Welcome to the lecture “Advanced Software Engineering”

Part 2

Dr. André Fachat
d228239@student.dhbw-mannheim.de

Lecture at 14.10.2022

IBM **Consulting**

Recap Lecture 01

What is Software Engineering

Software Development Process

- Role of the IT Architect
- Main Work products

Lecture Content

1. Introduction
2. Setup Course Project
3. SW Development Process (cont‘d)
4. Requirements Engineering
5. Functional Requirements
6. Non-Functional Requirements
7. Domain-Driven Design
8. Architecture
9. Analysis and Design
10. Design-Heuristics
11. Refactoring
12. Software Quality
13. Software Management

Software Development Process

What different types of SW development processes do you know?

Software Development Process

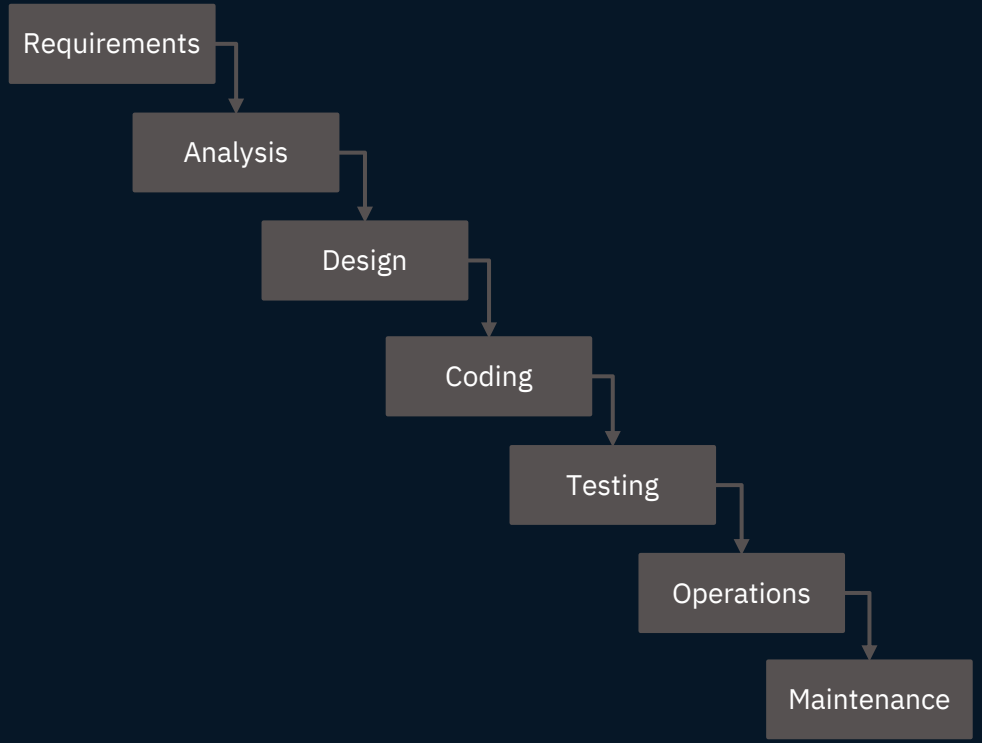
What different types of SW development processes do you know?

- Waterfall
- Agile
- SAFe ®
- Scrum
- Garage
- …

Waterfall

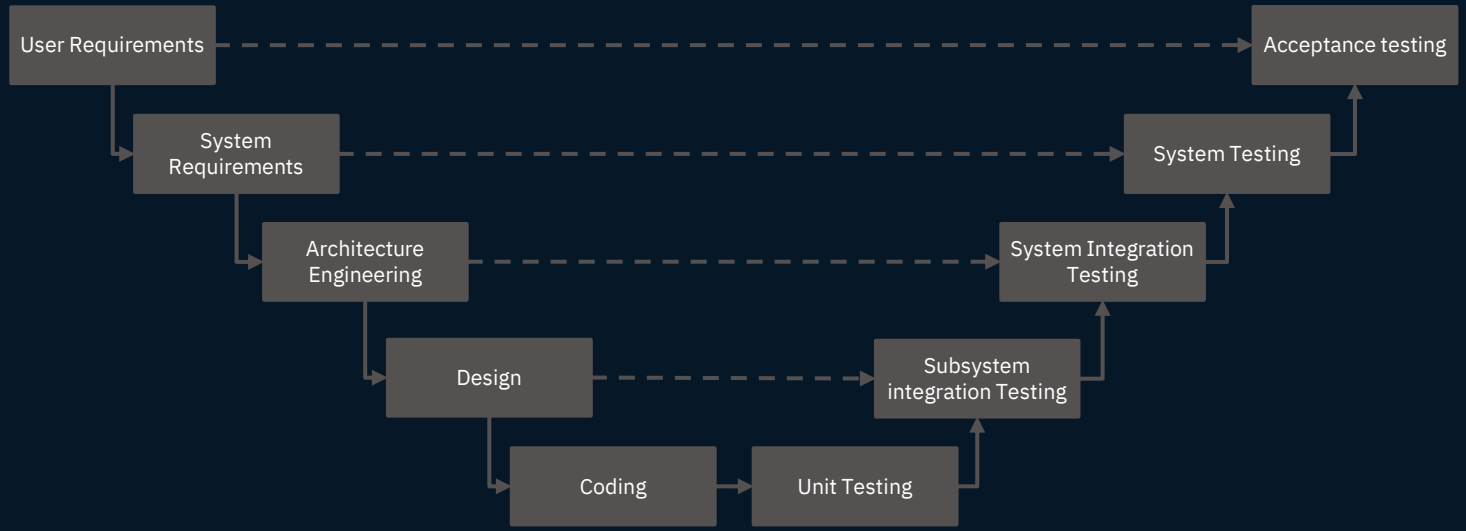
Sequential phases of software development

Each phase depends on the outcome of the previous phase(s)



Waterfall examples

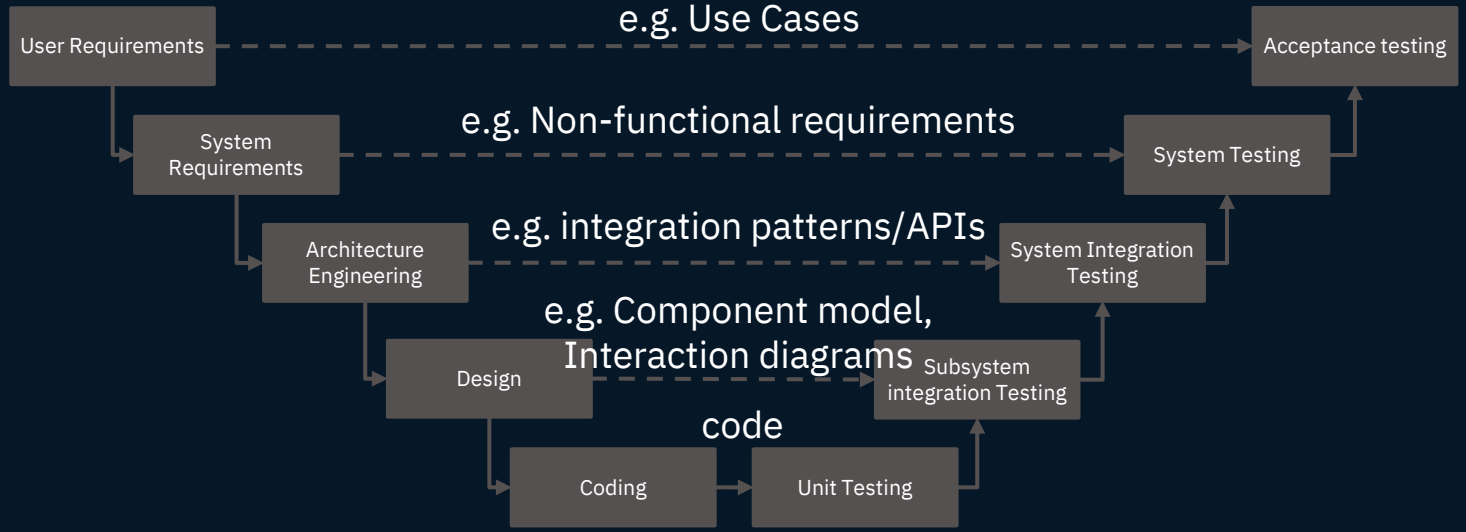
V-Model



Waterfall examples

V-Model

Each phase defines its own work products;
Output of initial phases provides base for corresponding test cases



Waterfall discussion

Typically:

- 30% Requirements & Design
- 30% Coding
- 30% Testing

Many variations have been defined & used, mostly introducing iterations

Advantages

- Easily understandable
- Focus also on documentation, supporting changing work force
- Outcome is known from the start

Disadvantages

- Not flexible if requirements change
- If end users don‘t accept the system, costly changes are required (if at all possible)

Agile

Agile Principles?

https://agilemanifesto.org/

Agile

Agile Manifesto 2001

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

https://agilemanifesto.org/
https://www.atlassian.com/agile/manifesto

Agile Processes

Scrum

Kanban

Agile at scale

Spotify model

…

https://agilemanifesto.org/

Agile / Scrum

Work products:

- Product Vision
- User stories

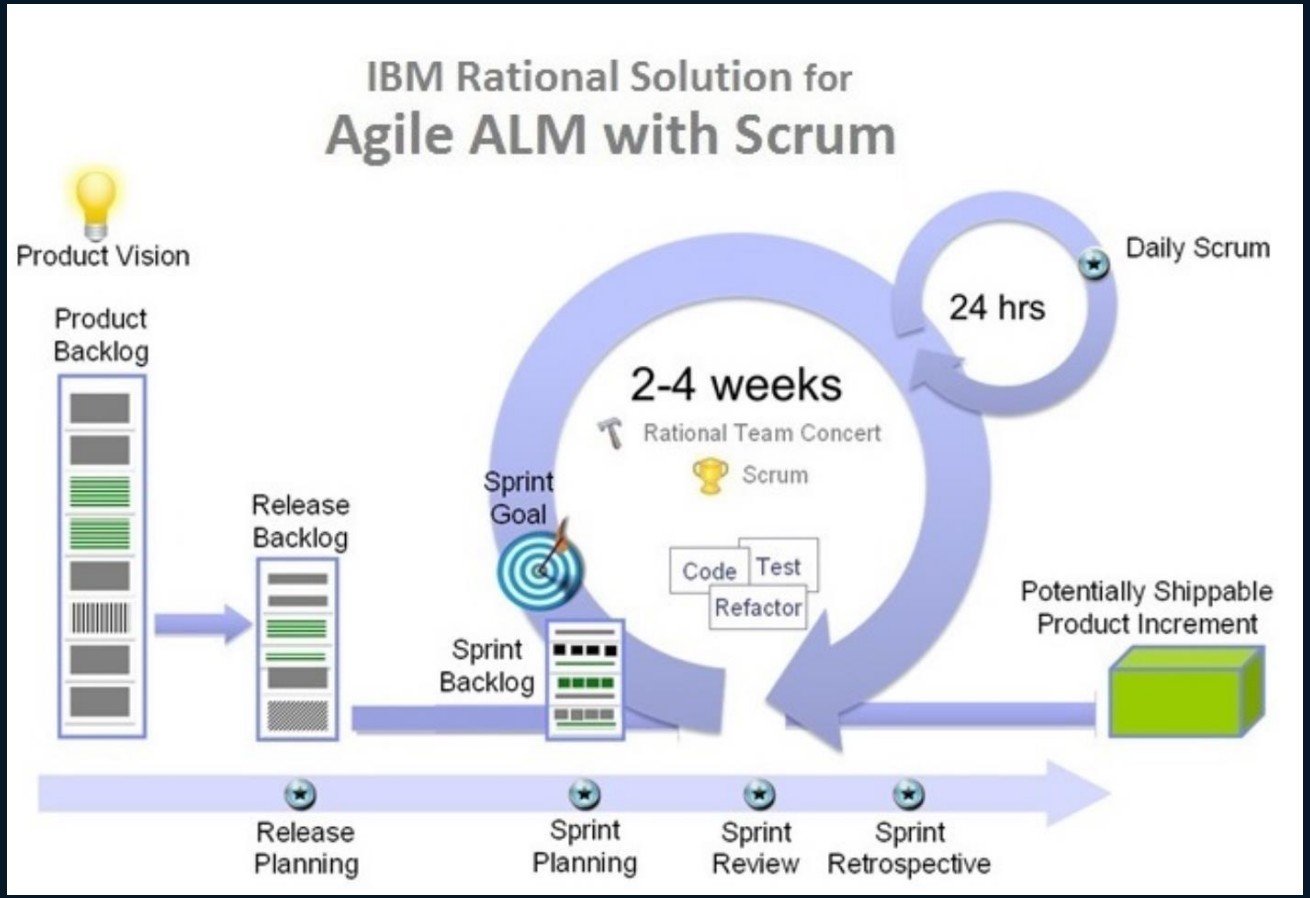
Activities:

- Release Planning
- Sprint Planning
- Backlog Refinement
- Daily Scrum
- Sprint Review
- Sprint Retrospective

Roles:

- Product Owner
  - Proxy for the users of the system and the client
- Scrum Master
  - Guide the team through the scrum process
- Development Team
  - Cross-functional, devs, designers, tester, …
  - Self-organized

Agile / Scrum


IBM Rational Solution for
Agile ALM with Scrum
Product Vision
Product Backlog
Release Backlog
Sprint Goal
Sprint Backlog
2-4 weeks
Rational Team Concert
Scrum
Code
Test
Refactor
24 hrs
Daily Scrum
Potentially Shippable Product Increment
Release Planning
Sprint Planning
Sprint Review
Sprint Retrospective

Agile / Scrum

Focus on:

- self-organization and cross-functional teams
- Collaboration through daily stand-up meetings
- Focus on delivering in short time
- Delivery after each sprint

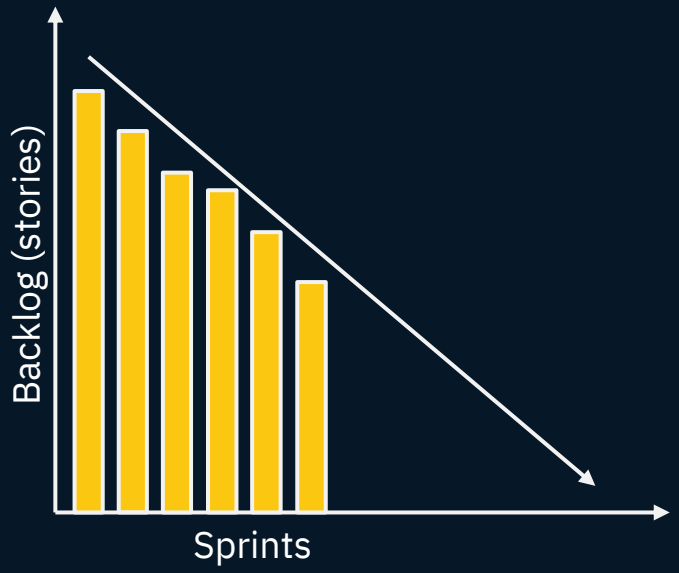
Tools:

- Backlog
  - E.g. github issues
- Burndown-Charts
  - Remaining work over time
- Task Board
- Velocity
  - Number of stories per sprint

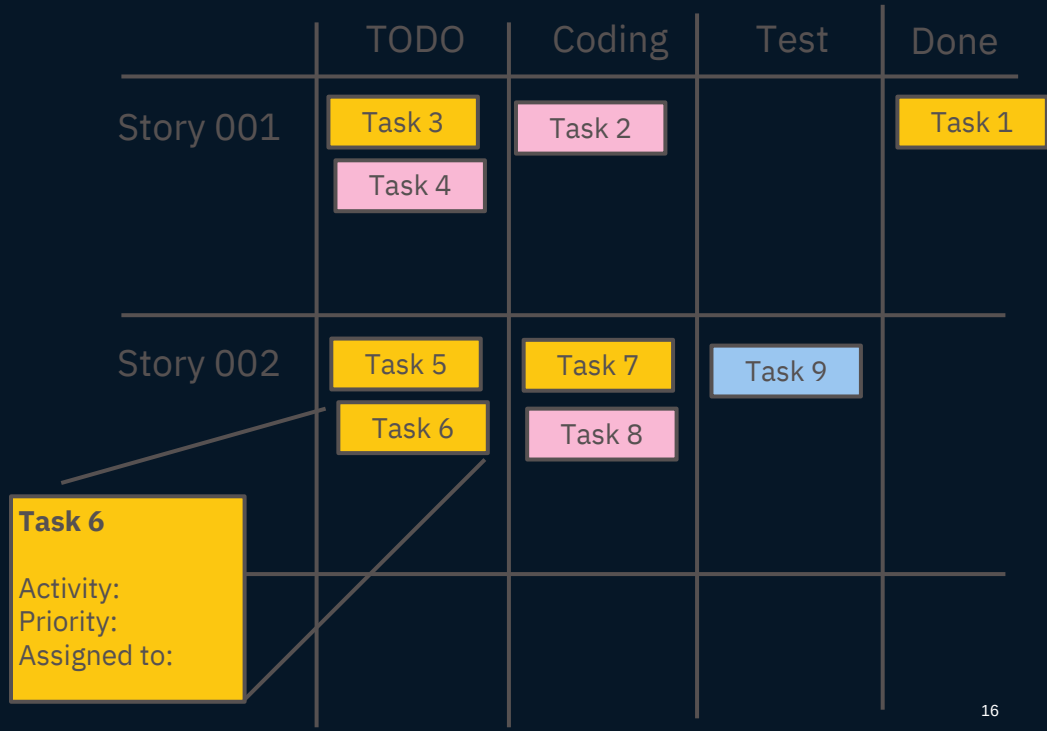
https://mindmajix.com/agile-vs-scrum

Agile / Scrum

Burndown chart



Task board



Agile / Kanban

Production line approach

Visualizing the work

Main Tool: Kanban board

No defined roles

Originating in the Automobile Industry (Toyota) to optimize Quality and Throughput, while reducing parts inventory.

In Kanban, tasks are „pulled“ from the end of the production line

Limits on the “Work in progess” for each state, to focus on getting work done

-> shows where the bottlenecks are

https://www.atlassian.com/agile/kanban

Agile / Kanban

Focus on:

- Delivery continuously
- Reduce cycle time

Tools:

- Kanban board

https://mindmajix.com/agile-vs-scrum

Agile

Definition of Ready

- Story needs to be properly defined before sprint

Definition of Done

- When is a story implementation finished

Delivering value quickly

- Continuously or after each sprint

Adapt to change quickly

- See it as „scientific experiment“ – try a functionality / user experience etc, see if it works, and learn from it
- Use Metrics to measure in an objective way

Agile „vs“ Waterfall

Both approaches can be managed well or not

The approach is often prescribed by the client.

Agile Principles can be followed in a Waterfall model as well – esp. people over process

Many blended approaches can be used, e.g. “iterative” and/or “incremental” modifications of waterfall

https://www.forbes.com/advisor/business/agile-vs-waterfall-methodology/

Agile „vs“ Waterfall

Following the agile principles is key, even in a waterfall project,

Where you can use the waterfall methods and tools (like change requests) „just“ as documentation

If you strictly just follow the process instead, you can even ruin an agile project…

People over Processes!

Customer collaboration over contract negotiation!

Responding to change over following the plan!

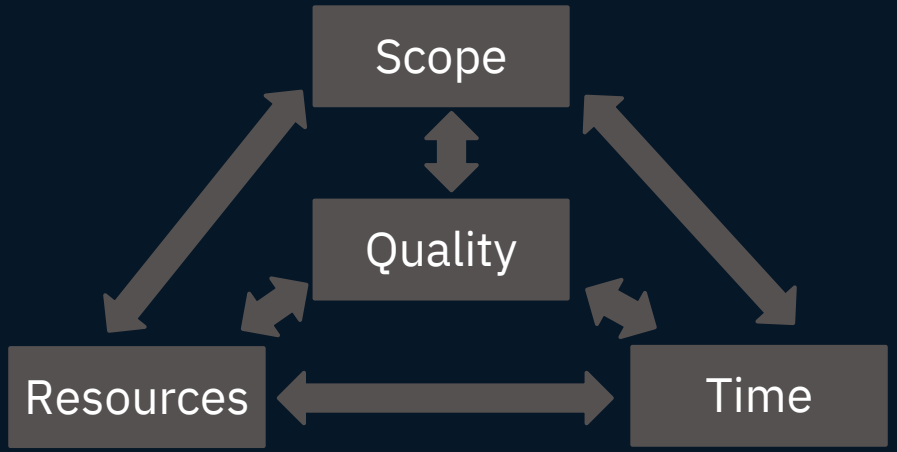
(Software over Documentation)

Watch out

Software Development „Triangle“

Scope, Resources and available time affect the quality of the project

A change in one will affect the others



Lecture Content

1. Introduction
2. Setup Course Project
3. SW Development Process (cont‘d)
4. Requirements Engineering
5. Functional Requirements
6. Non-Functional Requirements
7. Domain-Driven Design
8. Architecture
9. Analysis and Design
10. Design-Heuristics
11. Refactoring
12. Software Quality
13. Software Management

Requirements

- Description or specification what a system should do
- Features and functions
- Under which conditions the functions should be performed

… to create value for the client!

IEEE Glossary:

A condition or *capability* needed by a user to solve a problem or *achieve an objective*.

A *condition or capability* that must be met or possessed by a system or system component *to satisfy a contract, standard, specification,* or other formally imposed document.

Requirements Engineering

1. Elicitation
   - Discovering and gathering requirements from stakeholders
2. Analysis
   - Logical breakdown into manageable items
   - Ensure req’s are valid
3. Specification
   - Document req’s in specific work products
4. Validation
5. Management

Do you have gathered all requirements?

Are req’s clear, complete, valid, unduplicated, consistent?

Are stakeholders committed to req’s?

Are requirements at an appropriate level?

What is the “size” of the requirement? Break them into manageable parts

Use User Stories or Use Cases for functional requirements

Requirements Engineering

Functional requirements

- WHAT the system should do

e.g.

- Business rules
- Data manipulation (transactions)
- Authentication & Authorization
- Cross-functional aspects like Logging, Audit Tracking
- Administration
- …

Non-Functional requirements

- HOW, under what Conditions a system should work

e.g.

- Number of parallel users
- Response times
- Encryption requirements
- Accessibility
- …

Requirements Engineering

What are good requirements?

Is it testable?

- can I define a test case?

Is it measurable?

- can I measure the effectiveness?

Is it complete?

- is the scope complete?

Is it clear?

- Will developers understand it?

Is it unambiguous?

- Will they understand it the right way?

Is it consistent?

- Do the requirements fit together?

Functional Requirement Work products

Waterfall-ish

- Business requirements
- Use Cases
- Business Rules

Agile

- Epics
- User Stories

WP: Business Requirement

- A high level definition of a business value to be achieved with the system

E.g. a high level process like „order-to-cash“ that is subsequently broken down into manageable steps or user interactions

WP: Use Case

- A common understanding of the system behaviour,

used to

- Design elements supporting this behaviour
- Identify test cases
- Plan and assess work
- Write user Documentation

Attributes:

- ID
- Name
- Brief description
- Actors
- Key scenarios
- Flow of events, incl. Subflows and alternatives
- Pre- and post-conditions
- Special requirements (e.g. link to specific NFR)
- Extension points

WP: Use Case Example

1 Brief Description

This use case describes how the Bank Customer uses the ATM to withdraw money to his/her bank account.

2 Actors

2.1 Bank Customer

2.2 Bank

3 Preconditions

There is an active network connection to the Bank.

The ATM has cash available.

4 Basic Flow of Events

1. The use case begins when Bank Customer inserts their Bank Card.
2. Use Case: Validate User is performed.
3. The ATM displays the different alternatives that are available on this unit. [See Supporting Requirement SR-xxx for list of alternatives]. In this case the Bank Customer always selects ???Withdraw Cash???.
4. The ATM prompts for an account. See Supporting Requirement SR-yyy for account types that shall be supported.
5. The Bank Customer selects an account.
6. The ATM prompts for an amount.
7. The Bank Customer enters an amount.
8. Card ID, PIN, amount and account is sent to Bank as a transaction. The Bank Consortium replies with a go/no go reply telling if the transaction is ok.
9. Then money is dispensed.
10. The Bank Card is returned.
11. The receipt is printed.
12. The use case ends successfully.

WP: Use Case Example

5 Alternative Flows

5.1 Invalid User

If in step 2 of the basic flow Bank Customer the use case: Validate User does not complete successfully, then

1. The use case ends with a failure condition

5.2 Wrong account

If in step 8 of the basic flow the account selected by the Bank Customer is not associated with this bank card, then

1. The ATM shall display the message ???Invalid Account ??? please try again???.

2. The use case resumes at step 4.

5.3 Wrong amount

If in step 7 in the basic flow, the Bank Customer enters an amount that can't be 'created' with the kind of in the ATM (See Special Requirement WC-1 for valid amounts), then

1. The ATM shall display a the message indicating that the amount must be a multiple of the bills on hand, and ask the Bank Customer to reenter the amount.

2. The use case resumes at step 7.

5.4 Amount Exceeds Withdrawal Limit

If in step 7 in the basic flow, the Bank Customer enters an amount that exceeds the withdrawal limit (See Special Requirement WC-2 for maximum amount), then

1. the ATM shall display a warning message, and ask the Bank Customer to reenter the amount

2. The use case resumes at step 7

WP: Use Case Example

6 Key Scenarios

6.1 No Response from Bank

7 Post-conditions

7.1 Successful Completion

The user has received their cash and the internal logs have been updated.

7.2 Failure Condition

The logs have been updated accordingly.

8 Special Requirements

[SpReq:WC-1] The ATM shall dispense cash in multiples of $20.

[SpReq2:WC-2] The maximum individual withdrawal is $500.

[SpReq:WC-1] The ATM shall keep a log, including date and time, of all complete and incomplete transactions with the Bank.

WP: Use Case

- Flow of events can be textual or graphical
- Typically alternating
  - System does X
  - Actor does Y
- Special Requirements could refer to business rules

Remember:

- An external actor can also be another system

WP: Use Case Discussion

Critics:

- A very specific description of an interaction
- Prescribes a definitive order of events e.g. first Account number, then Amount
- Leaves less freedom to change / improve user experience

Advantages:

- Very clear for the client / stakeholder what can be expected
- Easy to define test cases

WP: Business Rule

A rule what to do in specific situations, that can be re-used in various places

Can be implemented in a re-usable piece of code, or by storing in and executing from a rules repository

- Decouples the process flow from the business rules decisions that can change more freely
- Must be clearly stated, concrete, not abstract

E.g.

- The maximum amount an account holder can draw from a teller
  - Initial: fixed amount of $500
  - After further development: depending on evaluation of client properties / account balance

WP: Epic

A large requirement that takes multiple sprints and potentially multiple teams on different components to implement

Broken down into multiple user stories

E.g. User Authentication

WP: User Story

A requirement that one team can implement during one sprint

„It‘s an end goal, not a feature, expressed from a [] user‘s perspective“

Properties:

- Definition of done
- Subtasks and who is responsible
- User personas (actors)

https://www.atlassian.com/agile/project-management/user-stories

WP: User Story

Links to user value

“As a <user/role> I want to <activity> so that I achieve <value>”

“A few sentences in simple language that outline the desired outcome”

“Requirements are added later, once agreed upon by the team”

Estimation items

https://www.atlassian.com/agile/project-management/user-stories

WP: User Story Example

„As a *bank customer* I want to *withdraw cash money from my account* so that I can *buy stuff in shops that don‘t take bank cards“*

WP: User Story Discussion

Critics:

- Team is responsible to refine the user story into a design and implementation. This can be a challenge, esp. for inexperienced teams
- The client only sees the result at the end of the sprint – the product owner is responsible to evaluate and challenge the solution created by the team; client involvement desirable
- Larger requirements need to be broken down into multiple stories, where later stories may require to modify code written and delivered in earlier stories. This may drive implementation and test costs

Advantages:

- Experienced Teams enjoy the challenge to be solved as a cross-functional team
- Team has the flexibility to find the best solution
- Stories keep the focus on the user

Functional Requirements Remarks

- Sometimes clients specify huge lists of functional requirements
  - Typically they are not complete, nor consistent, nor at the same level
  - Never assume the client has done the necessary refinement
  - Beware of cross-functional aspects hidden in such lists
- Client involvement is key
  - Agile or Waterfall

End of Lecture 02

Course project

- Assign groups
  - Group name
- Define application
  - Understand the business value behind the application

Groups & Projects

Team	Project
CyberLove	MemeTinder
Layla	Geschenklisten
Captain Cook	Cooking Webseite
Doctor-Docs	Patientenakte 2
Hello World	Patientenakte
Festplattenarm	Einkaufsliste

Project Presentation on 28.10.
~10 minutes each

Business Value

Actors

Architecture Overview

- Component Structure
- Concept: how to integrate external ID provider
- Concept: how to secure Database access (cred.)

Architectural Decision

- Why the used web stack was chosen (against alternatives)

Functional Requirements

- List of use Case / User Stories

Lecture Topics on 28.10.

Non-Functional Requirements

- Performance, Availability, …

Domain-Driven Design

- How to get from your requirements to a module structure