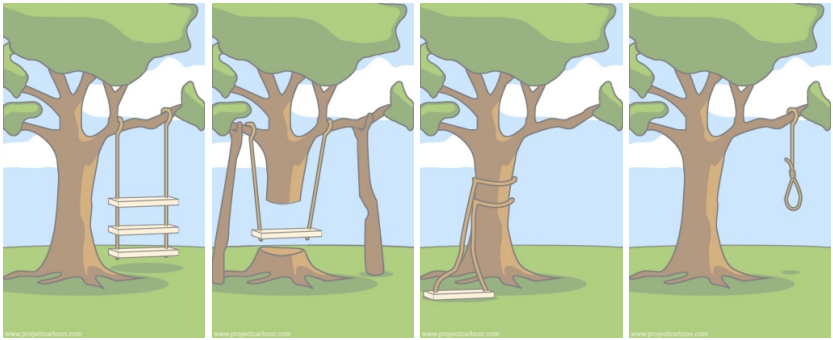
# Software Engineering

9. Development Process | Thomas Thüm | January 26, 2022

Software Engineering
Programming Languages

ulm university universität
uulm

# Development Process (Vorgehensmodelle)



how the customer explained it | how the analyst designed it | how the programmer implemented it | what the beta testers received

# Lecture Overview

1. Development Processes: The Waterfall Model
2. V-Model for Extensive Testing
3. Scrum for Agile Development

# Lecture Contents

# The Process of Food Delivery


eat, deliver, order, pay, choose, wait
– not necessarily in this order

# Questionnaire Results

1 Wirst du in 10 Jahren Software entwickeln?

| Antworten | relative Häufigkeit | absolute Häufigkeit |
|---|---|---|
| Ja | 77% | 34 |
| Nein | 23% | 10 |

2 Wirst du in 10 Jahren Software testen?

| Antworten | relative Häufigkeit | absolute Häufigkeit |
|---|---|---|
| Ja | 82% | 36 |
| Nein | 18% | 8 |

3 Wirst du in 10 Jahren Software beauftragen?

| Antworten | relative Häufigkeit | absolute Häufigkeit |
|---|---|---|
| Ja | 69% | 27 |
| Nein | 31% | 12 |

4 Wirst du in 10 Jahren Softwareanforderungen ermitteln?

| Antworten | relative Häufigkeit | absolute Häufigkeit |
|---|---|---|
| Ja | 93% | 40 |
| Nein | 7% | 3 |

5 Wirst du in 10 Jahren die Entwicklung von Software leiten?

| Antworten | relative Häufigkeit | absolute Häufigkeit |
|---|---|---|
| Ja | 57% | 24 |
| Nein | 43% | 18 |

# Software Development Process

**Motivation**

- how to **structure** the 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 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."

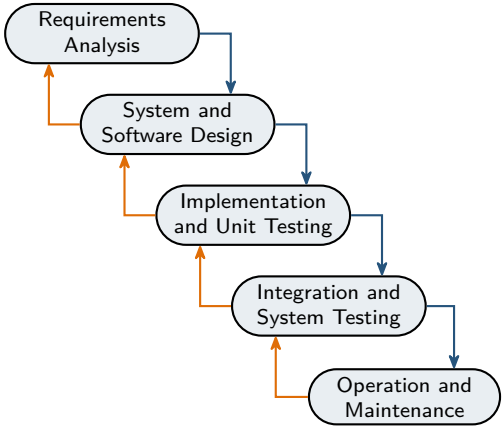
**Why Different Processes?** [Sommerville]

"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."

# Waterfall Model (Wasserfallmodell) [Sommerville]

**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



# Waterfall Model [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**

"During this stage, 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 [...]."

# Waterfall Model – Discussion [Sommerville]

**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 has to 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

# Development Processes: The Waterfall Model

**Lessons Learned**

- Software development process: motivation and goal
- Waterfall model: phases, results, example domains, advantages and disadvantages
- Further Reading: Sommerville, Chapter 2 Software Processes

**Practice**

- See Moodle
- Answer the quiz in Moodle to track your learning progress

# Lecture Contents

1. Development Processes: The Waterfall Model

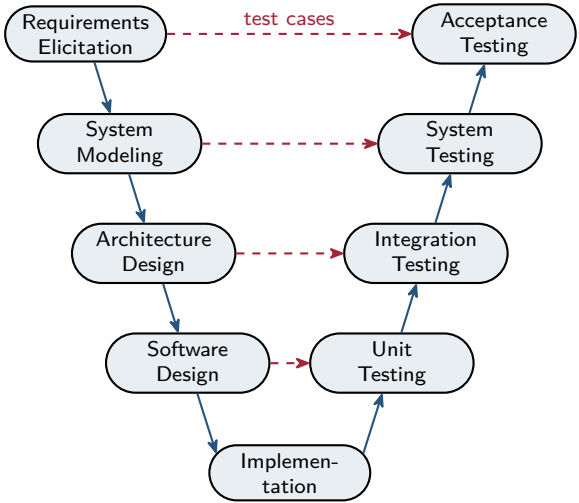
2. V-Model for Extensive Testing
   - V-Model
   - Stages of Testing
   - V-Model – Discussion
   - Lessons Learned

3. Scrum for Agile Development

# V-Model

**V-Model** [Ludewig and Lichter]

- 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 (Teststufen) [Ludewig and Lichter, Sommerville]

## 1. Unit Testing (Komponententest)

- 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 (Integrationstest)

- 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 (Systemtest)

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

## 4. Acceptance Testing (Abnahmetest)

- 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

# V-Model – Discussion [Ludewig and Lichter]

**Advantages**

- quality assurance in several testing stages
- completeness helps to not miss activities
- V-model 97/XT are widely applicable (e.g., hardware, incremental)

**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

**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

# V-Model for Extensive Testing

**Lessons Learned**

- V-model
- Testing stages: unit testing, integration testing, system testing, acceptance testing
- Further Reading: Ludewig and Lichter, Chapter 10.3 (Das V-Modell) Sommerville, Chapter 2.2.3 Software Validation and Chapter 8.1 Development Testing

**Practice**

- See Moodle
- Choose two of the four testing phases.
- Give an example of a fault that can be found in one of selected phases.
- Explain whether it could and should have been found in the other selected phase.
- Upload example and explanation to Moodle

# Lecture Contents

1. Development Processes: The Waterfall Model
2. V-Model for Extensive Testing
3. Scrum for Agile Development
   - Motivation for Agile Development
   - Manifesto of Agile Software Development
   - Principles behind the Agile Manifesto
   - Scrum
   - Burndown Chart
   - Scrum: Roles and Terms
   - Scrum – Discussion
   - Lessons Learned

# Motivation for Agile Development

**Motivation** [Sommerville]

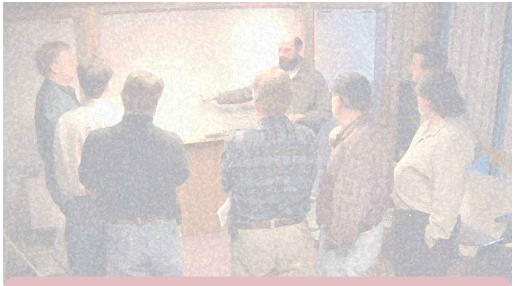
- businesses operate globally and in a rapidly changing environment
- software is part of almost all business operations
- new software has to be developed quickly
- often infeasible to derive a complete set of stable requirements
- plan-driven process models (e.g., waterfall) deliver software long after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

# Manifesto of Agile Software Development [agilemanifesto.org]



**Ski Resort in Utah (February 2001)**

17 experts on software development:

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas

**Manifesto**

"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 behind the Agile Manifesto [agilemanifesto.org]

**Principles 1–6**

1. "Our highest priority is to **satisfy the customer** through early and continuous delivery of valuable software.
2. **Welcome changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
3. **Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must **work together daily** throughout the project.
5. Build projects around **motivated individuals**. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation**."

**Principles 7–12**

7. "**Working software** is the primary measure of progress.
8. Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to **technical excellence and good design** enhances agility.
10. **Simplicity**–the art of maximizing the amount of work not done–is essential.
11. The best architectures, requirements, and designs emerge from **self-organizing teams**.
12. At regular intervals, the **team reflects** on how to become more effective, then tunes and adjusts its behavior accordingly."

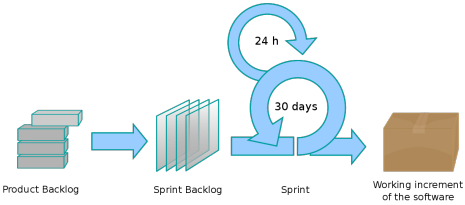
# Scrum

## User Story [Sommerville]

- a scenario of use that might be experienced by a system user
- aka. **story card** as user stories are sometimes written on physical cards
- user stories are prioritized by the customer
- subset of all user stories is chosen for the next iteration
- used in many agile methods

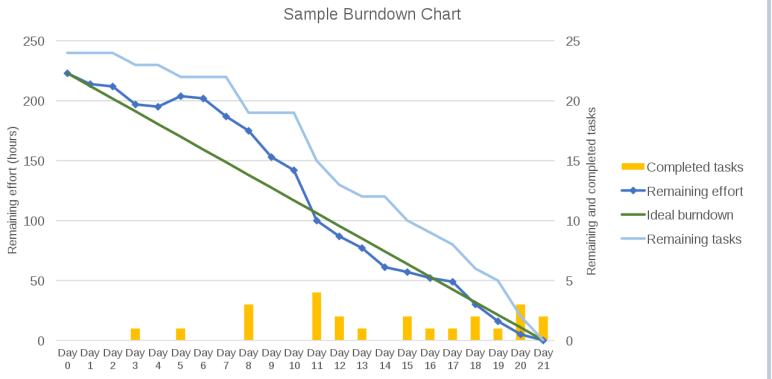
## Scrum [Sommerville]

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



# Burndown Chart

**Burndown Chart: used by scrum master to track progress**



# Scrum: Roles and Terms

**Scrum Roles** [Sommerville]

"**Development team**: A self-organizing group of software developers, which should be **no more than seven people**. They are responsible for developing the software and other essential project documents.

**Product owner**: An individual (or possibly a small group) whose job is to identify product features or requirements, prioritize these for development, and **continuously review the product backlog** to ensure that the project continues to meet critical business needs. The product owner can be a customer but might also be a product manager in a software company or other stakeholder representative.

**Scrum master**: The scrum master is responsible for ensuring that the scrum process is followed and **guides the team** in the effective use of scrum. He or she is responsible for interfacing with the rest of the company and for ensuring that the scrum team is not diverted by outside interference."

**Scrum Terms** [Sommerville]

"**Potentially shippable product increment**: The software increment that is delivered from a sprint. The idea is that this should be potentially shippable, which means that it is in a **finished state** and no further work, such as testing, is needed to incorporate it into the final product.

**Product backlog**: This is a list of **to-do items** that the scrum team must tackle. They may be feature definitions for the software, software requirements, user stories, or descriptions of supplementary tasks that are needed, such as architecture definition or user documentation.

**Daily scrum**: A daily meeting (cf. stand-up meeting) of the scrum team that **reviews progress and prioritizes work** to be done that day. Ideally, this should be a short face-to-face meeting that includes the whole team.

**Sprint**: A development iteration. Sprints are usually **2 to 4 weeks** long.

**Velocity**: An estimate of how much product backlog effort a team can cover in a single sprint. Understanding a team's velocity helps them estimate what can be covered in a sprint and provides a basis for **measuring and improving performance**."

# Scrum – Discussion [Sommerville]

**Advantages**

- 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**

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

# Scrum for Agile Development

**Lessons Learned**

- Motivation for agile development
- Agile Manifesto (four values, twelve principles)
- User stories
- Scrum: roles, terms, discussion
- Further Reading: Sommerville, Chapter 3 Agile Software Development

**Practice**

- See Moodle
- What did you learn? Formulate 1–2 questions and post them in Moodle.
- Check your learning progress! Answer 1–2 questions of your colleagues.