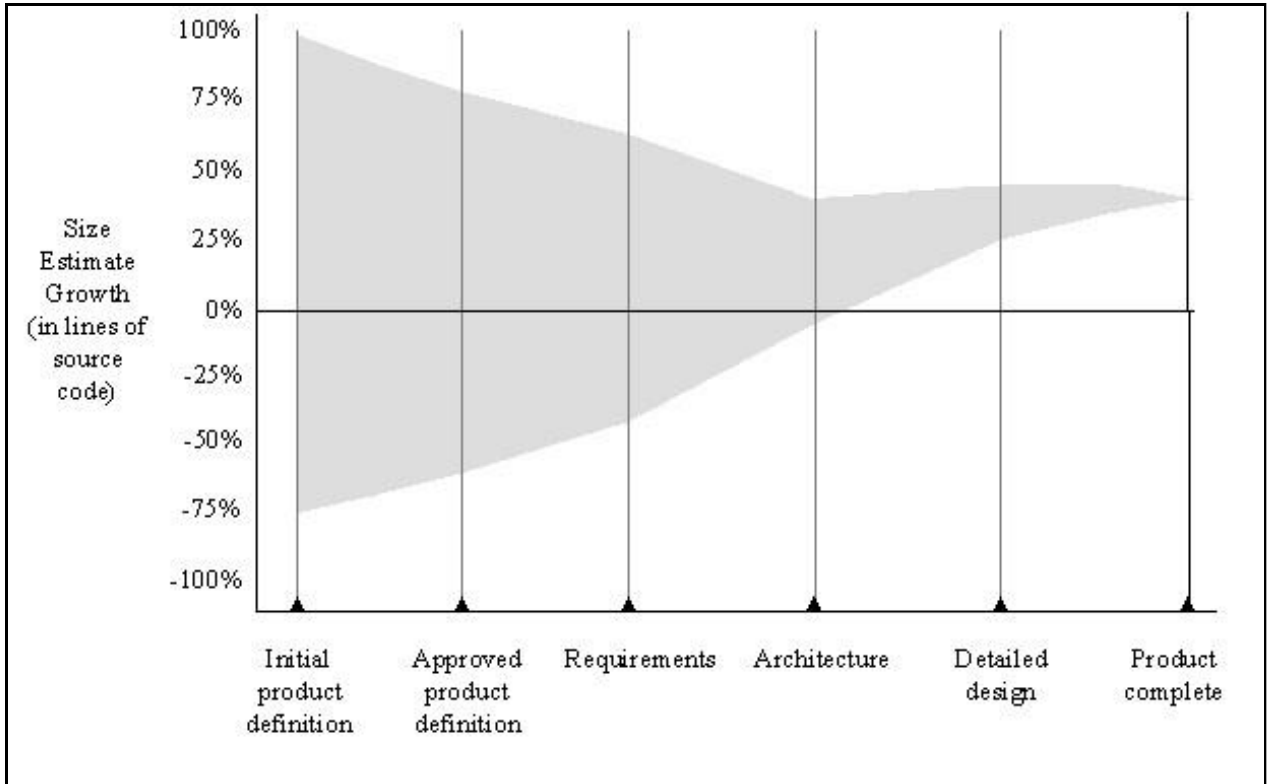
How much of which activity at which point?
How often and in which order?

Risks

Risks in SE Projects

- Project Risks
  - Project is late
  - Project is not well planned
  - Budget overrun
- System risks
  - Safety or security issues
  - Usability and acceptance issues
- Engineering risks
  - Inappropriate technology choices
  - Testing and validation issues
  - Performance and scalability issues

The “Cone of Uncertainty”

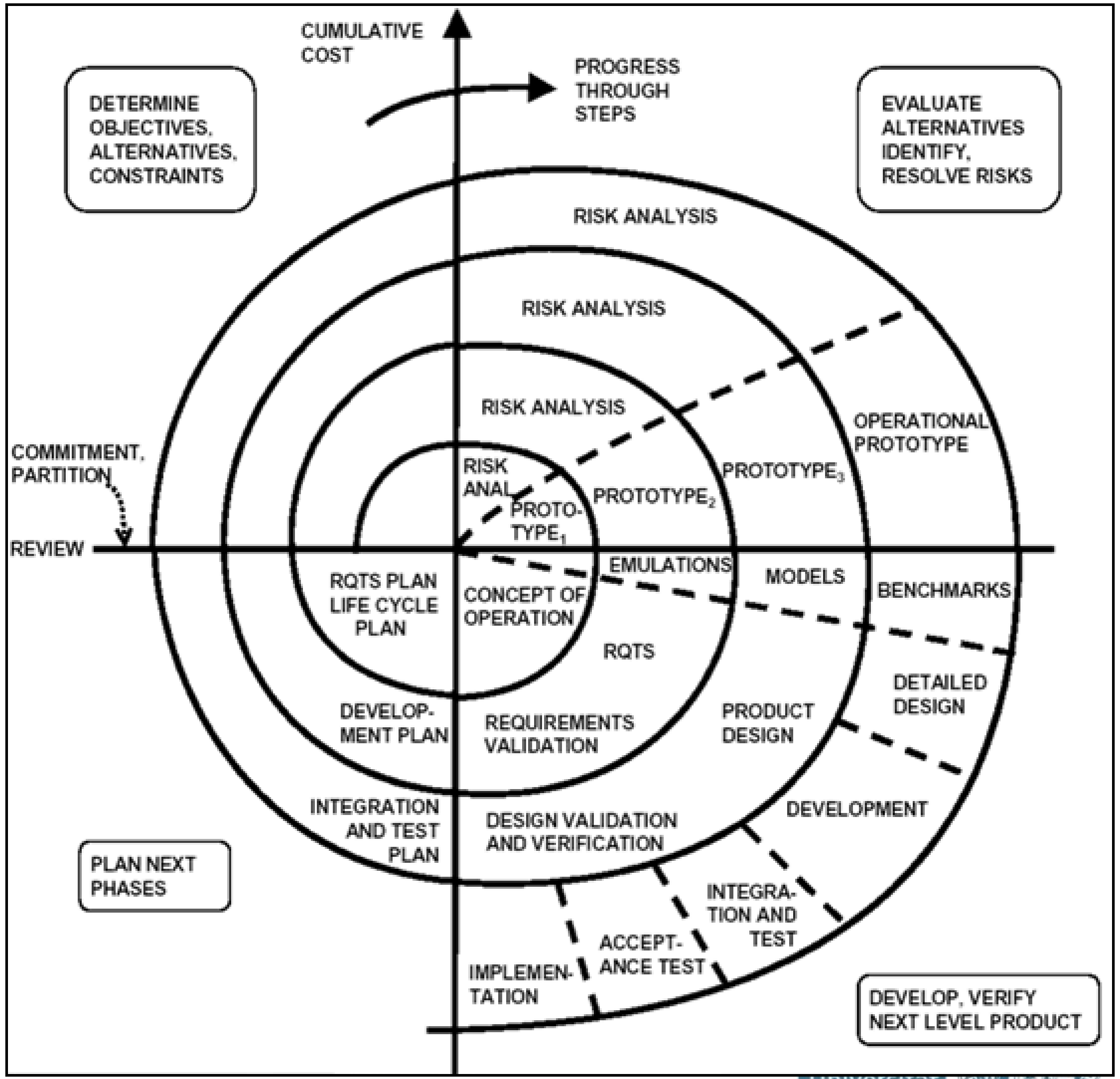


Copyright 1998 Steven C. McConnell. Reprinted with permission from *Software Project Survival Guide* (Microsoft Press, 1998).

The Spiral Model

The Spiral Model

- Iterative process driven by risks
- Each iteration cycle has four basic activities:
  1. Determine stakeholder objectives and solution alternatives
  2. Evaluate alternative solutions (e.g., by prototypes) to reduce risk
  3. Develop and test intermediate work product(s)
  4. Plan next iteration
- Risk determines the level of effort.
- After each cycle, stakeholders commit to continue the project



Agile Development

Iterative + Incremental = Agile

Agile Development

Motivation [Sommerville]

- businesses operate globally and in a rapidly changing environment
- software is part of almost all business operations
- new software has to be developed quickly
- often infeasible to derive a complete set of stable requirements
- plan-driven process models (e.g., waterfall) deliver software long after originally specified

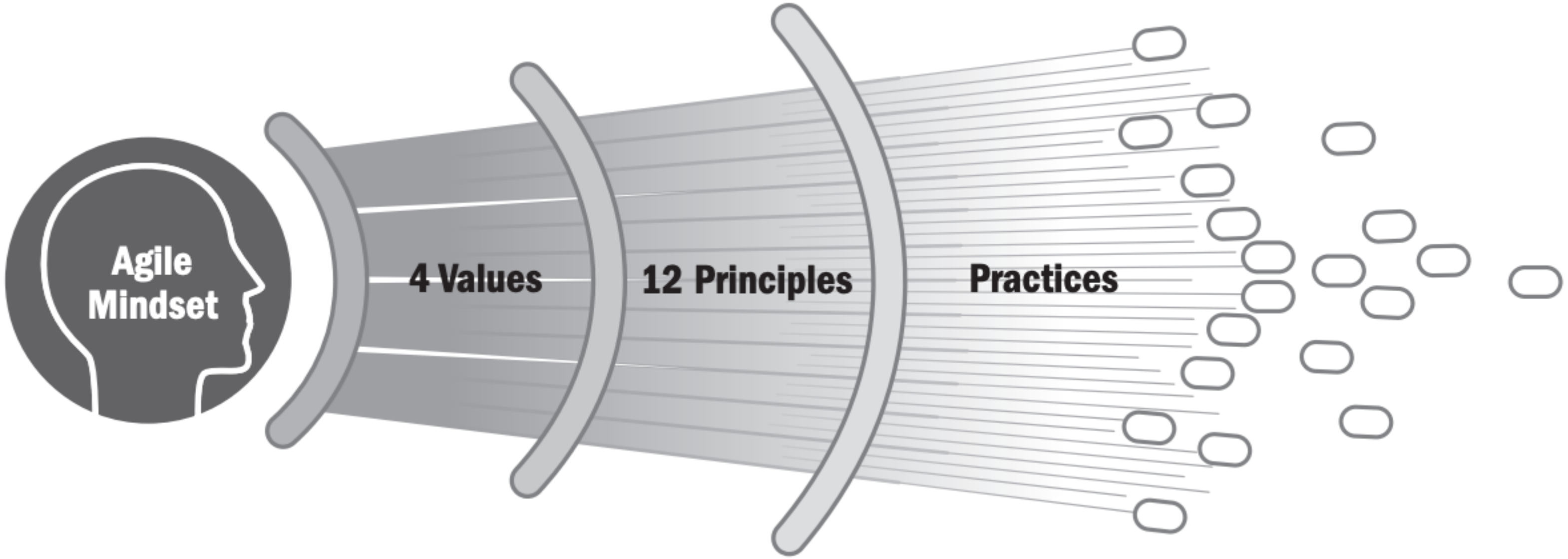
Agile (Development) Methods [Sommerville]

Development of agile methods since late 1990s:

1. specification, design, implementation are interleaved
2. each increment is specified and evaluated by stakeholders (e.g., end-users)
3. extensive tool support is used

Agile in a Nutshell

- A project management approach that aims to respond to change and unpredictability, primarily through incremental, iterative workflows (often referred to as "sprints").
- Also: a collection of practices to facilitate this approach.
- All are based on the principles described in "The Manifesto for Agile Software Development".



The Manifesto for Agile Development

Values of Agile Development [agilemanifesto.org]

“We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

- individuals and interactions
  over processes and tools
- working software
  over comprehensive documentation
- customer collaboration
  over contract negotiation
- responding to change
  over following a plan

That is, while there is value in the items on the right, we value the items on the left more.”

Principles of Agile Development

Principles [Individuals and Interactions]

- Build projects around **motivated individuals**. Give them the environment and support they need and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation**.
- The best architectures, requirements, and designs emerge from **self-organizing teams.**

Principles [Working Software]

- **Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- **Working software** is the primary measure of progress.
- Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to **technical excellence and good design** enhances agility.
- **Simplicity**–the art of maximizing the amount of work not done–is essential.

Principles [Customer Collaboration]

- Our highest priority is to **satisfy the customer** through early and continuous delivery of valuable software.
- Business people and developers must **work together daily** throughout the project.

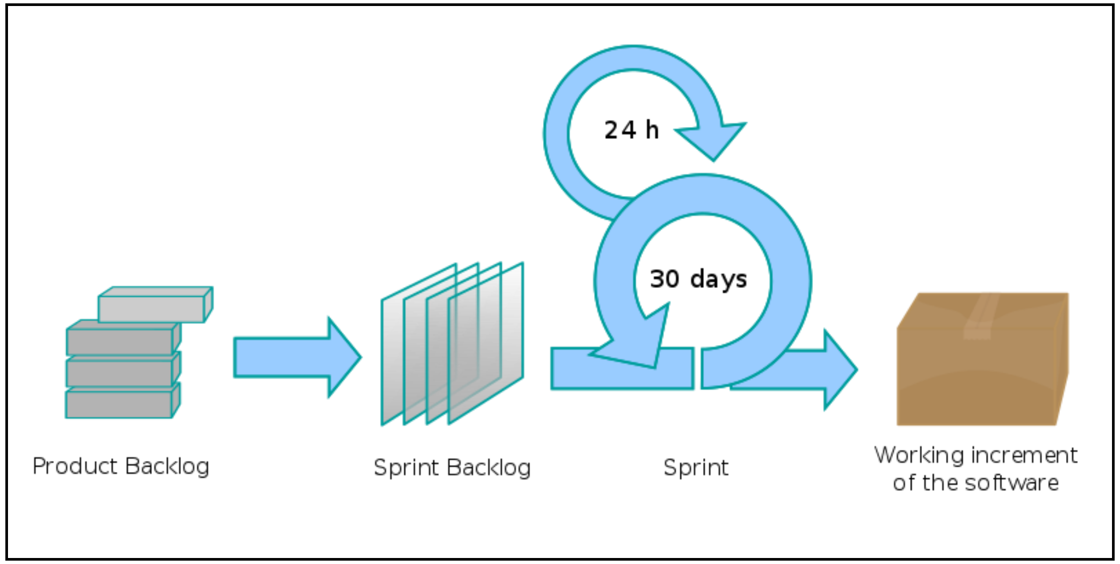
Principles [Responding to Change]

- **Welcome changing requirements**, even late in development. Agile processes harness change for the customer’s competitive advantage.
- At regular intervals, the **team reflects** on how to become more effective, then tunes and adjusts its behavior accordingly.

Scrum

Scrum

- an agile method, most-widely used method
- no special development techniques (like pair programming, test-driven development)
- **product backlog**: list of user stories, collected and prioritized by the **product owner**
- **sprint backlog**: user stories selected by the scrum team for the next **sprint**



The Scrum Team

- 5-11 persons
- cover all skills necessary to operate and further develop a product (i.e., requirements, development, testing, deployment, and operation)
- The entire team is responsible for the delivery result.
- Team members work closely together and learn from each other (cross-functionally).
- Products (applications, components, services) are developed and operated over a long period of time in unchanged teams.

● Product Owner

● Scrum Master

● Business Analyst

● Software Engineer

● Software Engineer

● Quality Engineer

● Operations Engineer

● Operations Engineer

The Product Owner

- One PO per agile team
- Functional responsibility for the deliverable
- Responsible for maximizing business value
- Main point of contact for customers and stakeholders.
- Represents the interests of customers and users
- Prioritizes the product backlog
- Defines acceptance criteria

The Scrum Master

- Concerned 100% with the development of the team
- Goals
  - Optimize and maximize outcome (generated value)
  - Adherence to the Agile/Scrum framework and live agile values
- Coaches the Product Owner and Team Members
- Moderates the regular meetings
- Pays attention to group dynamic aspects
- Removes hurdles

Start of a Sprint: The Sprint Planning Meeting

- Regular meeting that kicks off a sprint (lasts 1-4 hours)
- Goal: Establish a shared understanding and commitment among the Scrum team regarding the work to be accomplished during the upcoming sprint
- Agenda
  1. Reviewing the Product Backlog (PO + Dev team)
  2. Clarifying User Stories (PO, Stakeholders, Dev team)
  3. Estimating User Stories (Dev team)
  4. Selecting User Stories for the Sprint (Dev team + PO)
  5. Creating the Sprint Backlog (PO + Dev team)
  6. Planning for the Sprint (Dev team)

At the End of a Sprint

The Sprint Review

- Regular meeting at the end of a sprint (lasts 1-2 hours)
- Participants: Scrum team and stakeholders
- Goal: Present sprint results (ideally in a running demo) and get approval from stakeholders
- Key question: Which tasks have been completed, which have not been completed?

The Sprint Retrospective

- Regular meeting at the end of a sprint (lasts 0.5-2 hours)
- Participants: Only Scrum team
- Goal: Reflect on the process, identify issues in the process
- Key question: What went well and went should be improved?

During a Sprint

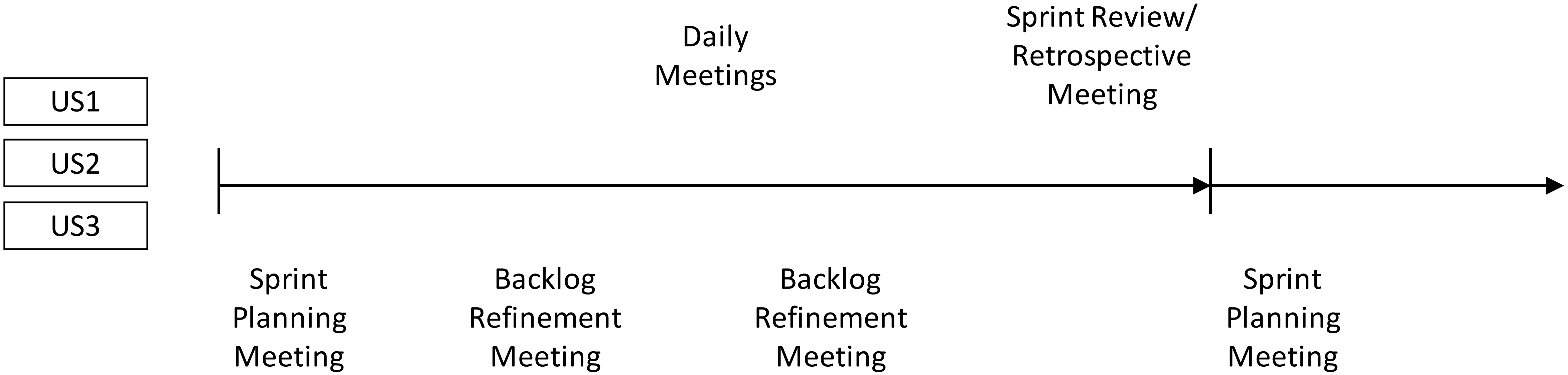
The Backlog Refinement Meeting (aka. Backlog Grooming)

- Regular meeting within a sprint
(0.5-1 meeting per week; 1-2 h per meeting)
- Goal: Ensure that the Scrum team has a well-prepared and refined backlog of user stories that are ready to be included in future sprints.
- Agenda
  - Welcome and Context
  - User Story Review
  - Break-Down User Stories
  - (Effort estimation)
  - Prioritization and Refinement
  - Dependencies and Risks

The Daily Scrum Meeting

- Daily meeting of the Scrum team
(max. 15 minutes (!))
- Goal: Everyone stays updated
- Agenda
  - Stand up in a circle
  - Every team member answer 3 questions briefly
    - What did I do yesterday?
    - What will I do today?
    - What barriers do I face?
  - Update Scrum Board

Scrum Process



US1

US2

US3

Daily Meetings

Sprint Review/ Retrospective Meeting

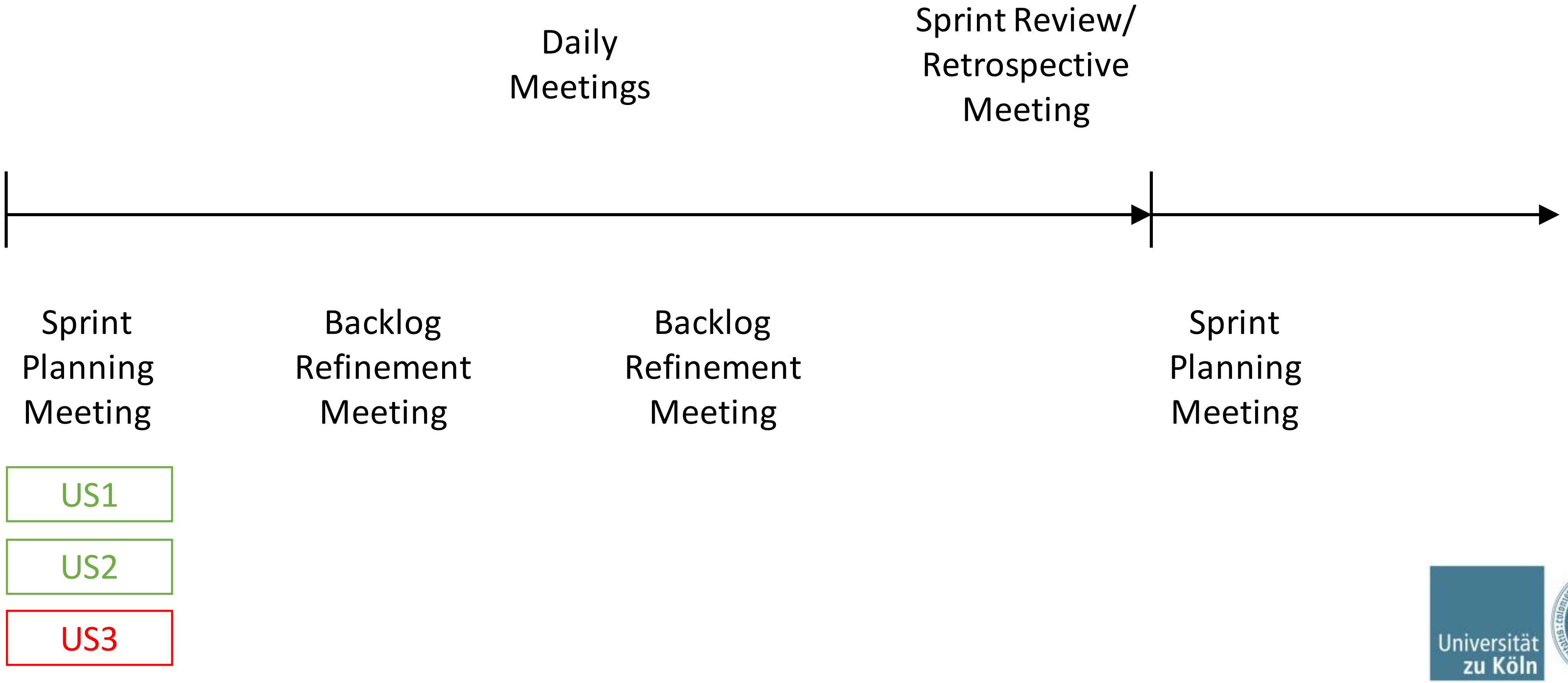
Sprint Planning Meeting

Backlog Refinement Meeting

Backlog Refinement Meeting

Sprint Planning Meeting

Scrum Process



Daily Meetings

Sprint Review/ Retrospective Meeting

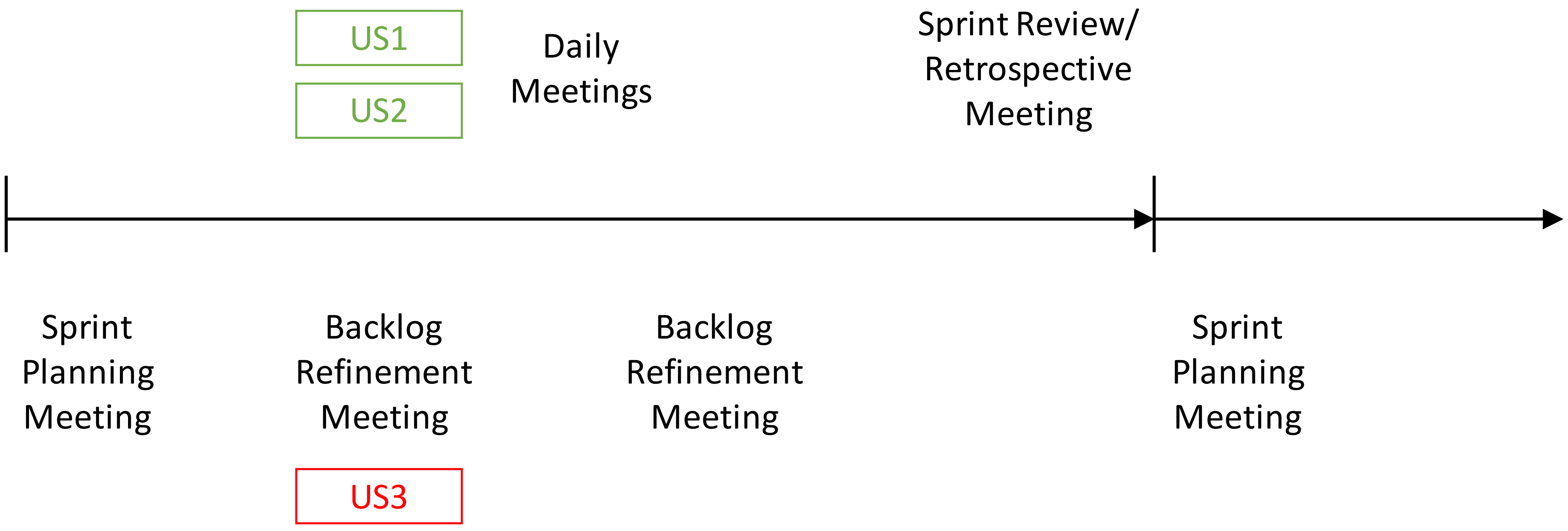
Sprint Planning Meeting

Backlog Refinement Meeting

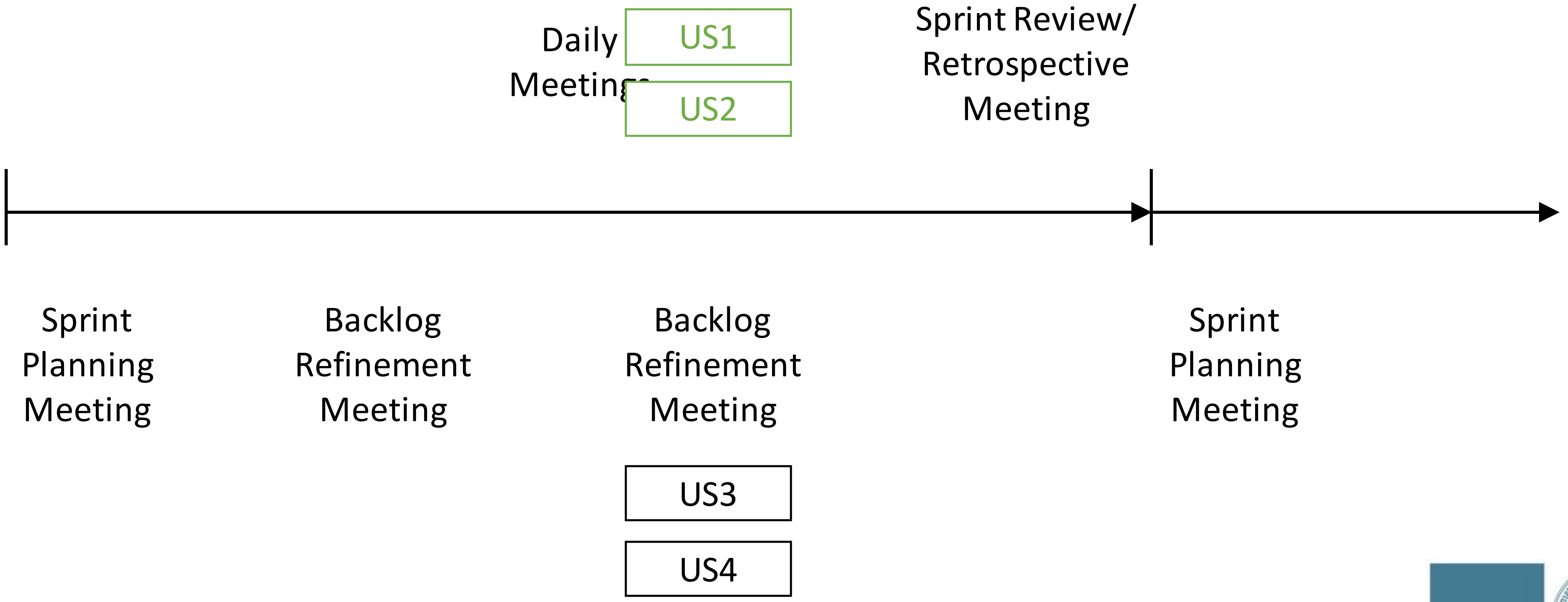
Backlog Refinement Meeting

Sprint Planning Meeting

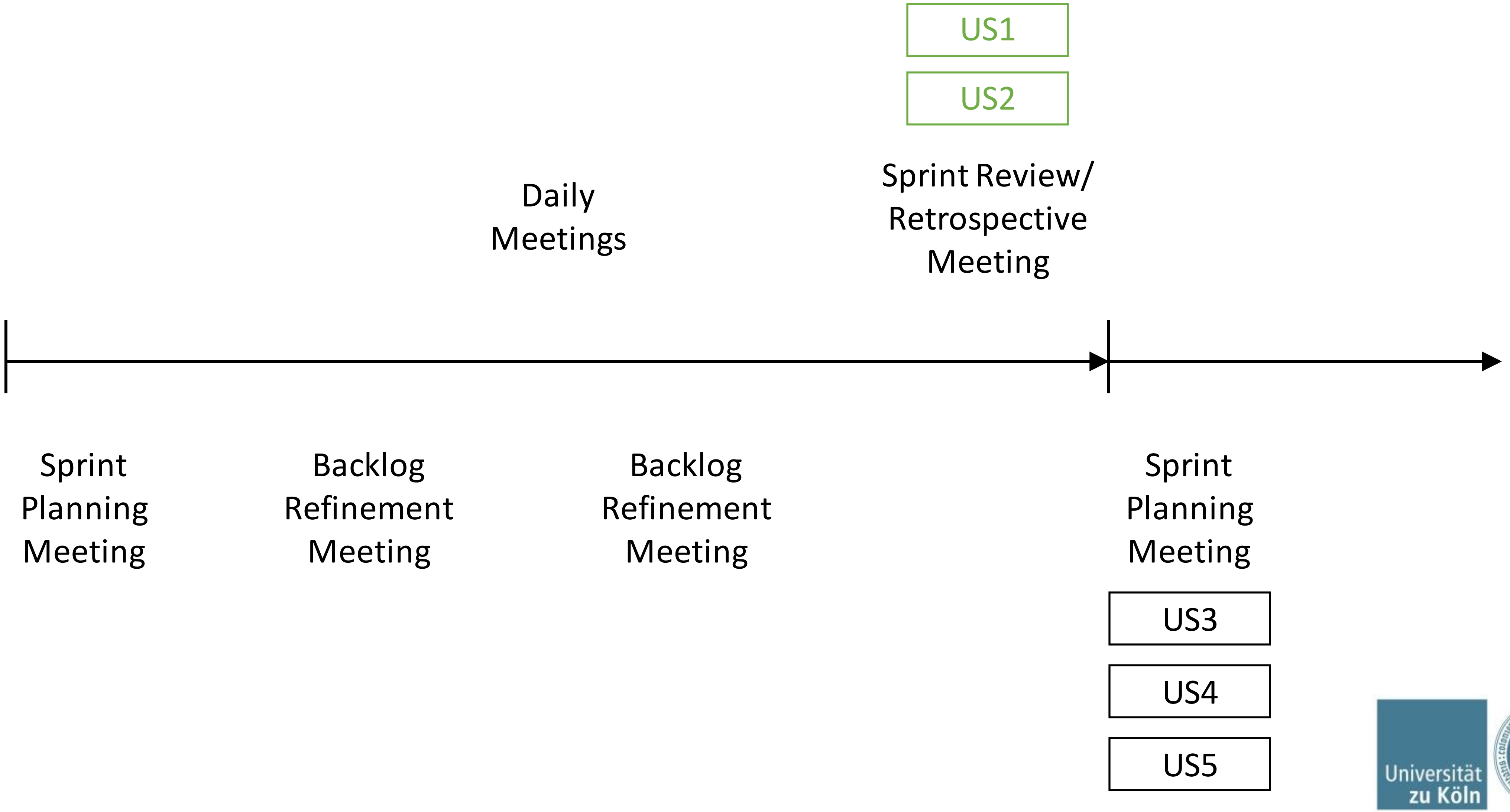
Scrum Process


US1
US2
Daily Meetings
Sprint Review/ Retrospective Meeting
Sprint Planning Meeting
Backlog Refinement Meeting
Backlog Refinement Meeting
Sprint Planning Meeting
US3

Scrum Process


Daily Meetings
US1
US2
Sprint Review/ Retrospective Meeting
Sprint Planning Meeting
Backlog Refinement Meeting
Backlog Refinement Meeting
Sprint Planning Meeting
US3
US4

Scrum Process



US1

US2

Daily Meetings

Sprint Review/ Retrospective Meeting

Sprint Planning Meeting

Backlog Refinement Meeting

Backlog Refinement Meeting

Sprint Planning Meeting

US3

US4

US5

Scrum – Discussion

Advantages [Sommerville]

- product is broken down into manageable and **understandable chunks**
- **unstable requirements** can be easily incorporated
- good **team communication** and transparency
- **customers can inspect increments** and understand how the product works
- establishes **trust** between customers and developers

Disadvantages [Sommerville]

- unclear how scale to **larger teams**
- problematic when **contract negotiation** is required (as customer pays for development time rather than set of requirements)
- requires **continuous customer input**
- **tacit knowledge** not available during maintenance (of long-life systems)
- detailed documentation required for **external regulation** and **outsourcing**

Software Process Management

Issues

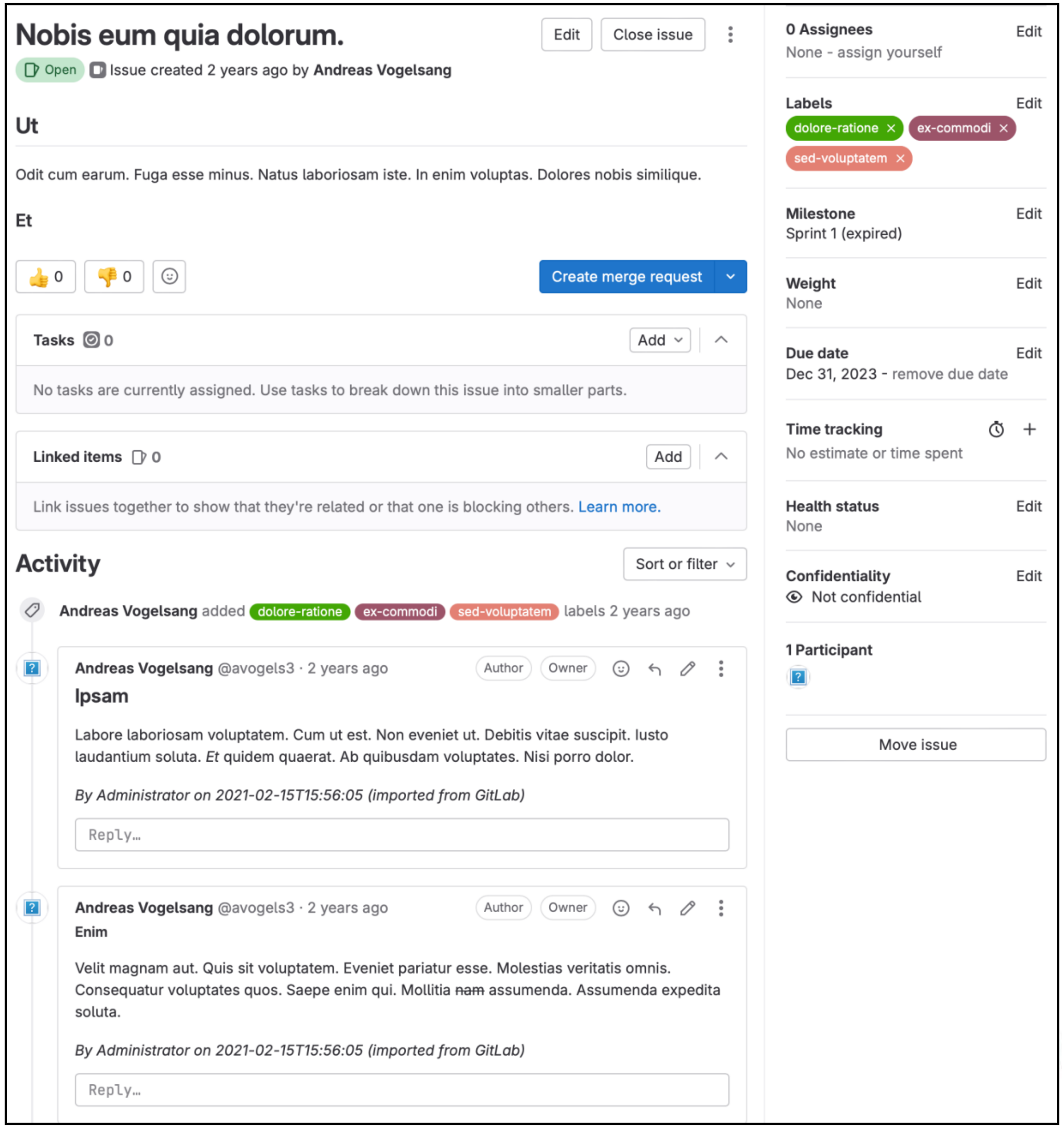
Issues

- Central container for all information related to a work item (e.g., a new feature, a bug fix, a code refactoring)
- Granularity: should be implementable within one sprint
- May be further characterized by *labels.*
- May be part of a larger container (e.g., *epic*)
- May be linked to other *issues*
- May consist of smaller *tasks*

Issue Hierarchies

To plan larger projects, issues may be grouped into containers

- Epic: Strategic planning (> 6 months)
- Feature: Increment planning (<3 months)
- Issues: Sprint Planning (2-4 weeks)



Nobis eum quia dolorum.

Open · Issue created 2 years ago by **Andreas Vogelsang**

Ut

Odit cum earum. Fuga esse minus. Natus laboriosam iste. In enim voluptas. Dolores nobis similique.

Et

Tasks 0

No tasks are currently assigned. Use tasks to break down this issue into smaller parts.

Linked items 0

Link issues together to show that they're related or that one is blocking others. Learn more.

Activity

Andreas Vogelsang added dolore-ratione ex-commodi sed-voluptatem labels 2 years ago

Andreas Vogelsang @avogels3 · 2 years ago Author Owner

Ipsam

Labore laboriosam voluptatem. Cum ut est. Non eveniet ut. Debitis vitae suscipit. Iusto laudantium soluta. *Et* quidem quaerat. Ab quibusdam voluptates. Nisi porro dolor.

By Administrator on 2021-02-15T15:56:05 (imported from GitLab)

Reply…

Andreas Vogelsang @avogels3 · 2 years ago Author Owner

Enim

Velit magnam aut. Quis sit voluptatem. Eveniet pariatur esse. Molestias veritatis omnis. Consequatur voluptates quos. Saepe enim qui. Mollitia ~~nam~~ assumenda. Assumenda expedita soluta.

By Administrator on 2021-02-15T15:56:05 (imported from GitLab)

Reply…

0 Assignees Edit
None - assign yourself

Labels Edit
dolore-ratione ✕ ex-commodi ✕ sed-voluptatem ✕

Milestone Edit
Sprint 1 (expired)

Weight Edit
None

Due date Edit
Dec 31, 2023 - remove due date

Time tracking +
No estimate or time spent

Health status Edit
None

Confidentiality Edit
Not confidential

1 Participant

Issue Board (Information Radiators)

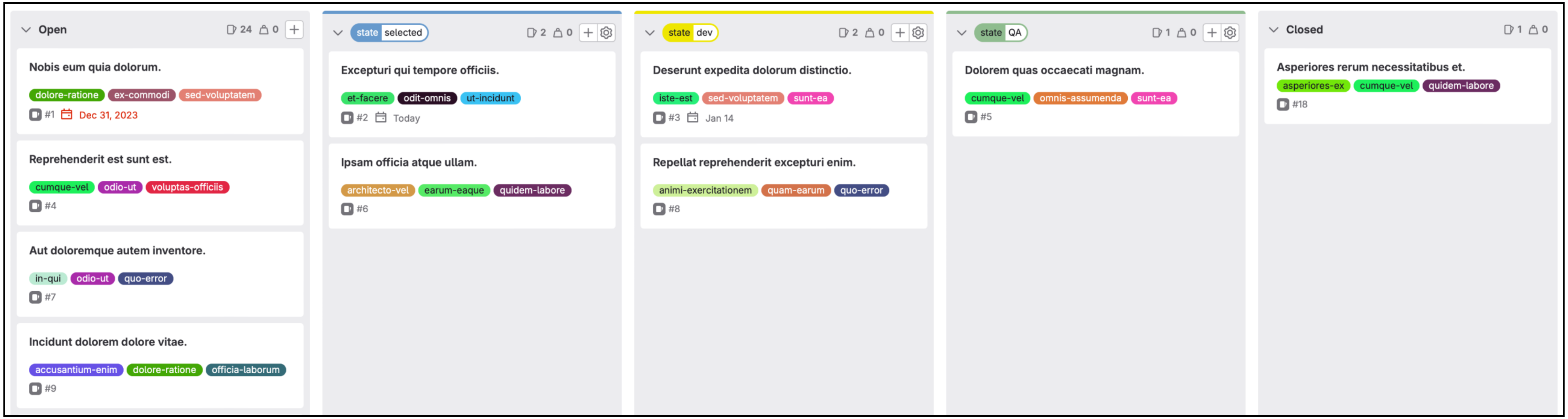
Product backlog

Issues selected for the current sprint

Issues currently in development

Issues currently in QA (review, testing)

Done issues



Open 24 0 +

Nobis eum quia dolorum.
dolore-ratione ex-commodi sed-voluptatem
#1 Dec 31, 2023

Reprehenderit est sunt est.
cumque-vel odio-ut voluptas-officiis
#4

Aut doloremque autem inventore.
in-qui odio-ut quo-error
#7

Incidunt dolorem dolore vitae.
accusantium-enim dolore-ratione officia-laborum
#9

state selected 2 0 +

Excepturi qui tempore officiis.
et-facere odit-omnis ut-incidunt
#2 Today

Ipsam officia atque ullam.
architecto-vel earum-eaque quidem-labore
#6

state dev 2 0 +

Deserunt expedita dolorum distinctio.
iste-est sed-voluptatem sunt-ea
#3 Jan 14

Repellat reprehenderit excepturi enim.
animi-exercitationem quam-earum quo-error
#8

state QA 1 0 +

Dolorem quas occaecati magnam.
cumque-vel omnis-assumenda sunt-ea
#5

Closed 1 0

Asperiores rerum necessitatibus et.
asperiores-ex cumque-vel quidem-labore
#18

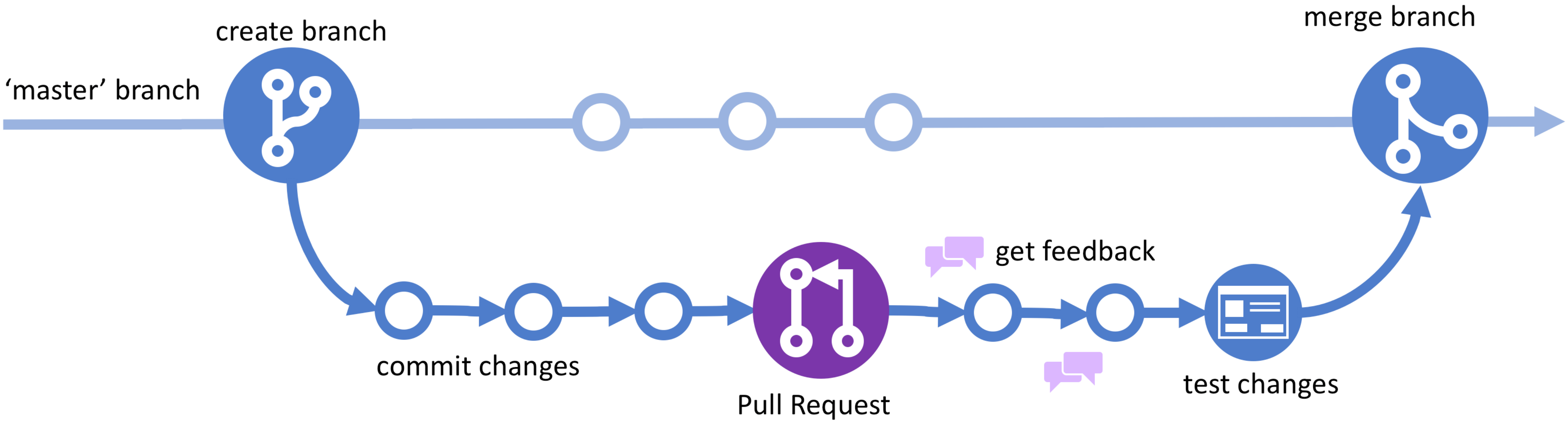
The issue board can and should be updated regularly (e.g., as part of the Daily Scrum meeting)

Feature Branches

Feature Branches

- Use of branching for issue development
- Use of **Pull Request** to manage merges

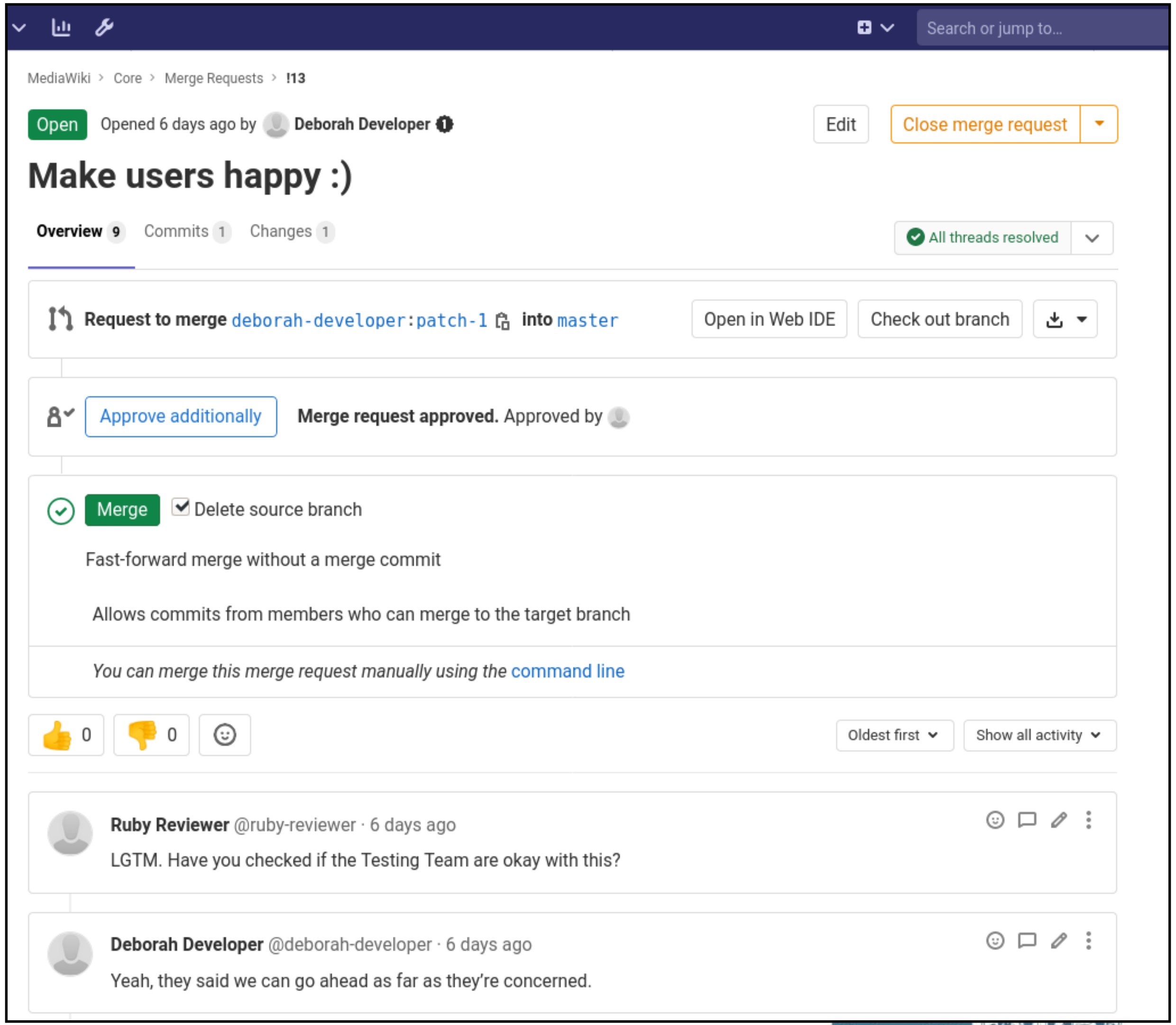
GitHub Flow



Feature Branches

Rules

- Anything in the *main* branch is deployable
- To work on an issue, create a descriptively named branch from *main* (e.g., 42-new-*order-workflow*)
- Commit to that branch locally and regularly push your work to the same named branch on the server
- When you need feedback or help, or you think the branch is ready for merging, open a **pull request** (a.k.a. merge request)
- After someone else has reviewed and signed off on the issue, you can merge the feature branch into *main*
- Once it is merged and pushed to *main*, you can and should deploy immediately



Opened 6 days ago by Deborah Developer

Make users happy :)

Request to merge `deborah-developer:patch-1` **into** `master`

Merge request approved. Approved by

☑ Delete source branch

Fast-forward merge without a merge commit

Allows commits from members who can merge to the target branch

You can merge this merge request manually using the command line

Ruby Reviewer @ruby-reviewer · 6 days ago
LGTM. Have you checked if the Testing Team are okay with this?

Deborah Developer @deborah-developer · 6 days ago
Yeah, they said we can go ahead as far as they're concerned.

Information and Process Consistency

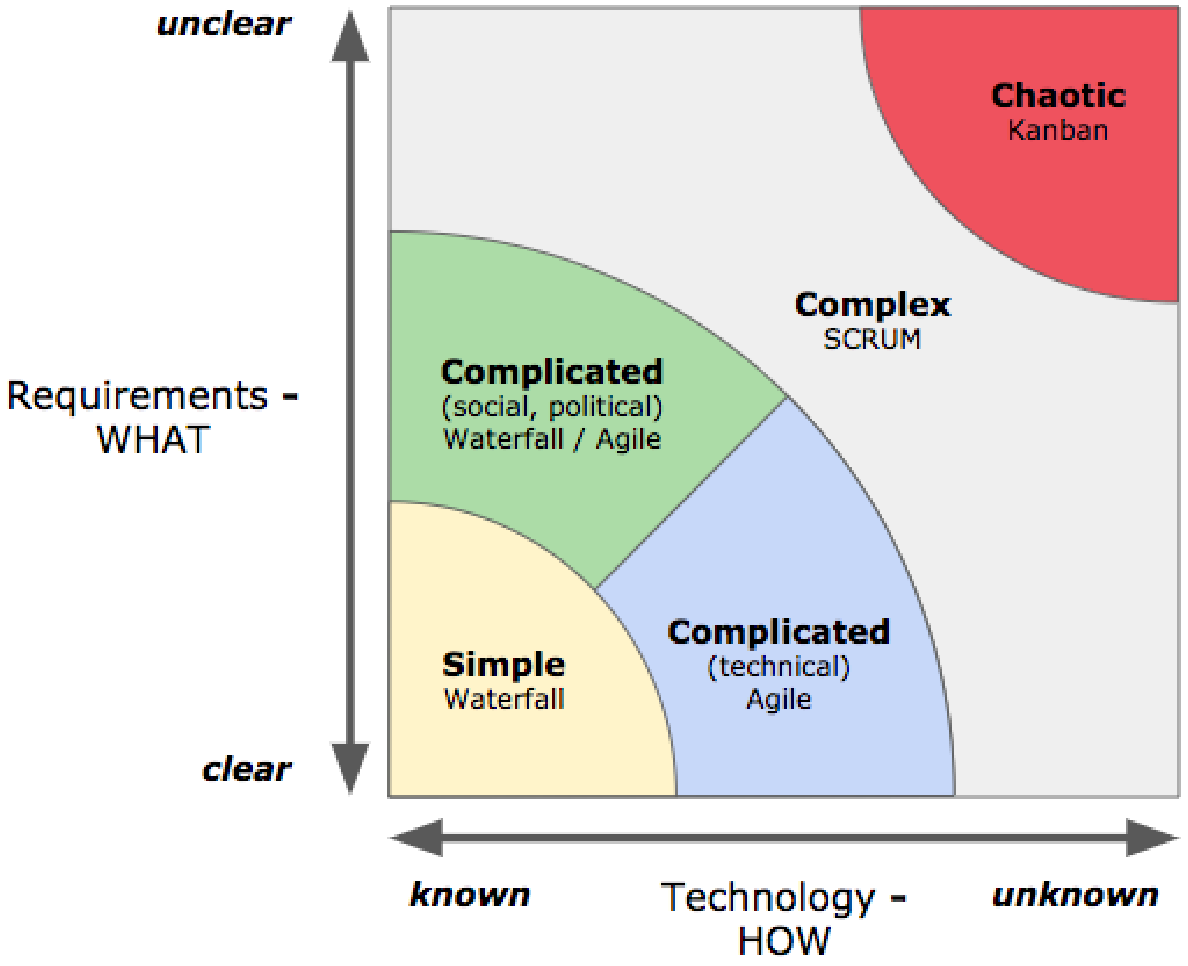
Rules and best practices (in agile processes)

- Document all information about a work item in its *issue* (e.g., requirements, acceptance criteria, responsibilities, deadlines, etc.).
- Document all discussions about the code that implements an *issue* in its corresponding *pull request*.
- Create a *branch* for each *issue* and link the branch to a corresponding *pull request*. This *branch* should contain all code changes (and only those) that contribute to the corresponding *issue*.
- Make sure that *issues*, *pull requests*, and *branches* are consistent and linked.
- Pull and branch only from *main*, never from other branches. If you need updates from other branches, these should first be merged into *main*.
- Since *branches* are linked to *issues*, they should not live longer than one sprint (because *issues* should be resolved within one sprint).

Process Selection

The Right Process for the Problem

Stacey Complexity Model



unclear

clear

Requirements - WHAT

Chaotic
Kanban

Complex
SCRUM

Complicated
(social, political)
Waterfall / Agile

Complicated
(technical)
Agile

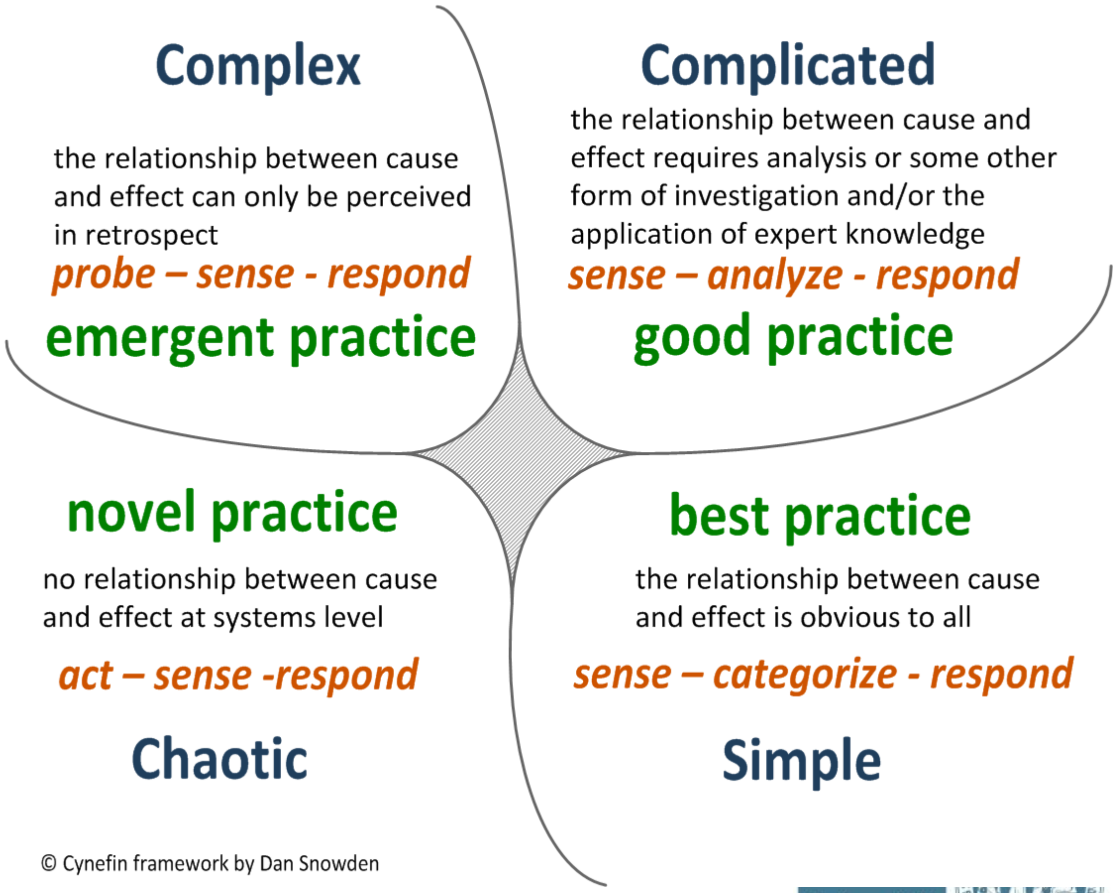
Simple
Waterfall

known

Technology - HOW

unknown

Cynefin Framework



Complex

the relationship between cause and effect can only be perceived in retrospect

probe – sense - respond

emergent practice

Complicated

the relationship between cause and effect requires analysis or some other form of investigation and/or the application of expert knowledge

sense – analyze - respond

good practice

novel practice

no relationship between cause and effect at systems level

act – sense -respond

Chaotic

best practice

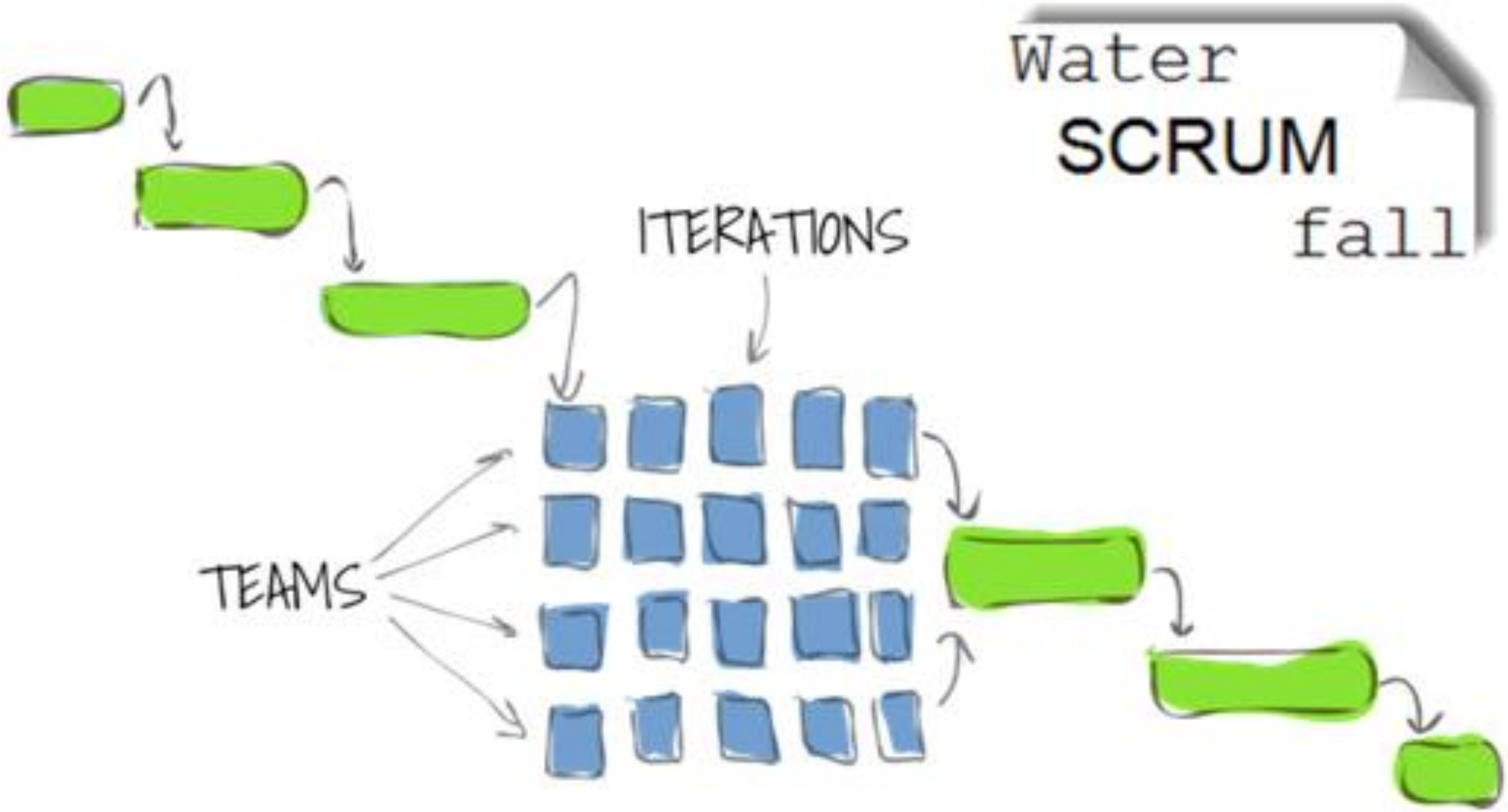
the relationship between cause and effect is obvious to all

sense – categorize - respond

Simple

© Cynefin framework by Dan Snowden

Hybrid Approaches



Summary

SE Processes

- Software development processes define a set of related activities that leads to the production of a software system
- Sequential processes emphasize “thinking before coding”.
- Often too rigid with changing requirements and environments
- Risks can be addressed via iterations
- Agile development describes a set of practices that aim to align development with the needs of the customer and deal with change in the most efficient way possible
- Feedback and changes are even desired ("Embrace Change")
- SCRUM is the most popular agile development process (roles, artifacts, meetings)
- Depending on the situation, the appropriate practices must be chosen for a project.