Software Engineering

Ausbildung Junior Software-Entwickler (SWE)

Dezember 2021

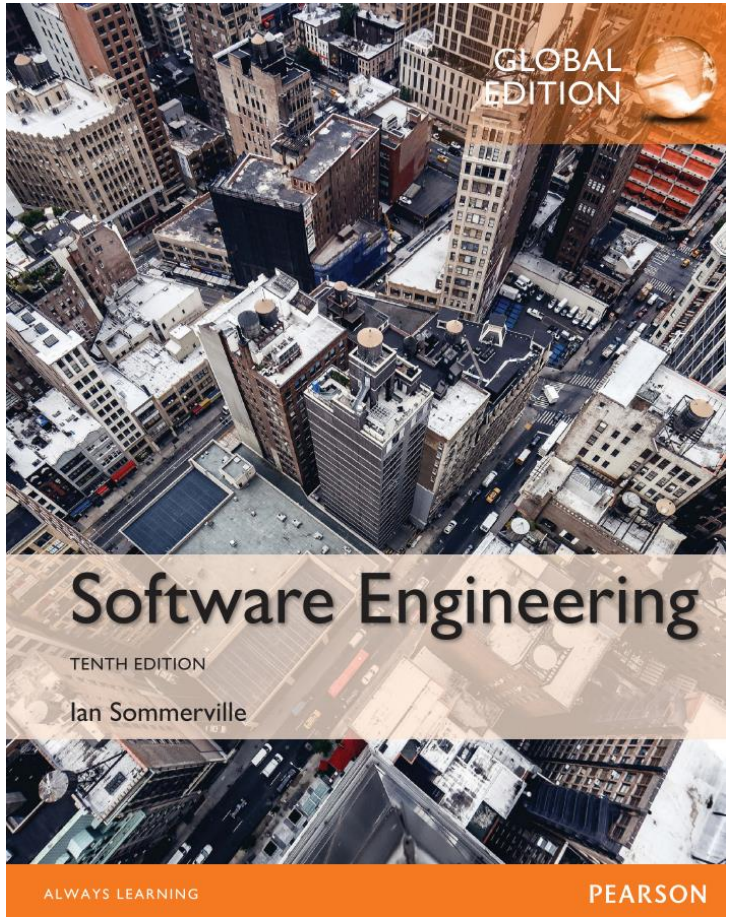
Wolfgang Kremser

- Früher: Full-Stack Webentwicklung bei Agenturen
- Heute: Forscher bei Salzburg Research
  - Product Owner
  - Systems Engineer
  - Research Software Engineer
- Forschungsgebiete:
  - Data Management, Design Science Research, Software Engineering
- wolfgang@enuff.at

Kursinhalt

- Ausgewählte Kapitel aus:
  Ian, Sommerville. 2015:
  Software Engineering, (Global Edition).
  Pearson Education.

- Praktische Übungen



Orga

- 14. & 15. Dezember Online 9:00 – 17:00
- 16. Dezember WIFI Sbg 9:00 – 17:00

- ~ 80 Minuten Einheiten
- Mittagspause: ~12:00 – 13:00

- Interaktion über Miro-Boards

- Test am 16. Dezember, ~45 Minuten Zeit

COVID Maßnahmen

- 3G wird von Kursleitung (= mir) kontrolliert
  - Bitte Nachweis und Ausweis mitnehmen
- Vor Ort FFP2 Maske im WIFI Gebäude, Kursraum und am Platz
- Gruppenbildung im Gebäude bitte vermeiden

Aufsetzen unserer Collaboration Tools

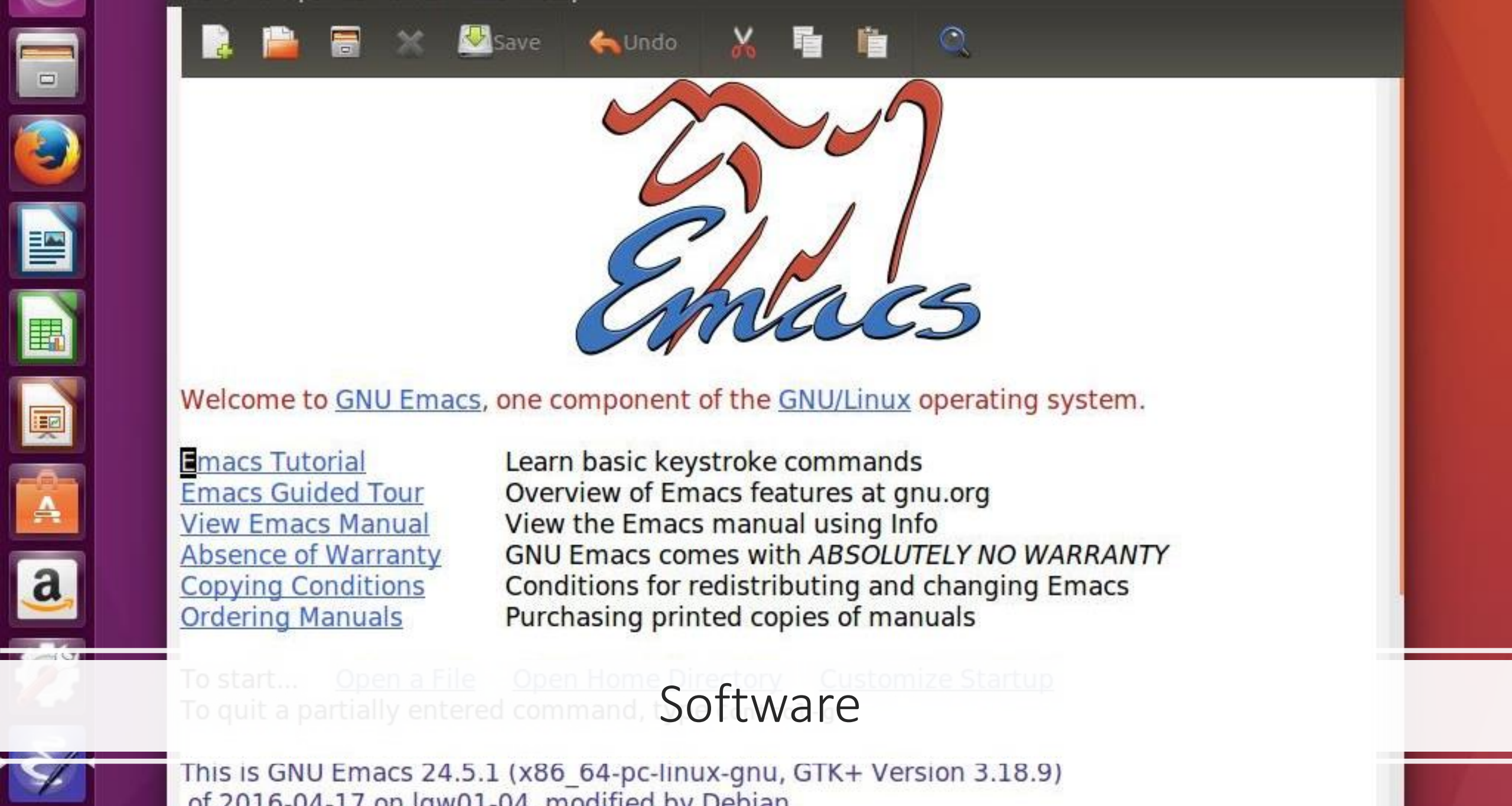
- Slack
  - https://bit.ly/3DOjEG7
- Miro Board
  - https://bit.ly/3dM1Tgg
- Github Classroom
  - https://bit.ly/3lYqDWM

Erste Übung auf Miro!

Vorstellungsrunde

Einführung Software Engineering

Wieso? Weshalb? Warum?


Save
Undo
Welcome to GNU Emacs, one component of the GNU/Linux operating system.
Emacs Tutorial
Learn basic keystroke commands
Emacs Guided Tour
Overview of Emacs features at gnu.org
View Emacs Manual
View the Emacs manual using Info
Absence of Warranty
GNU Emacs comes with ABSOLUTELY NO WARRANTY
Copying Conditions
Conditions for redistributing and changing Emacs
Ordering Manuals
Purchasing printed copies of manuals
To start... Open a File Open Home Directory Customize Startup
To quit a partially entered command, type Control-g
This is GNU Emacs 24.5.1 (x86_64-pc-linux-gnu, GTK+ Version 3.18.9)
of 2016-04-17 on lgw01-04, modified by Debian
Copyright (C) 2015 Free Software Foundation, Inc.

Software

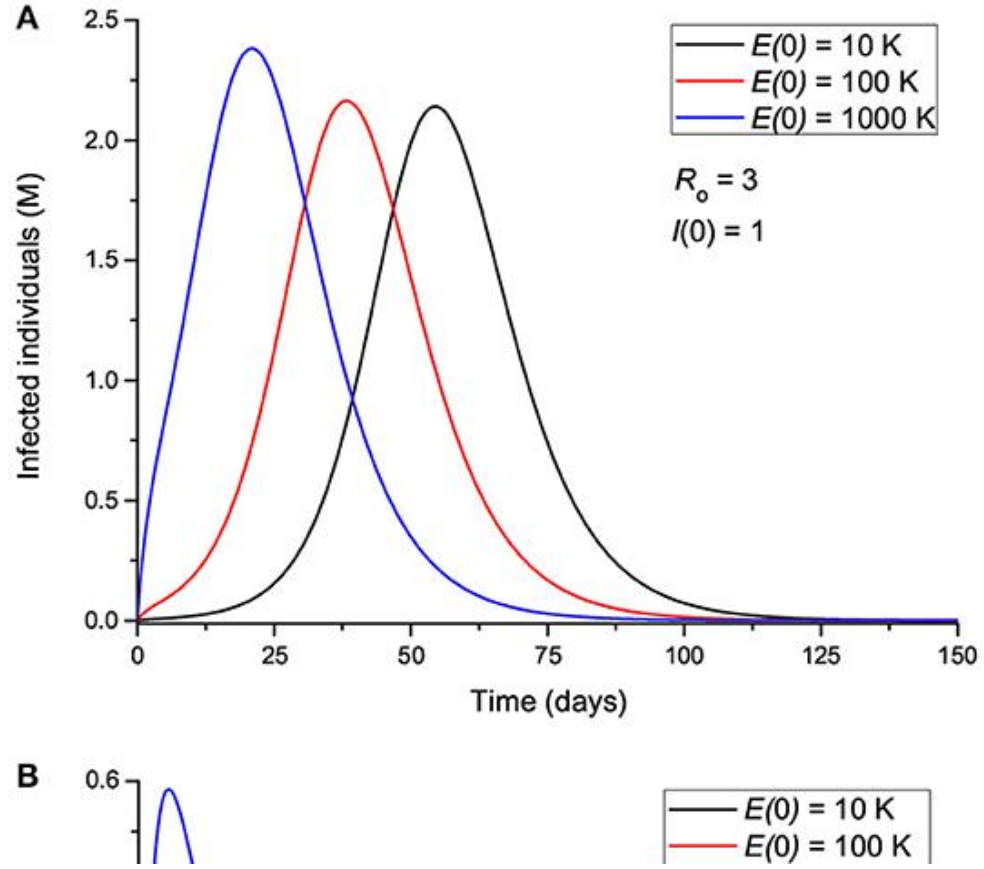
```
harbinger:~/mobapps-logs% ll *
lrwxrwxrwx 1 root root 33 2010-01-27 12:31 mobapps_access_log -> /var/log/httpd/mobapps_access_log
lrwxrwxrwx 1 root root 32 2010-01-27 12:31 mobapps_error_log -> /var/log/httpd/mobapps_error_log
harbinger:~/mobapps-logs% ll /var/log/httpd/mobapps_access_log
-rw-r--r-- 1 root root 3052499 2010-04-14 12:14 /var/log/httpd/mobapps_access_log
harbinger:~/mobapps-logs% wc /var/log/httpd/mobapps_access_log
  10747  260848 3052499 /var/log/httpd/mobapps_access_log
harbinger:~/mobapps-logs% ll /var/log/httpd/mobapps_error_log
-rw-r--r-- 1 root root 203624 2010-04-14 11:34 /var/log/httpd/mobapps_error_log
harbinger:~/mobapps-logs% wc /var/log/httpd/mobapps_error_log
   976  14451 203624 /var/log/httpd/mobapps_error_log
harbinger:~/mobapps-logs% tail /var/log/httpd/mobapps_access_log
128.32.226.135 - - [14/Apr/2010:12:14:17 -0700] "GET /favicon.ico HTTP/1.1" 301 344 "-" "Mozilla/5.0 (
Windows; U; Windows NT 5.1; en-US; rv:1.9.2.3) Gecko/20100401 Firefox/3.6.3 (.NET CLR 3.5.30729)"
128.32.226.135 - - [14/Apr/2010:12:14:20 -0700] "GET /favicon.ico HTTP/1.1" 301 344 "-" "Mozilla/5.0 (
Windows; U; Windows NT 5.1; en-US; rv:1.9.2.3) Gecko/20100401 Firefox/3.6.3 (.NET CLR 3.5.30729)"
128.32.226.135 - - [14/Apr/2010:12:14:43 -0700] "GET /~dret HTTP/1.1" 301 343 "-" "Mozilla/5.0 (Window
s; U; Windows NT 5.1; en-US; rv:1.9.2.3) Gecko/20100401 Firefox/3.6.3 (.NET CLR 3.5.30729)"
128.32.226.135 - - [14/Apr/2010:12:14:43 -0700] "GET /~dret/ HTTP/1.1" 200 1814 "-" "Mozilla/5.0 (Wind
ows; U; Windows NT 5.1; en-US; rv:1.9.2.3) Gecko/20100401 Firefox/3.6.3 (.NET CLR 3.5.30729)"
128.32.226.135 - - [14/Apr/2010:12:14:43 -0700] "GET /icons/blank.gif HTTP/1.1" 301 348 "http://mobapp
s.ischool.berkeley.edu/~dret/" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.3) Gecko/2010
0401 Firefox/3.6.3 (.NET CLR 3.5.30729)"
128.32.226.135 - - [14/Apr/2010:12:14:43 -0700] "GET /icons/back.gif HTTP/1.1" 301 347 "http://mobapps
.ischool.berkeley.edu/~dret/" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.3) Gecko/20100
401 Firefox/3.6.3 (.NET CLR 3.5.30729)"
128.32.226.135 - - [14/Apr/2010:12:14:43 -0700] "GET /icons/text.gif HTTP/1.1" 301 347 "http://mobapps
.ischool.berkeley.edu/~dret/" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.3) Gecko/20100
401 Firefox/3.6.3 (.NET CLR 3.5.30729)"
128.32.226.135 - - [14/Apr/2010:12:14:43 -0700] "GET /icons/folder.gif HTTP/1.1" 301 349 "http://mobap
ps.ischool.berkeley.edu/~dret/" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.3) Gecko/201
00401 Firefox/3.6.3 (.NET CLR 3.5.30729)"
128.32.226.135 - - [14/Apr/2010:12:14:43 -0700] "GET /icons/p.gif HTTP/1.1" 301 344 "http://mobapps.is
chool.berkeley.edu/~dret/" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.3) Gecko/20100401
```

Software

Software

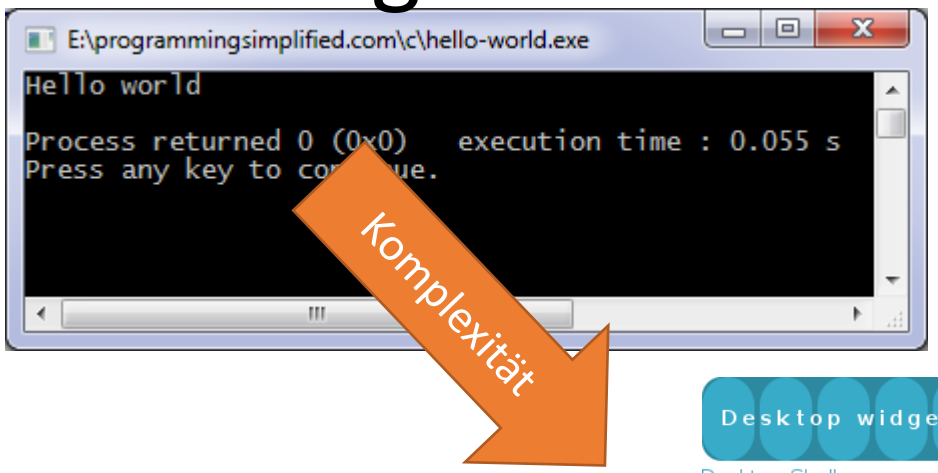

Centro di comando
1500 / 1500
Corazzata - Meccanica - Struttura

Software

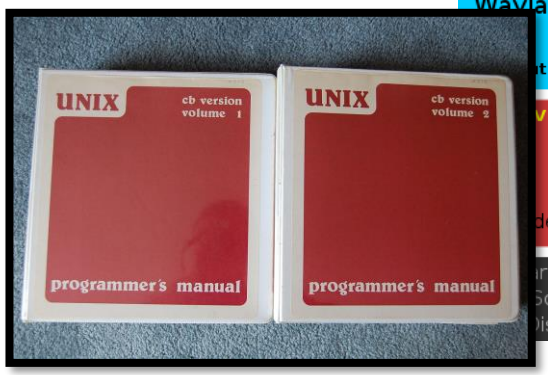

A
2.5
2.0
1.5
1.0
0.5
0.0
Infected individuals (M)
0
25
50
75
100
125
150
Time (days)
E(0) = 10 K
E(0) = 100 K
E(0) = 1000 K
R_o = 3
I(0) = 1
B
0.6
E(0) = 10 K
E(0) = 100 K

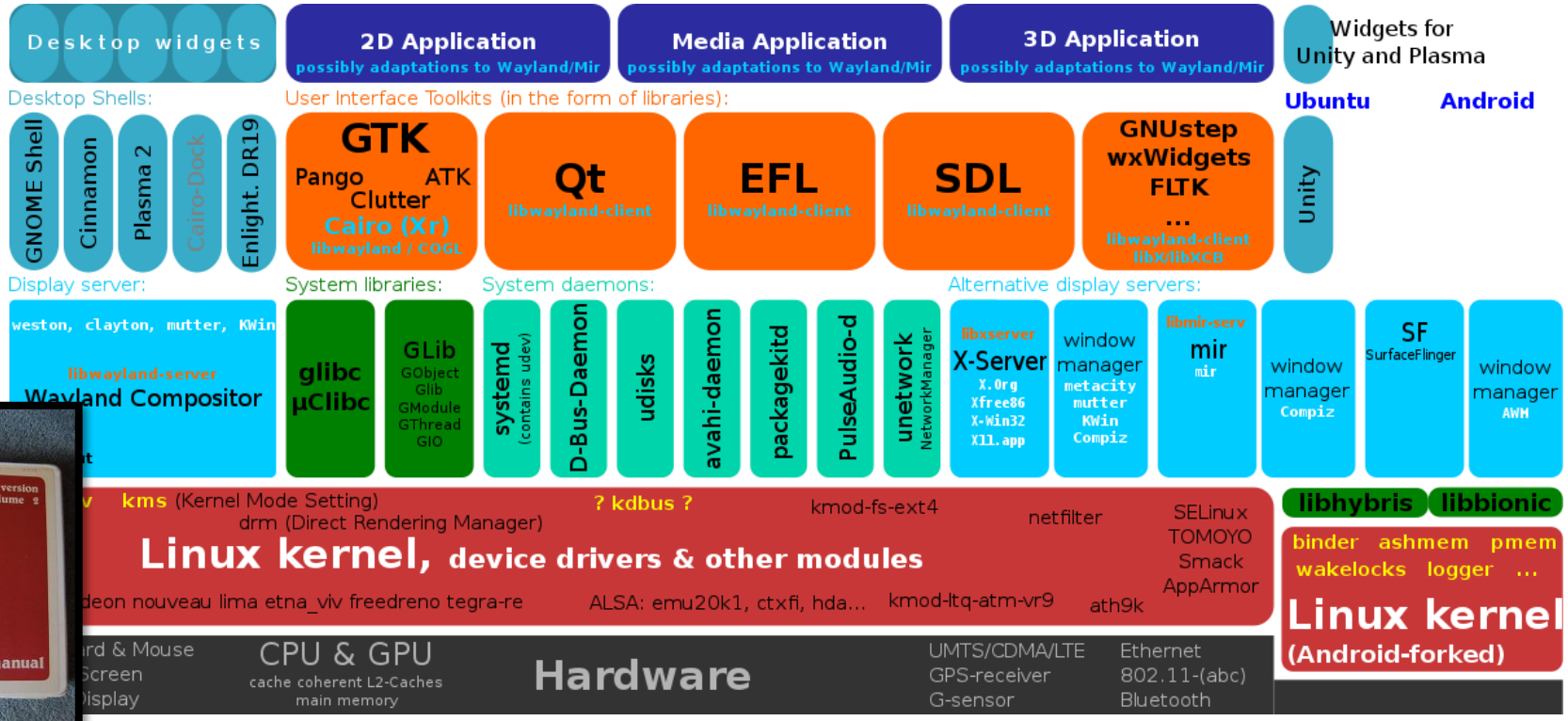
Software

Programm



Software = Programme + Dokumentation





Eigenschaften guter Software

Akzeptanz	• Zieluser versteht die Software • Kompatibel mit anderer Software des Zielusers
Verlässlichkeit / Sicherheit	• Systemfehler darf keinen Schaden anrichten • Gesichert gegen bösartigen Zugriff
Effizienz	• Zeiteffizient • Energieeffizient • CPU- / Speichereffizient
Wartbarkeit	• Erweiterbar / Veränderbar

Eigenschaften guter Software

Akzeptanz	• Zieluser versteht die Software • Kompatibel mit anderer Software des Zielusers
Verlässlichkeit / Sicherheit	• Systemfehler darf keinen Schaden anrichten • Gesichert gegen bösartigen Zugriff
Effizienz	• Zeiteffizient • Energieeffizient • CPU- / Speichereffizient
Wartbarkeit	• Erweiterbar / Veränderbar

Wir wollen gute Software bauen,
deshalb Software Engineering!

Warum scheitern Software Projekte?

- @ Miro Board

Warum scheitern Software Projekte?

- @ Miro Board

- Auflösung folgt!

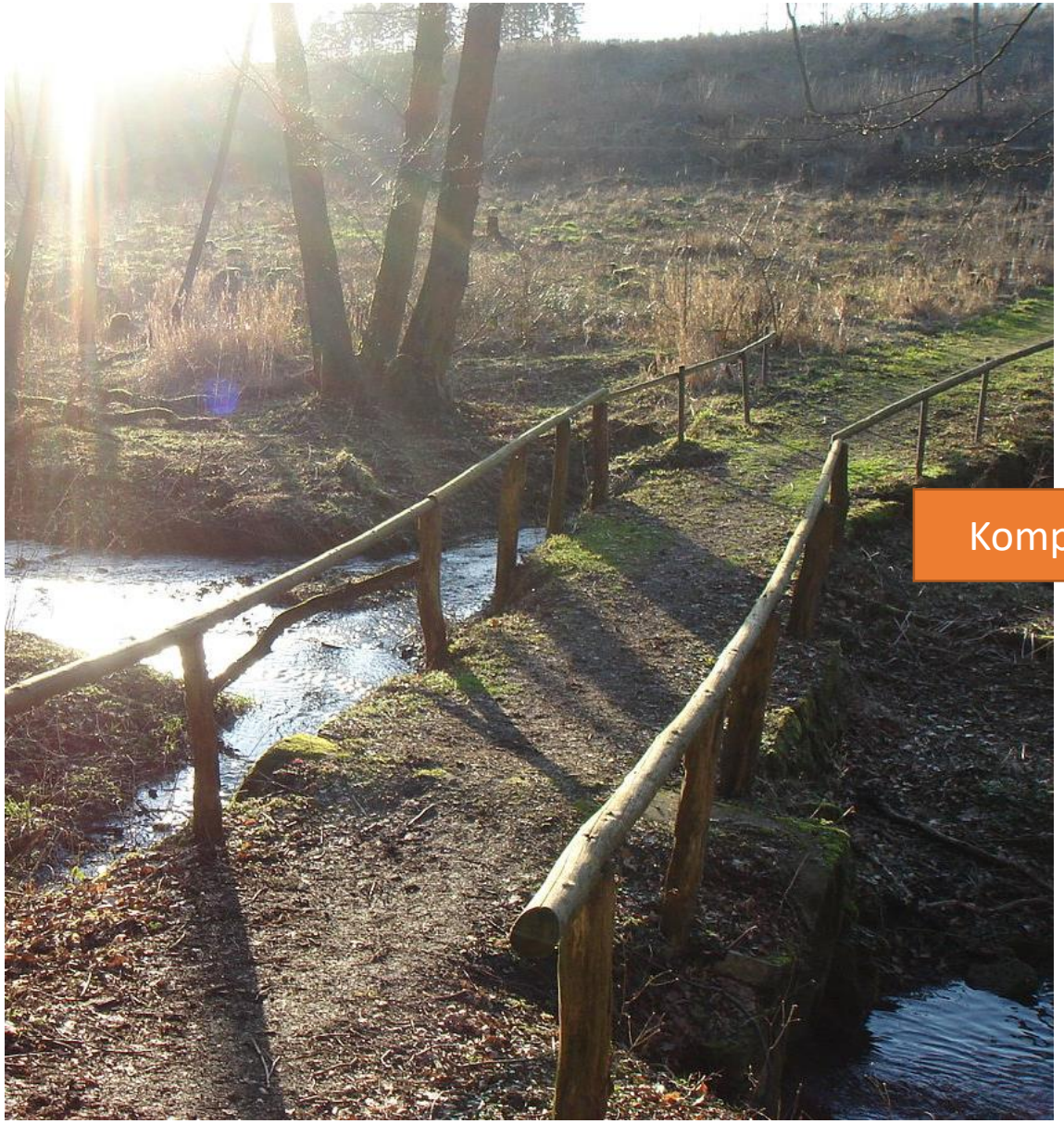
Engineering

- „Als Ingenieurwissenschaften […] werden diejenigen Wissenschaften bezeichnet, die sich mit der [bestehenden und bald möglichen] Technik beschäftigen. Zentrale Fragestellungen betreffen die Forschung und Entwicklung, **Konstruktion, Produktion und die Prüfung**.“

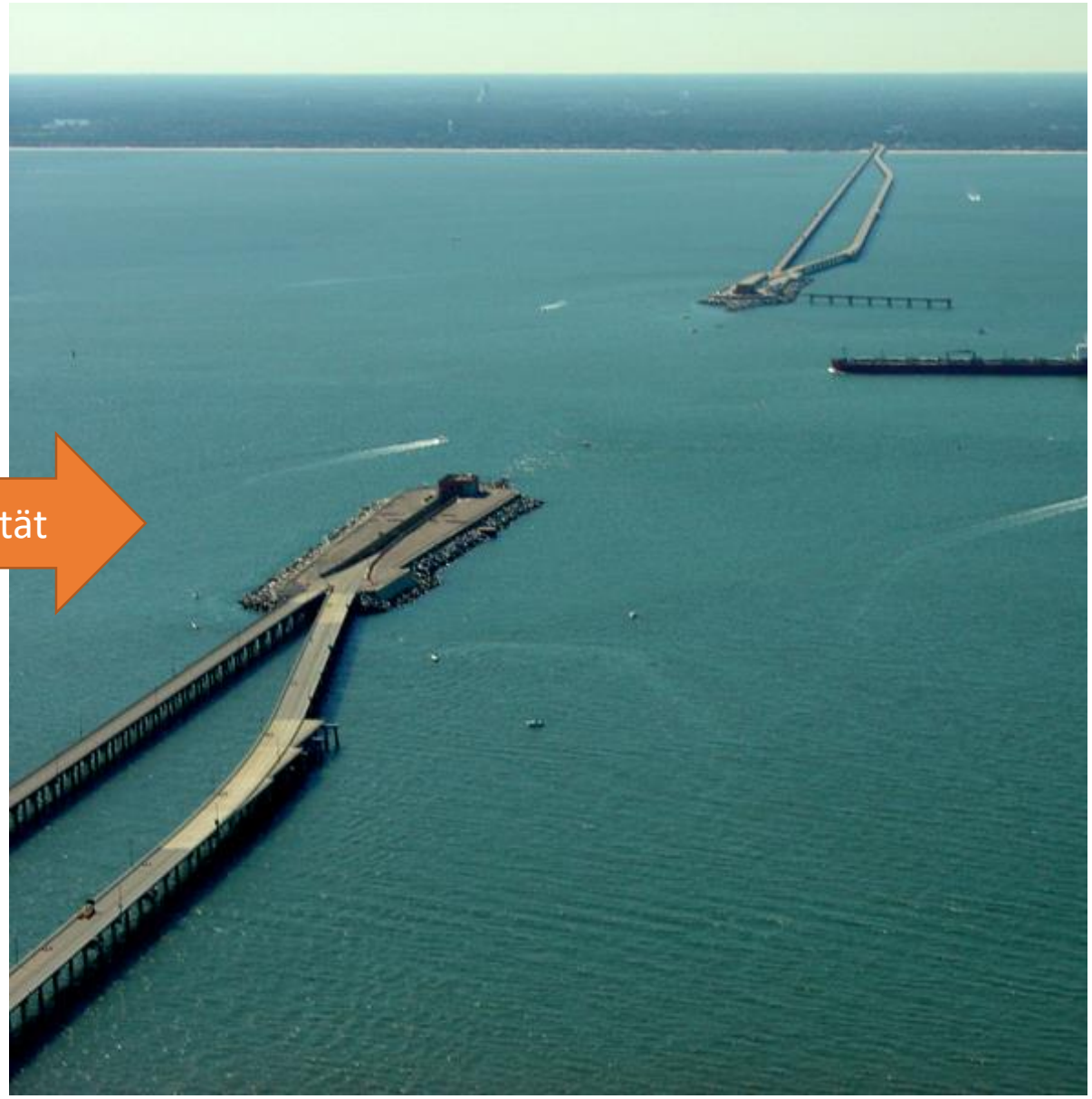
  - https://de.wikipedia.org/wiki/Ingenieurwissenschaften (emphasis mine)

- “Engineering is about getting results of the **required quality within schedule and budget.**”

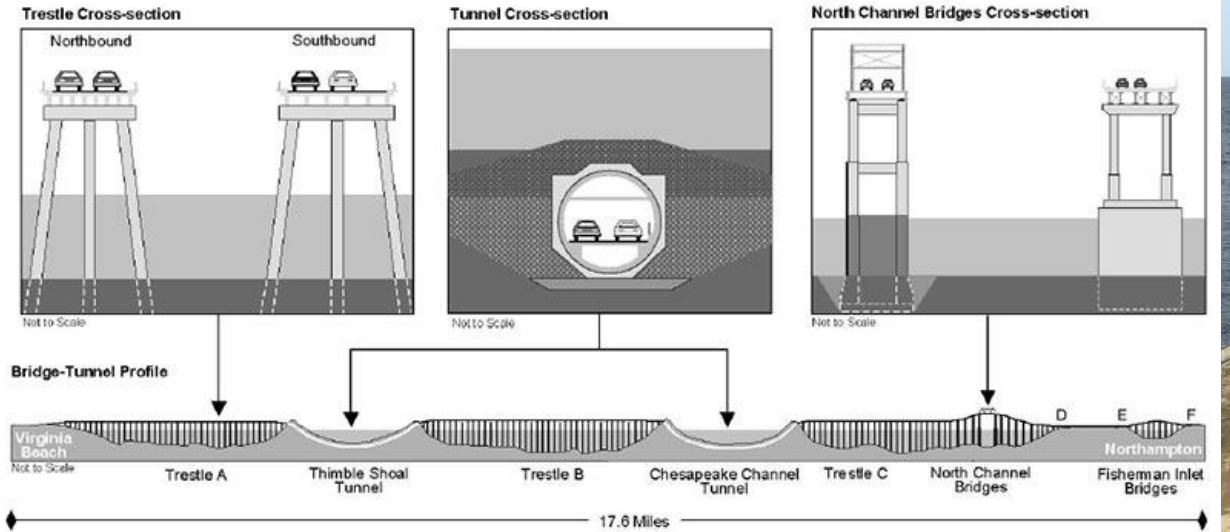
  - Sommerville, 2015, p.22 (emphasis mine)



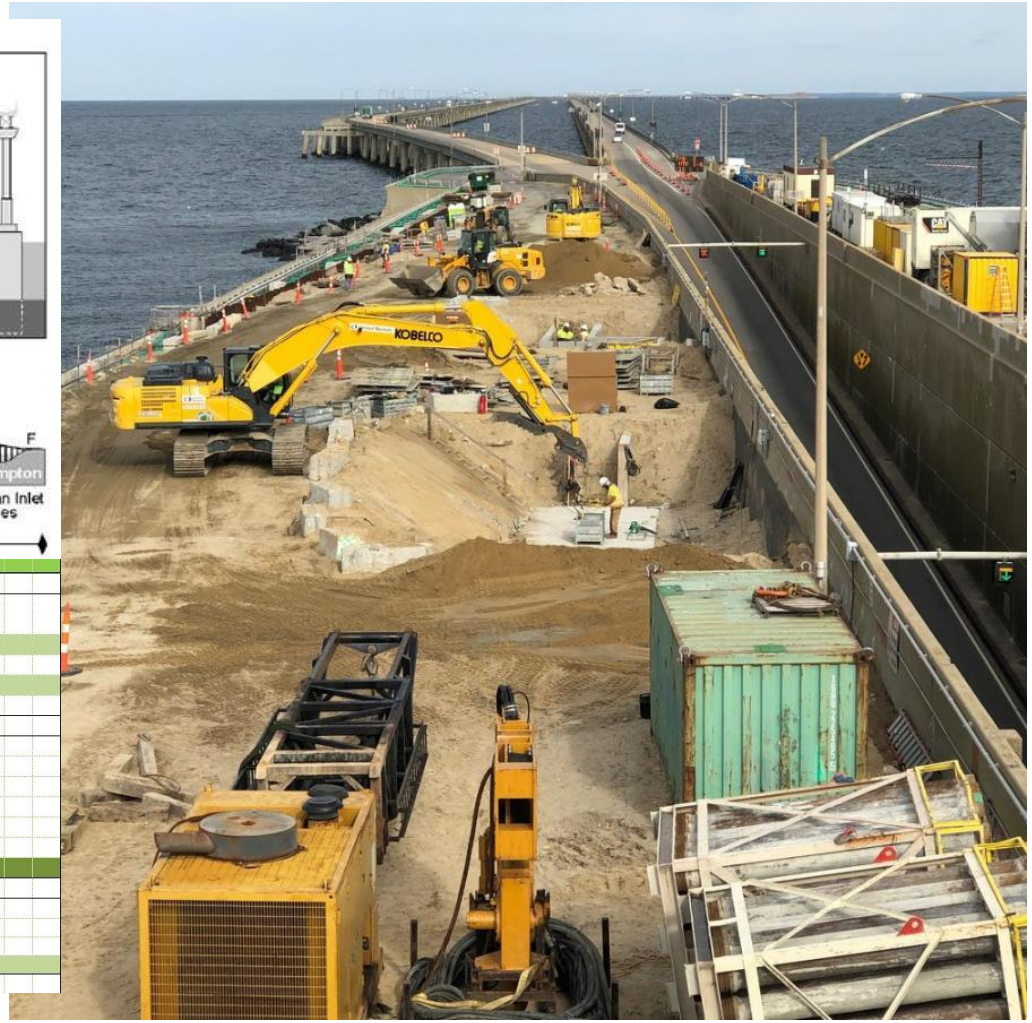
Komplexität



Das, nur für Software!



Project planning & management	
Project setup & targets	
SEO project plan creation	
Onsite support	
Documentation and recommendations	
Project communications	
SEO monthly reporting	
Semantic Analysis	
Finding potential keywords	
Keyword search volume analysis	
Keyword validation for SEO	
Keyword mapping for top level pages	
Site content analysis based on keyword analysis	
Customized Title and Meta Description elements	
Content modifications	
Technical Analysis	
Page template analysis	
Site crawl analysis	
Site quality evaluation	
Monthly Google Webmaster Tools checkup	
Link analysis & building	



Software Engineering

- “Software engineering is an engineering discipline that is concerned with **all aspects of software production** from initial conception to operation and maintenance.”

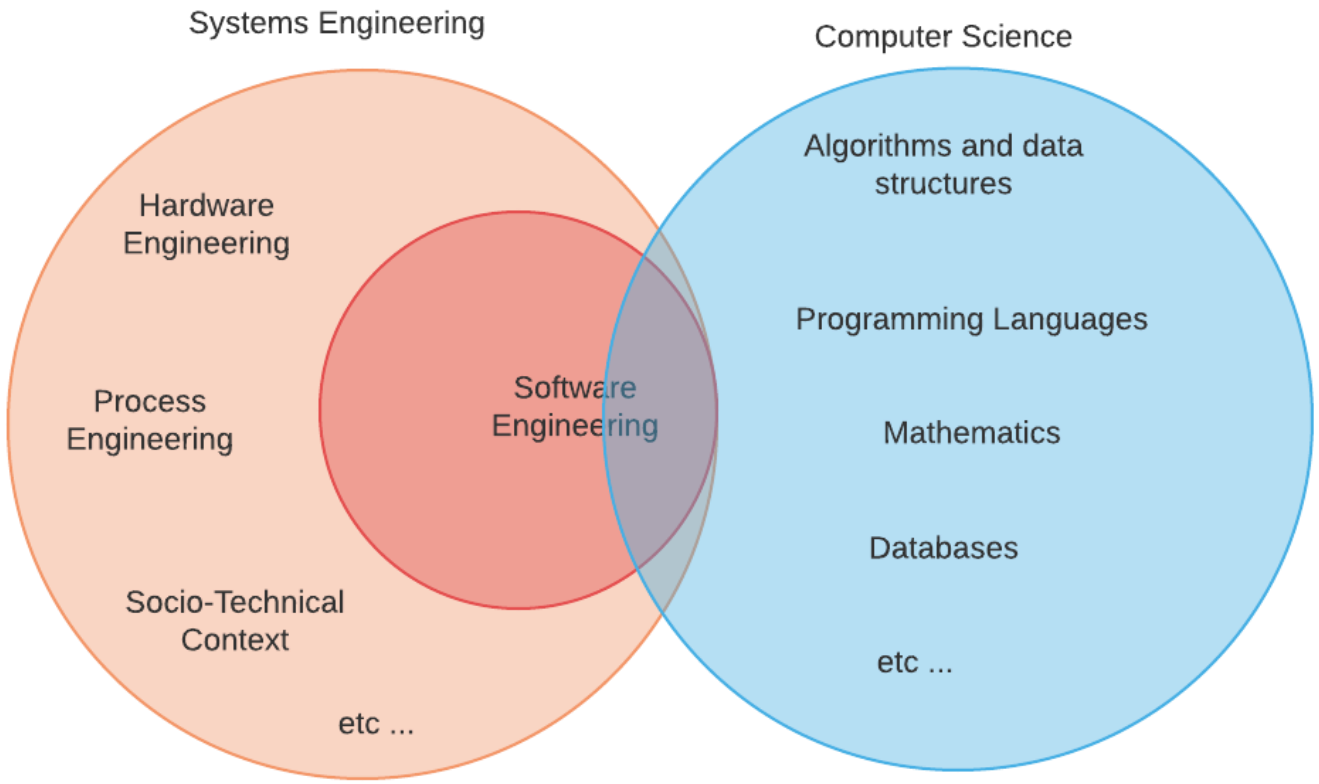
Sommerville, 2015, p.20
Emphasis mine

ALLE Aspekte der Software Produktion

- Specification
- Development
- Validation
- Evolution

mehr dazu gleich…

SE vs. Systems Engineering vs. Computer Science



Guter Programmierer ⇉ Software Engineer

- Entwicklung und Anwendung von…
  - … Tools
  - … Methoden
  - … Theorien

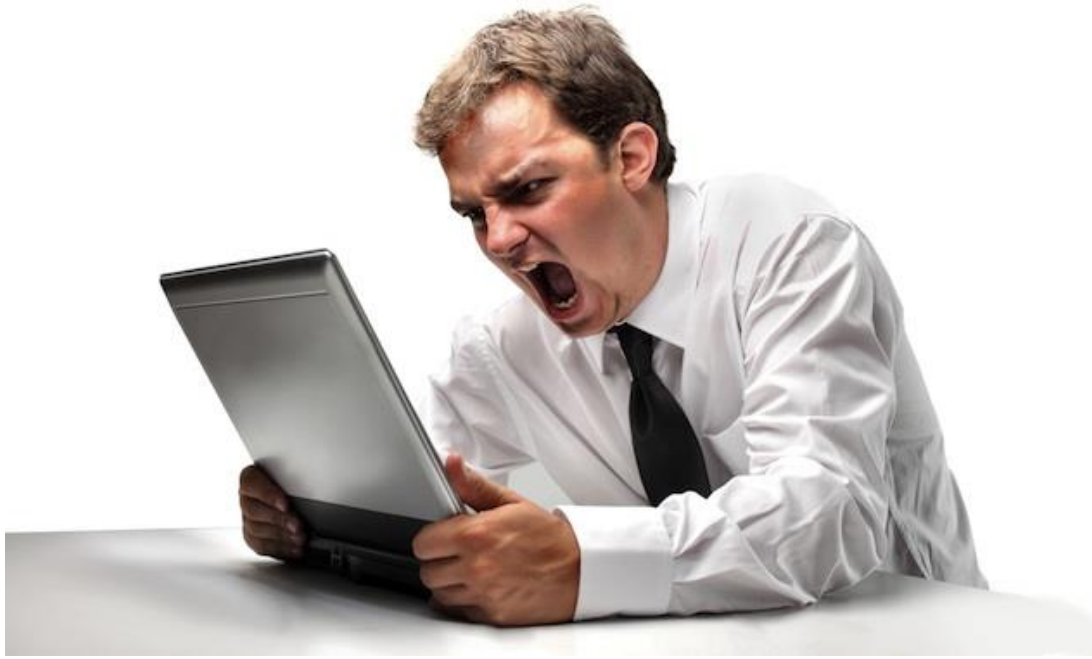
  für die Entwicklung von Software
- Software Projektmanagement

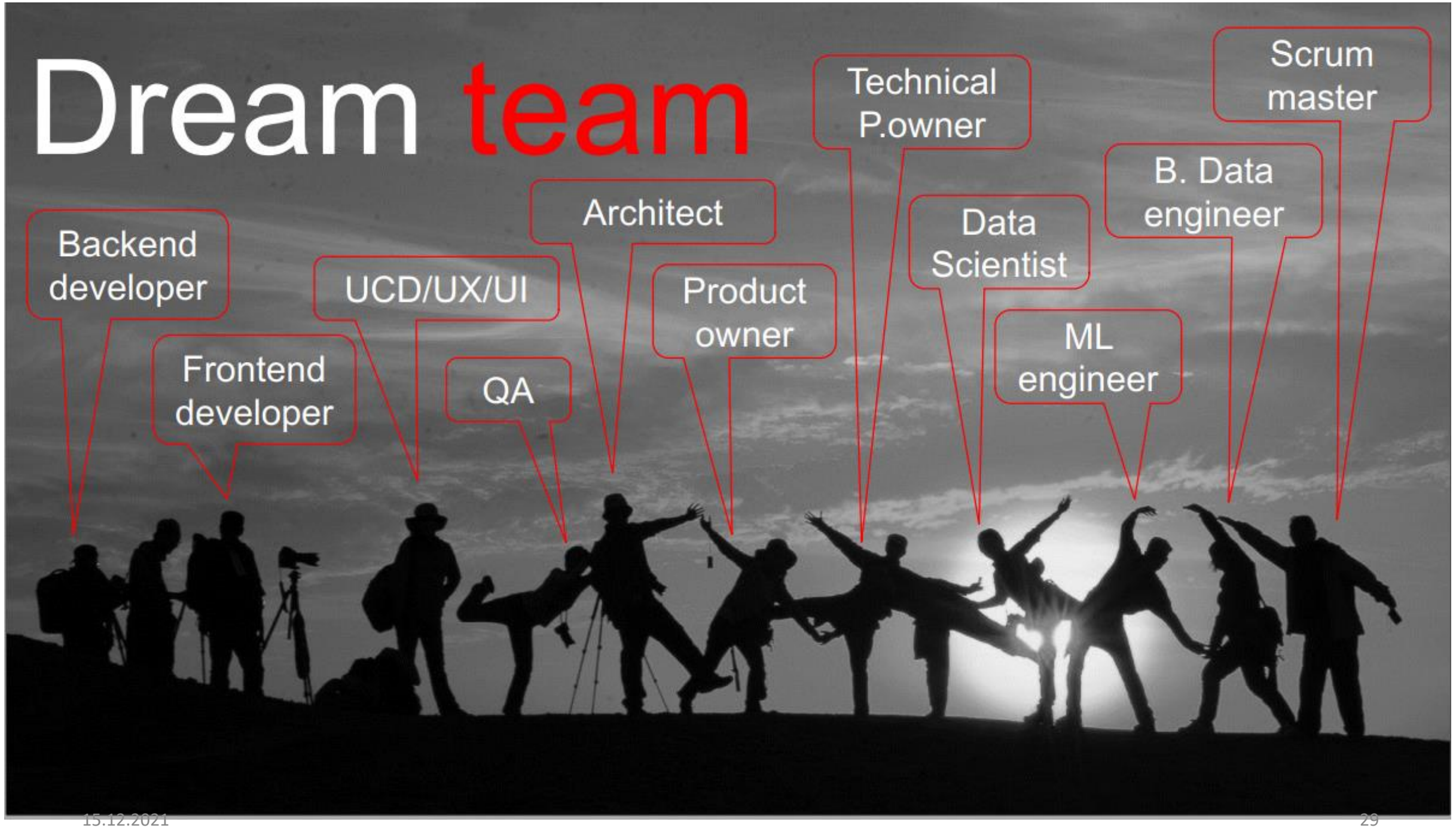
„It works on my machine…“

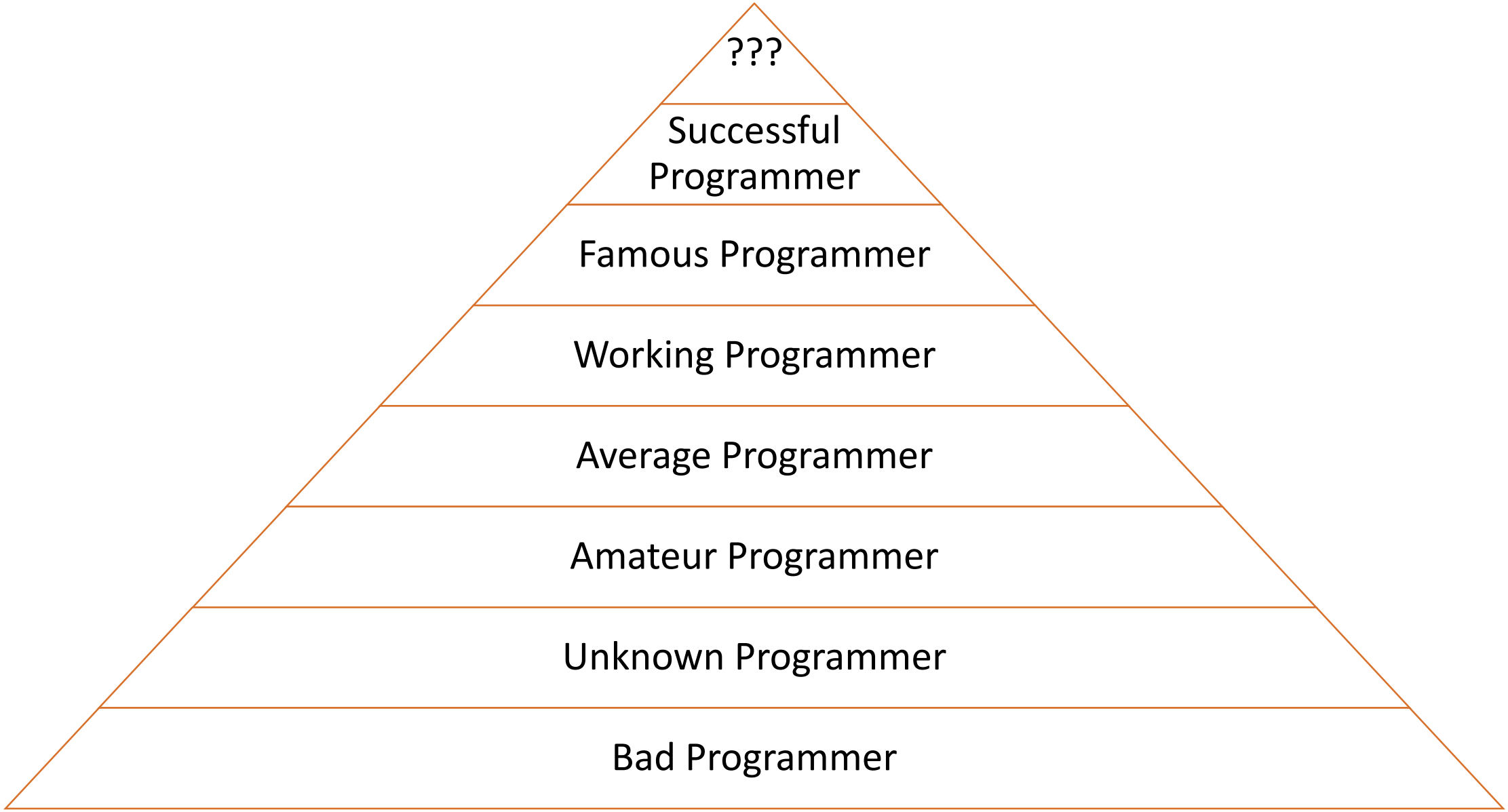
Profis schreiben Software für Andere

Development Team

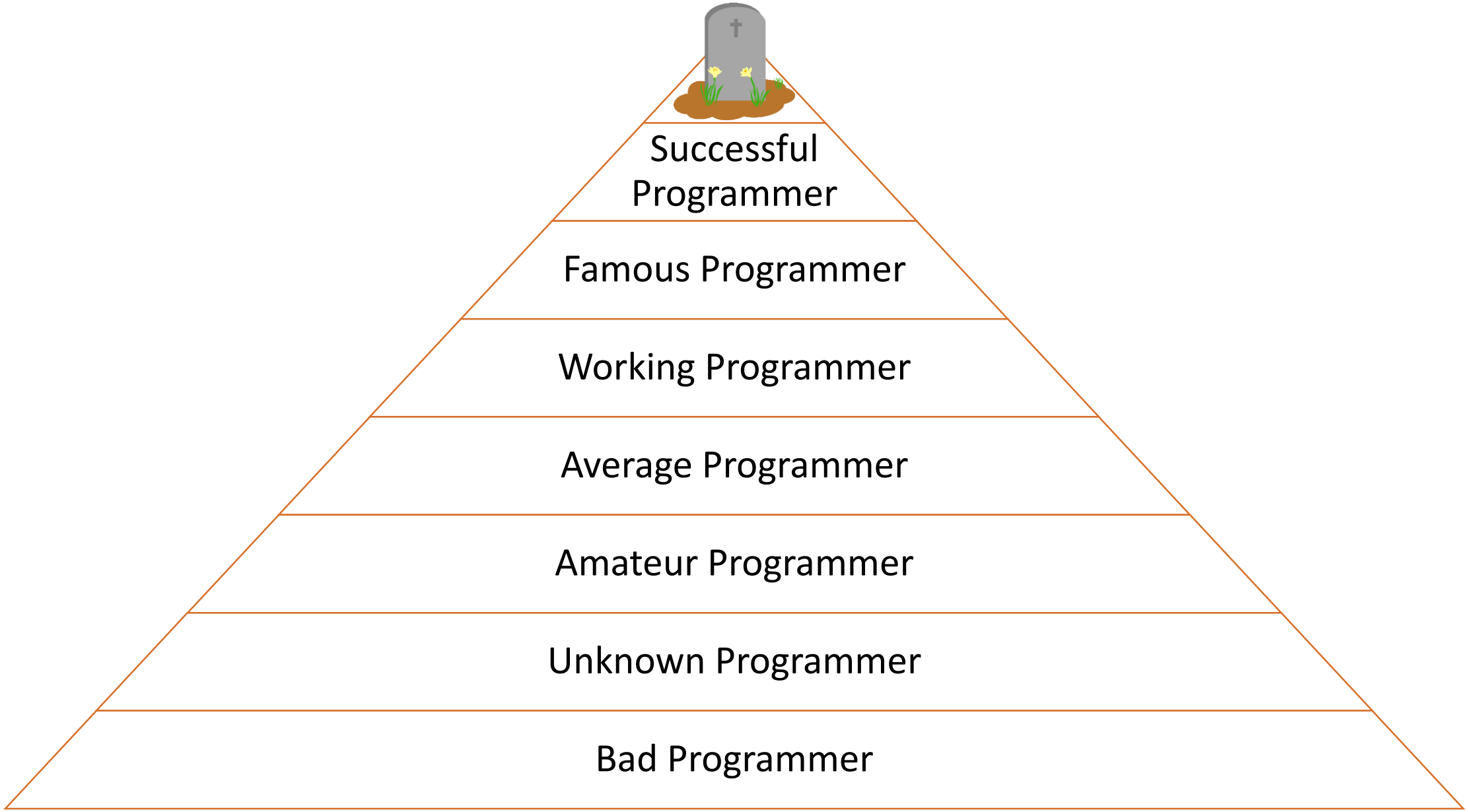
Kunden



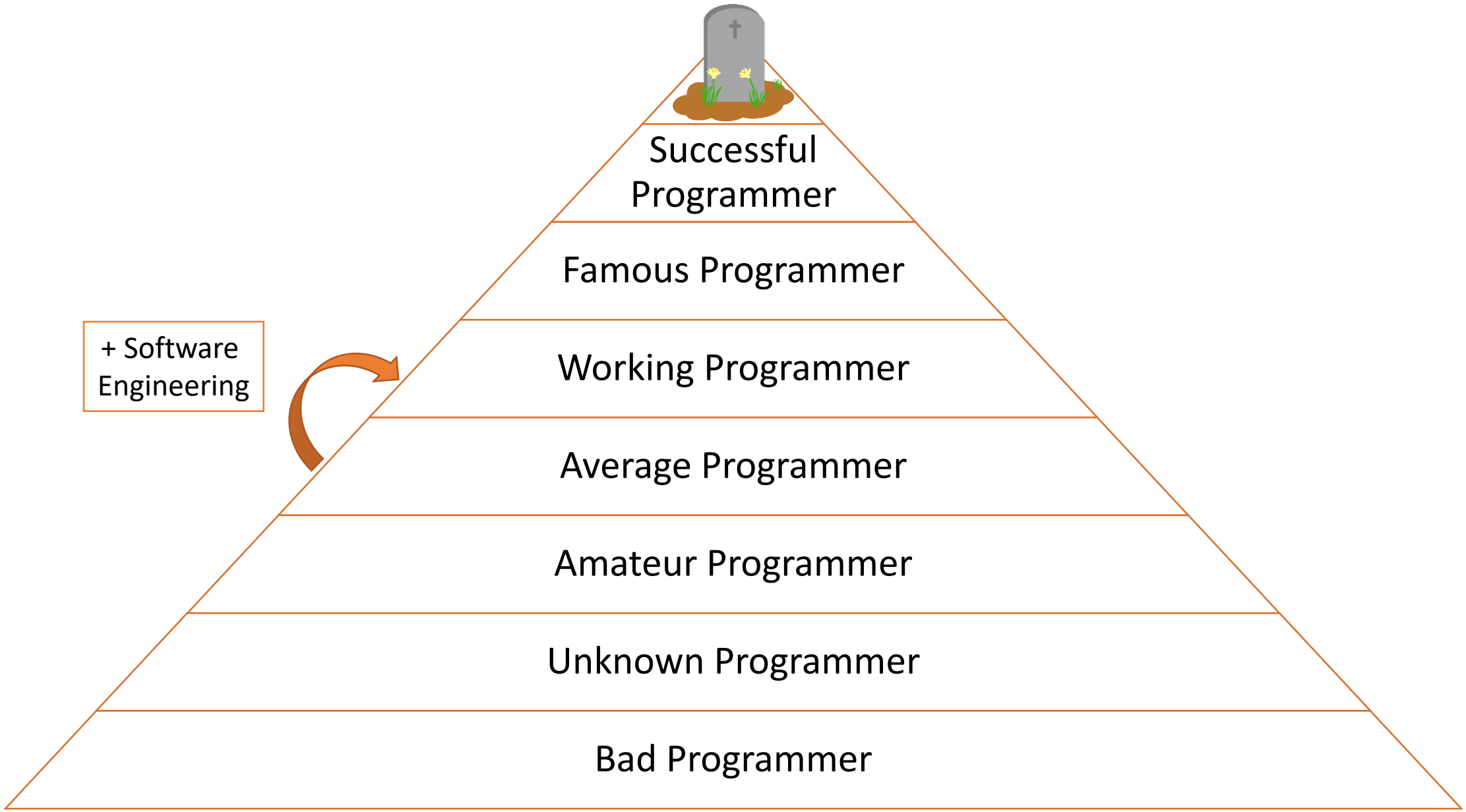

Dream team
Backend developer
Frontend developer
UCD/UX/UI
QA
Architect
Product owner
Technical P.owner
Data Scientist
ML engineer
B. Data engineer
Scrum master

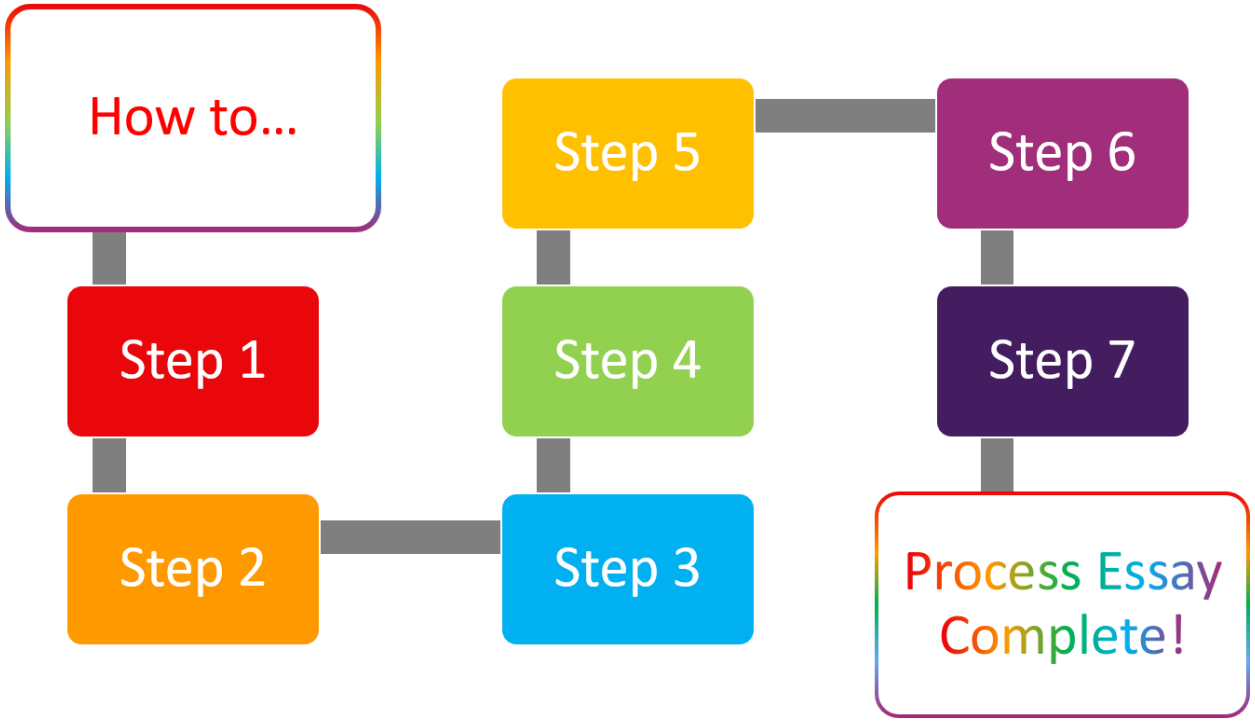


https://blog.codinghorror.com/the-eight-levels-of-programmers/



https://blog.codinghorror.com/the-eight-levels-of-programmers/


Successful Programmer
Famous Programmer
Working Programmer
Average Programmer
Amateur Programmer
Unknown Programmer
Bad Programmer
+ Software Engineering



Software Prozesse

Wie?

Essentielle Aktivitäten

1. Requirements Engineering
   Funktionen und Rahmenbedingungen müssen definiert werden

2. Software Design und Entwicklung
   Die Software muss entsprechend den Spezifikationen entworfen & gebaut werden

3. Software Validierung
   Es muss geprüft werden, ob das Produkt alle Spezifikationen erfüllt

4. Software Evolution
   Die Software muss verändert warden um sich neuen Anforderungen anzupassen

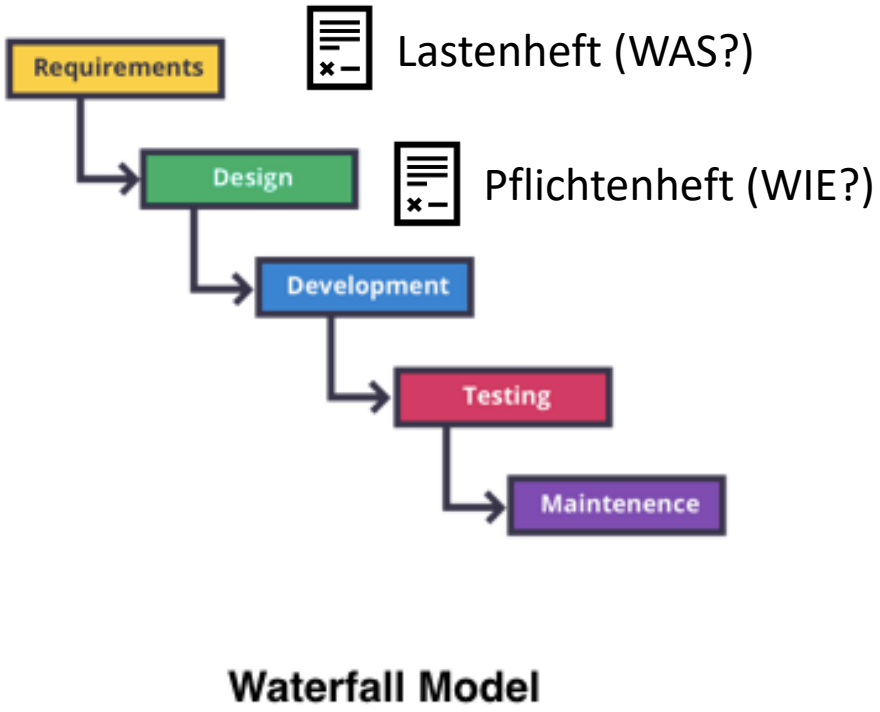
Wann werden diese Aktivitäten ausgeführt?

- „Prozess-Paradigma“ -> Allgemeine Anleitung für SE

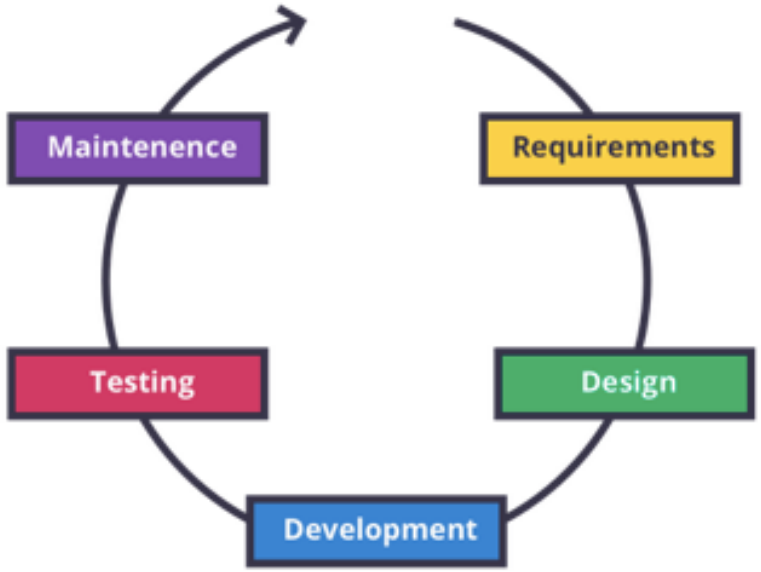
- Es gibt viele Paradigmen. Wir behandeln:

1. Wasserfallmodell
2. Inkrementelle Entwicklung
3. Integration und Konfiguration

A Tale of Two Cities



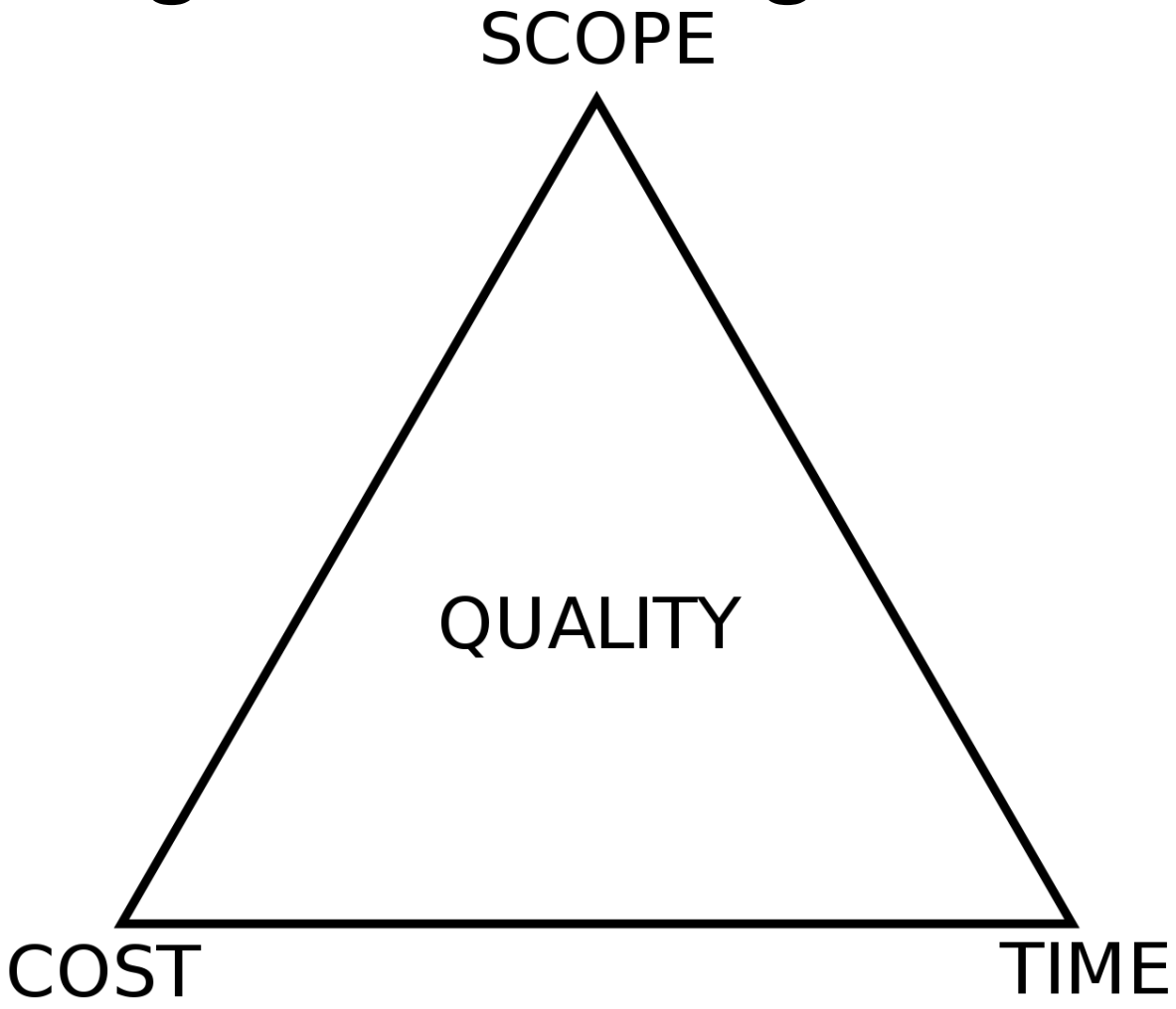
Waterfall Model



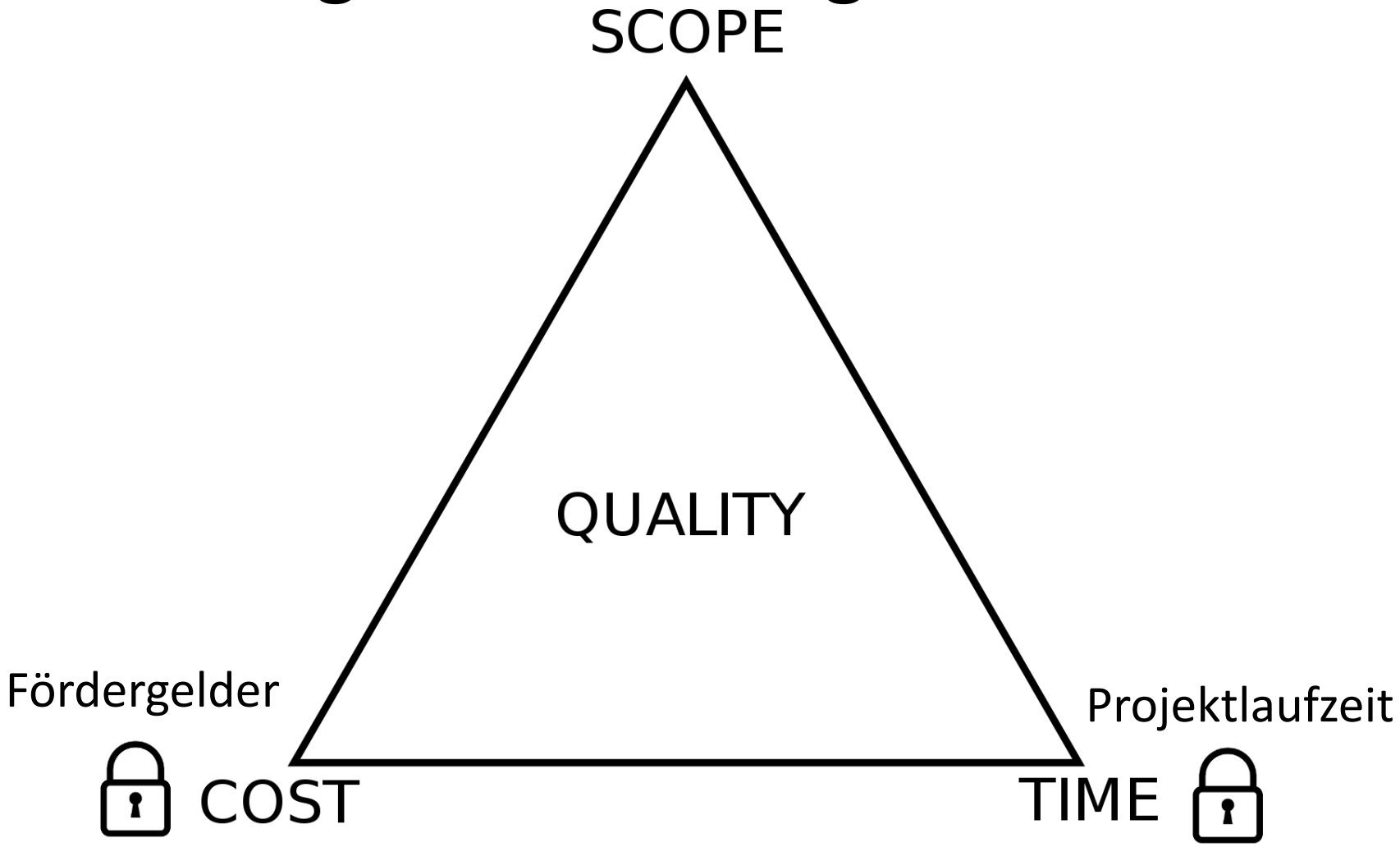
Agile Cycle

https://workingmouse.com.au/innovation/waterfall-vs-agile-choosing-the-right-fit

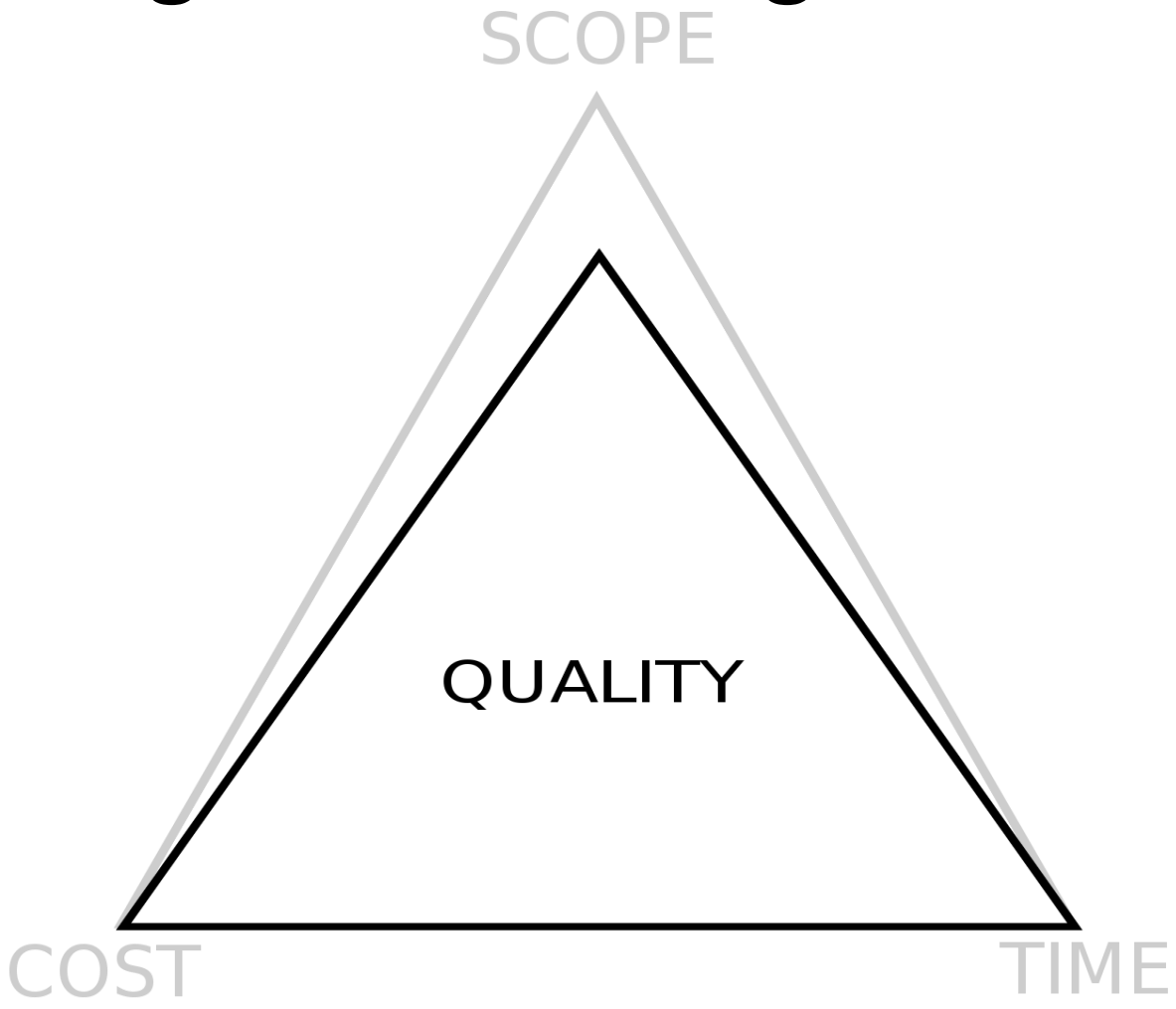
Project Management Triangle

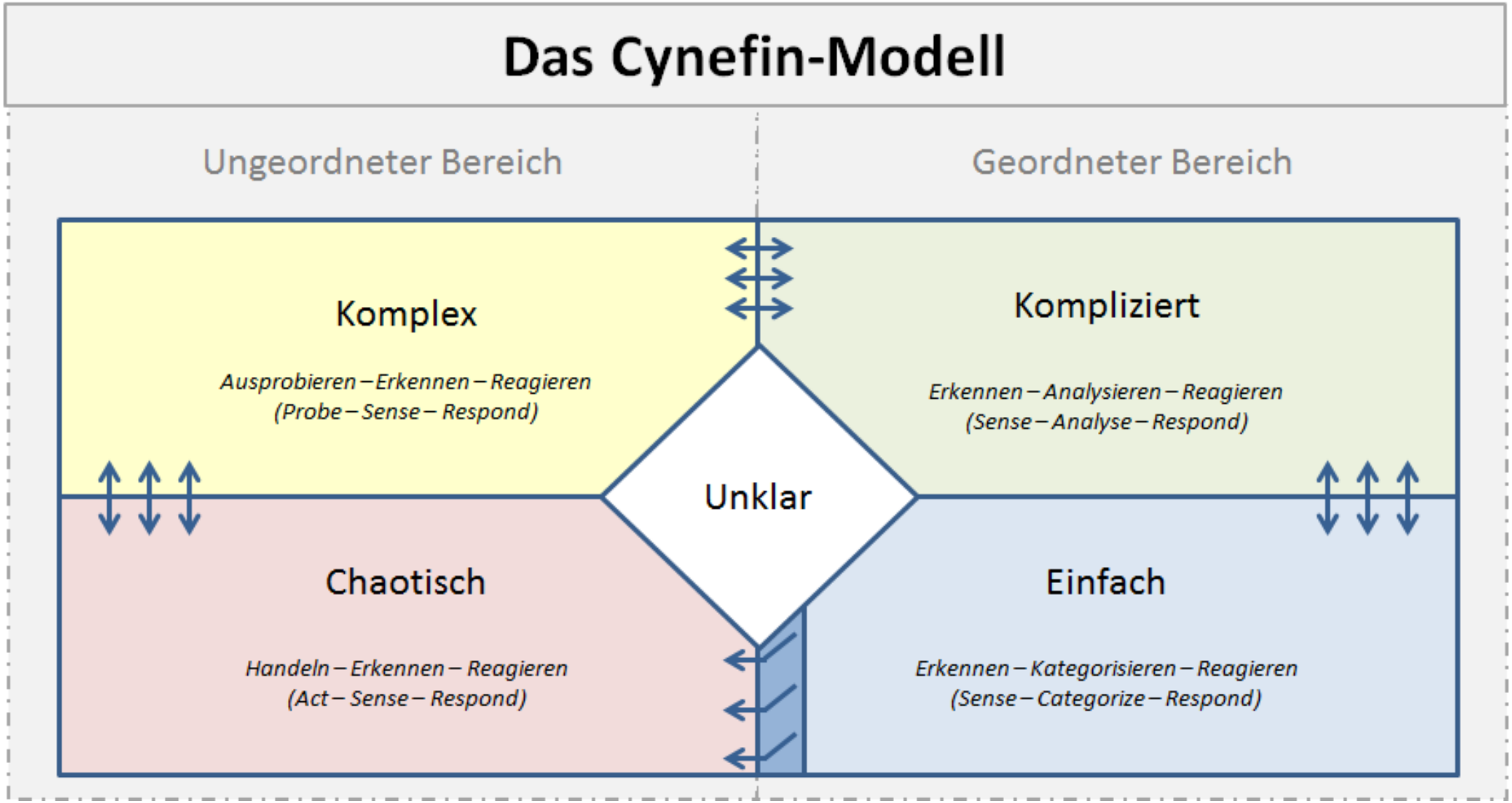


Project Management Triangle



Project Management Triangle





https://images.app.goo.gl/e5iTSFjZVz5svBsT7

Übung zu Cynefin

1. Reihum: Jeder sagt einen Buchstaben des Alphabets

Übung zu Cynefin

1. Reihum: Jeder sagt einen Buchstaben des Alphabets

2. 26 Wörter lange Geschichte, Anfangsbuchstaben A-Z

   Thema: Wie ich ans WIFI Salzburg angereist bin
   JEDER NUR 1 WORT!

Übung zu Cynefin

1. Reihum: Jeder sagt einen Buchstaben des Alphabets

2. 26 Wörter lange Geschichte, Anfangsbuchstaben A-Z

   Thema: Wie ich ans WIFI Salzburg angereist bin
   JEDER NUR 1 WORT!

3. 3 Minuten

   Thema: Das letzte Konzert / Feier / Party….
   JEDER NUR 1 WORT!

Übung zu Cynefin

1. Reihum: Jeder sagt einen Buchstaben des Alphabets

Feste Regeln, kein Raum für Kreativität, Best Practice, Einfach zu Googlen

1. 26 Wörter lange Geschichte, Anfangsbuchstaben A-Z

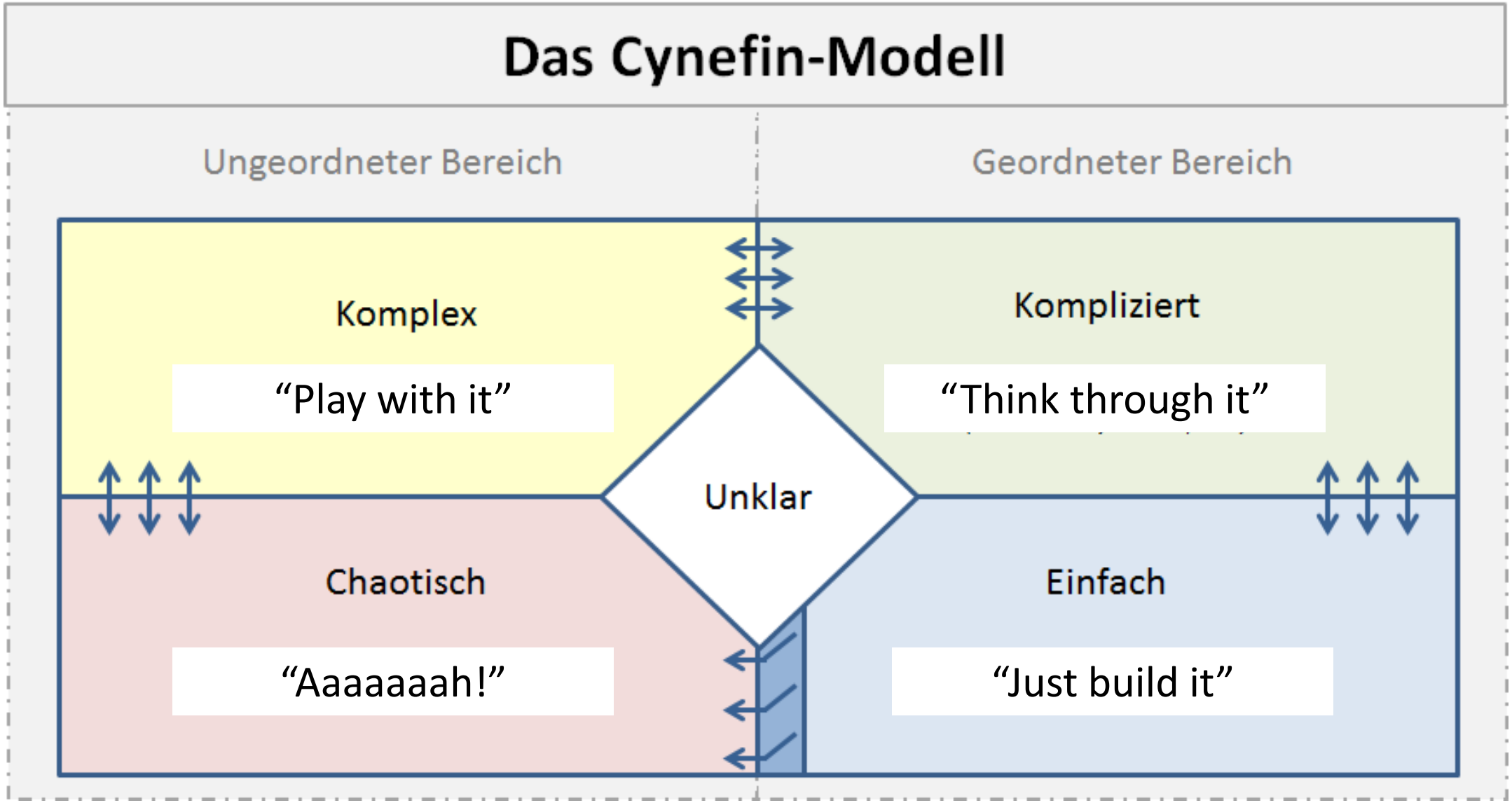
Gelockerte Regeln, Versch. Sichtweisen, Fehler fallen auf, Endergebnis nicht fix

1. 3 Minuten

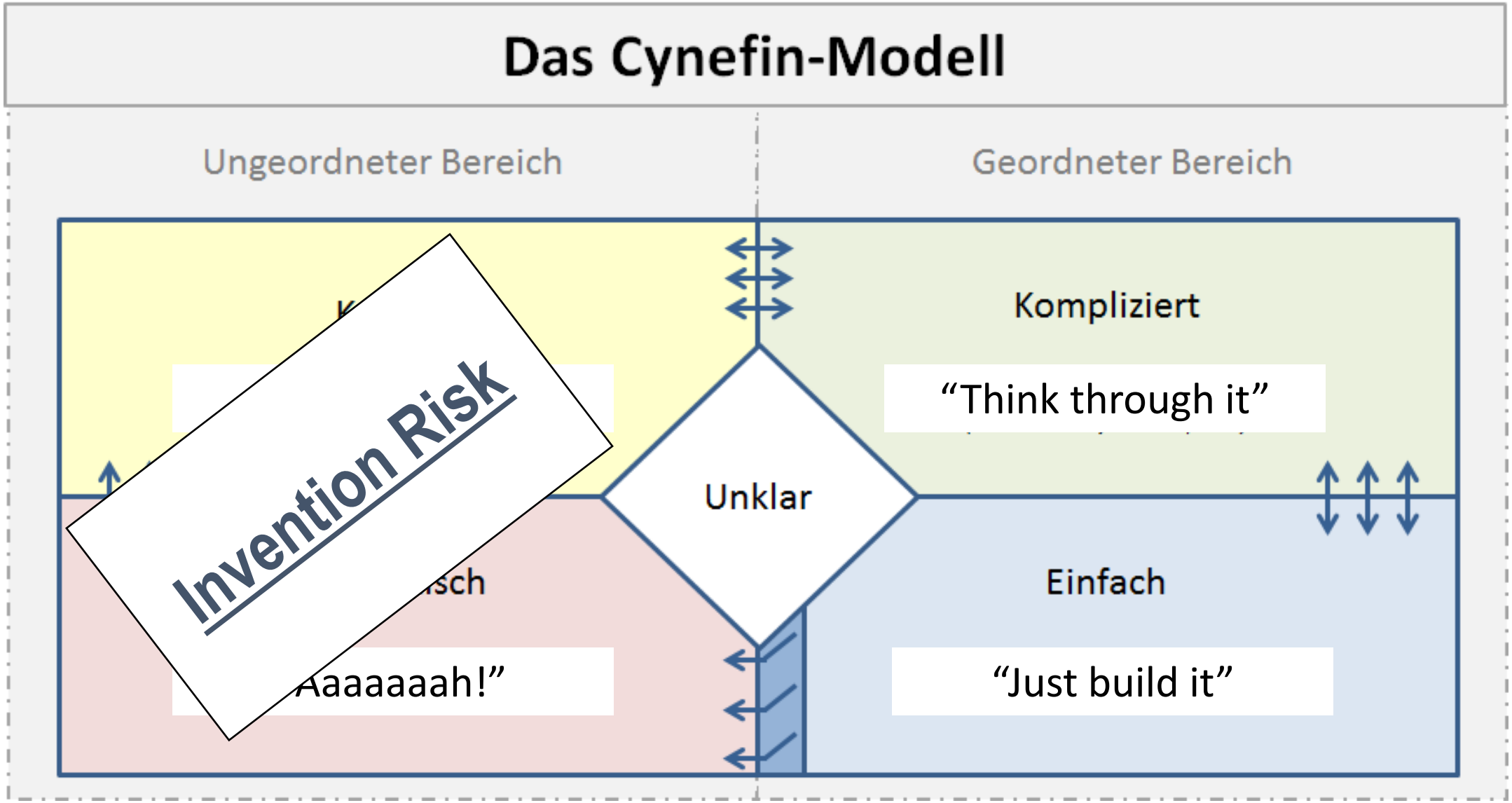
Nur eine Regel, Raum für Kreativität, Endergebnis völlig unbekannt

Chaos

- https://www.youtube.com/watch?v=yQeQwwXXa7A

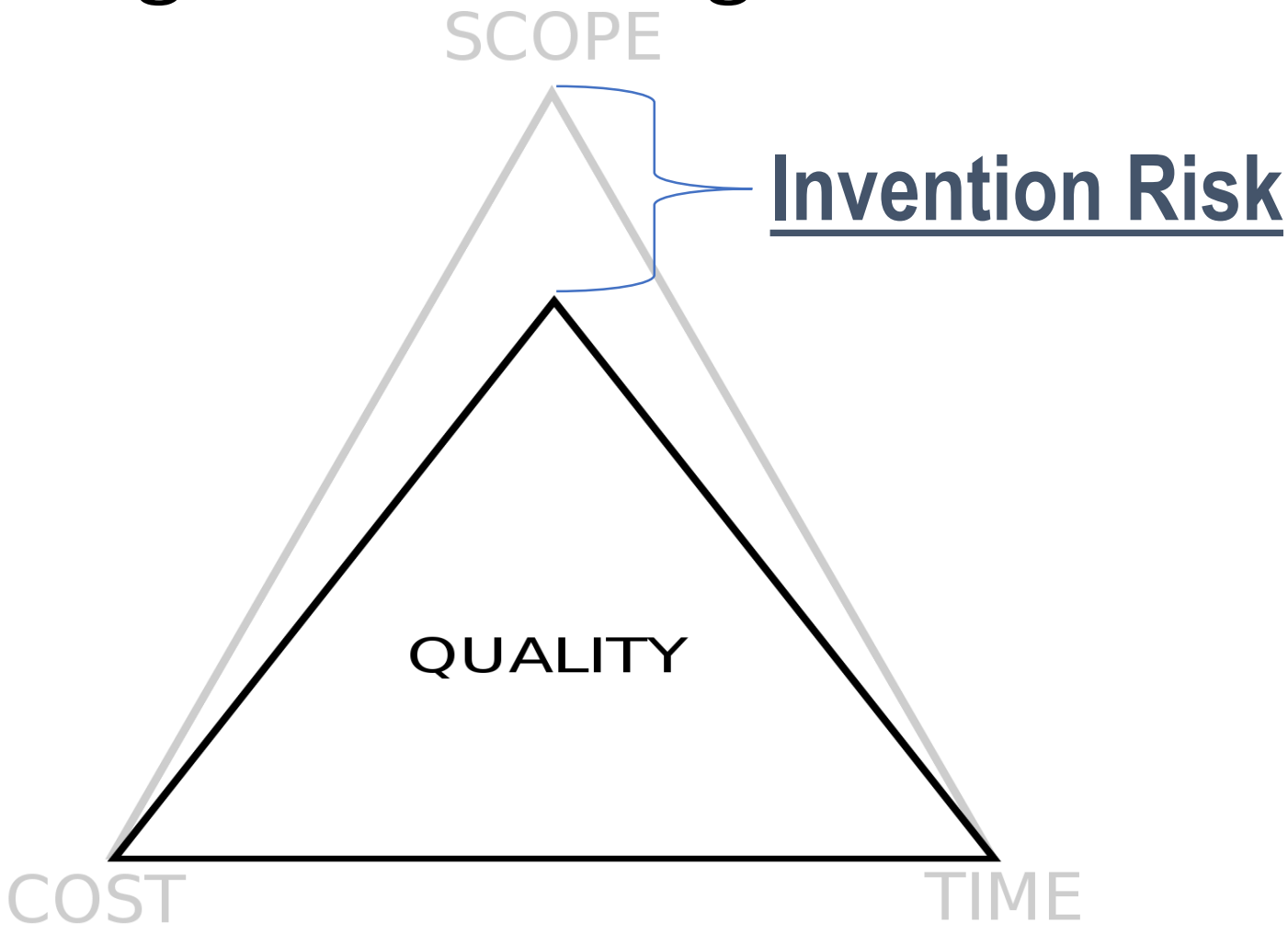


https://images.app.goo.gl/e5iTSFjZVz5svBsT7



https://images.app.goo.gl/e5iTSFjZVz5svBsT7

Project Management Triangle



Das Cynefin-Modell

Ungeordneter Bereich

Geordneter Bereich

Komplex

“Play with it”

Kompliziert

“Think through it”

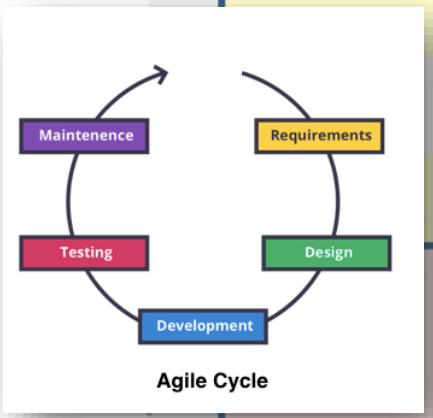
Unklar

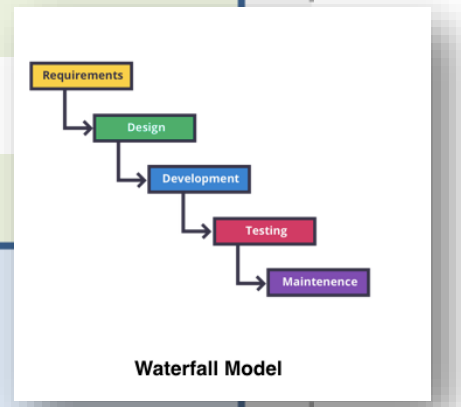
Chaotisch

“Aaaaaaah!”

Einfach

“Just build it”





https://images.app.goo.gl/e5iTSFjZVz5svBsT7

Requirements Engineering

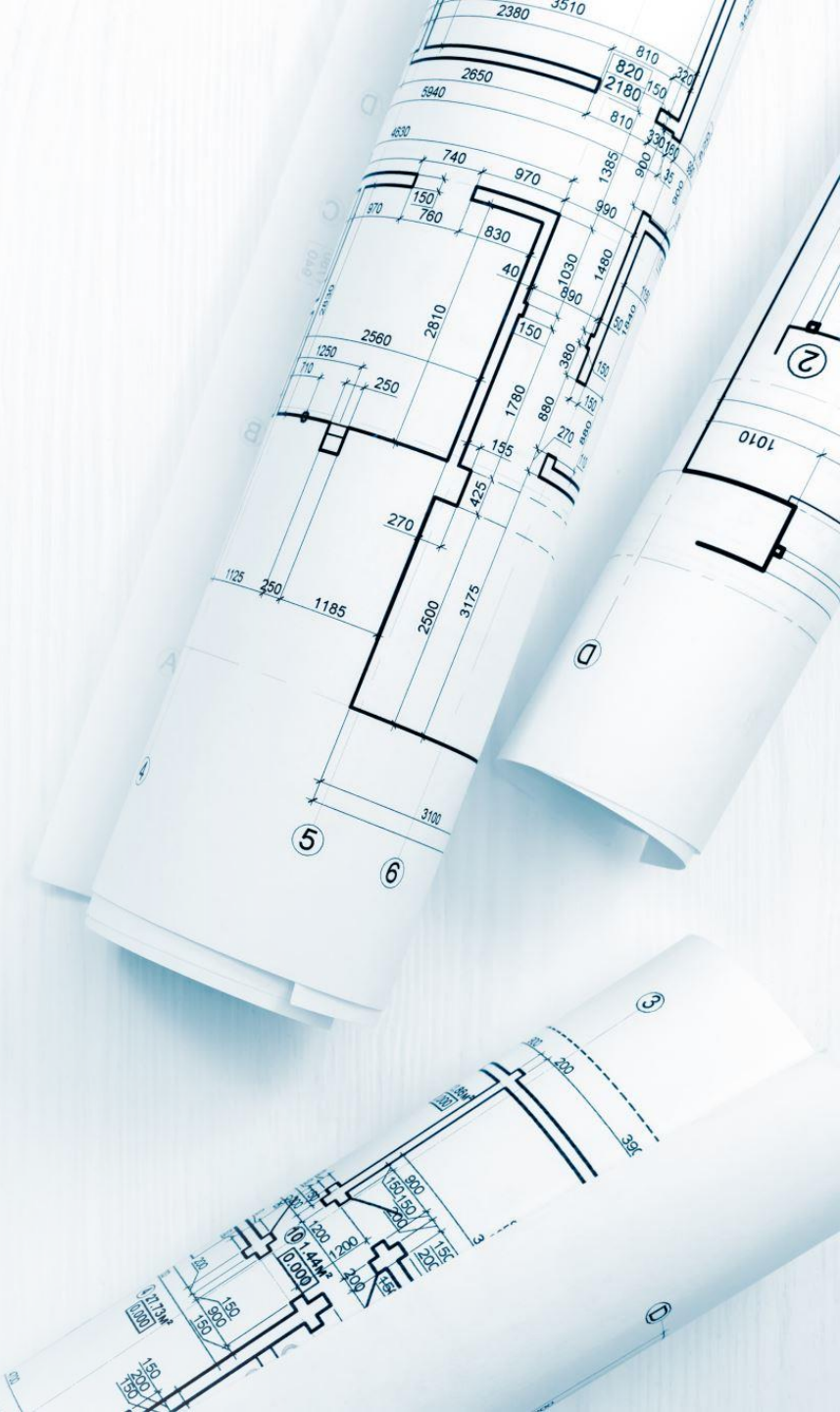


Table 1.1 Reasons for project failure (Standish Group (1995))

**	Incomplete requirements	13.1%
**	Lack of user involvement	12.4%
	Lack of resources	10.6%
*	Unrealistic expectations	9.9%
	Lack of executive support	9.3%
**	Changing requirements/specifications	8.7%
	Lack of planning	8.1%
**	Didn't need it any longer	7.5%

Dick, Jeremy, Elizabeth Hull, and Ken Jackson. 2017. 53 *Requirements Engineering*. Cham: Springer International Publishing.

Table 1.2 Project success factors (Standish Group (1995))

**	User involvement	15.9%
	Management support	13.9%
**	Clear statement of requirements	13.0%
	Proper planning	9.6%
*	Realistic expectations	8.2%
	Smaller milestones	7.7%
	Competent staff	7.2%
*	Ownership	5.3%

Dick, Jeremy, Elizabeth Hull, and Ken Jackson. 2017. 53 *Requirements Engineering*. Cham: Springer International Publishing.

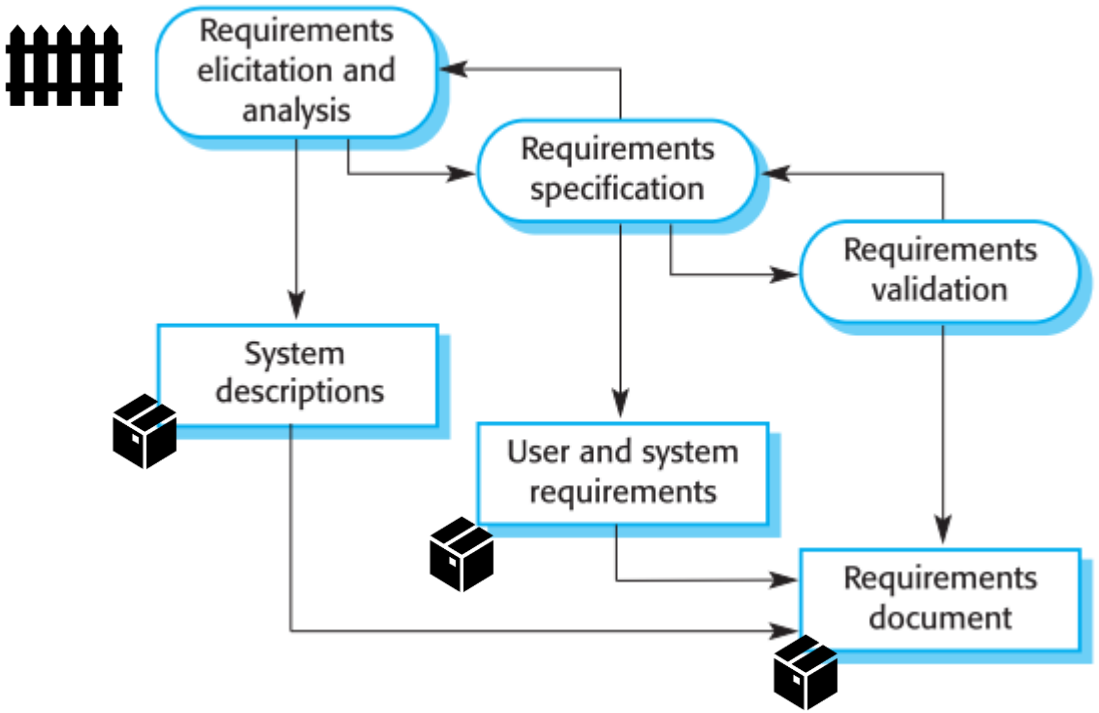
Table 1.3 Project success factors (Standish Group (2015).)

*	Executive management support	20%
**	User involvement	15%
	Optimization	15%
	Skilled resources	13%
	Project management expertise	12%
*	Agile process	10%
**	Clear business objectives	6%
*	Emotional maturity	5%
	Execution	3%
	Tools and infrastructure	1%

Dick, Jeremy, Elizabeth Hull, and Ken Jackson. 2017. 53 *Requirements Engineering*. Cham: Springer International Publishing.

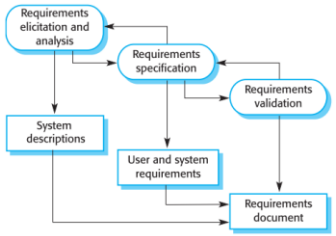
Requirements Engineering

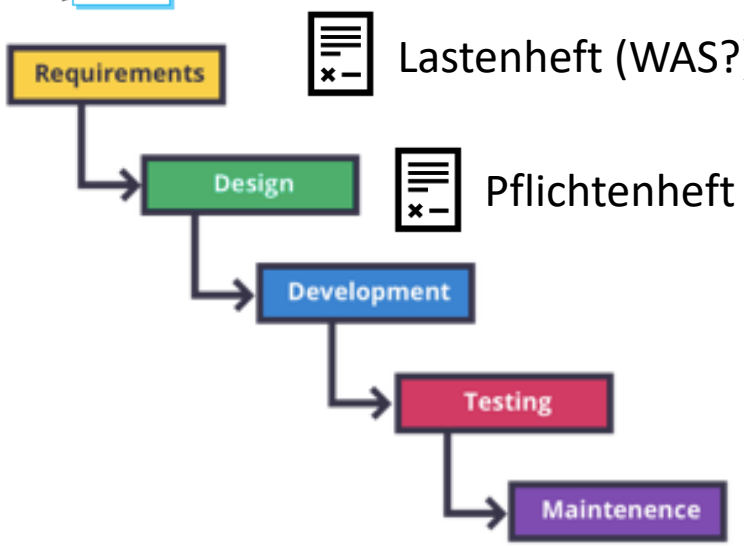
Marktanalyse
Stand der Technik (SOTA)
Machbarkeitsstudie



Alle Stakeholder

Konsens unter Stakeholdern

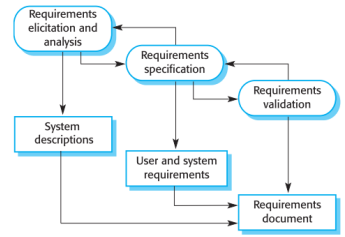


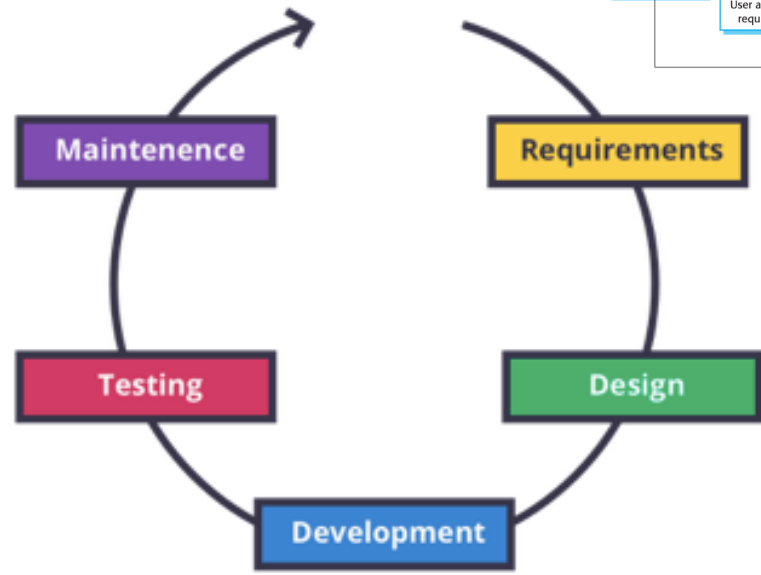


Lastenheft (WAS?)

Pflichtenheft (WIE?)

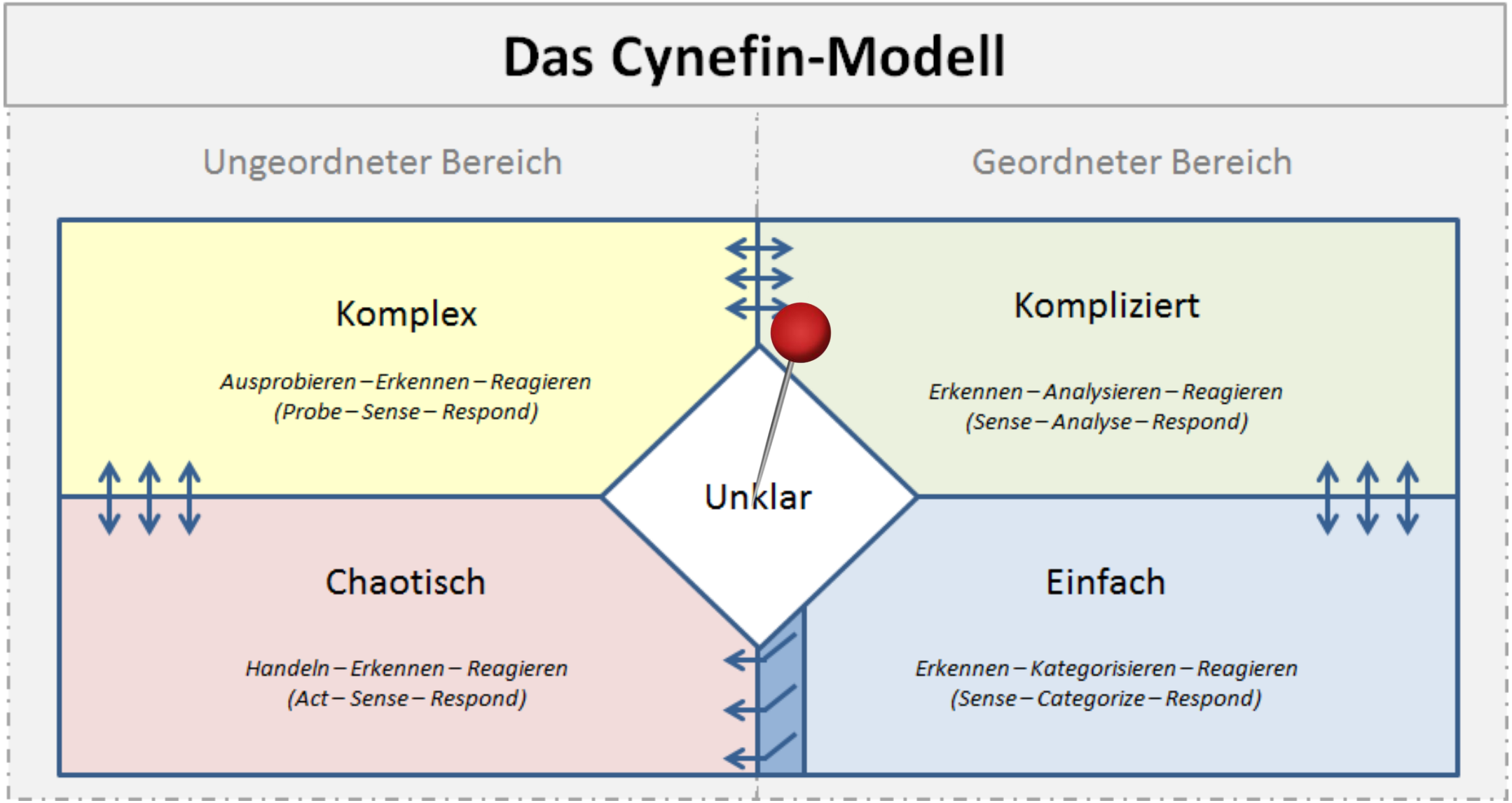
Waterfall Model



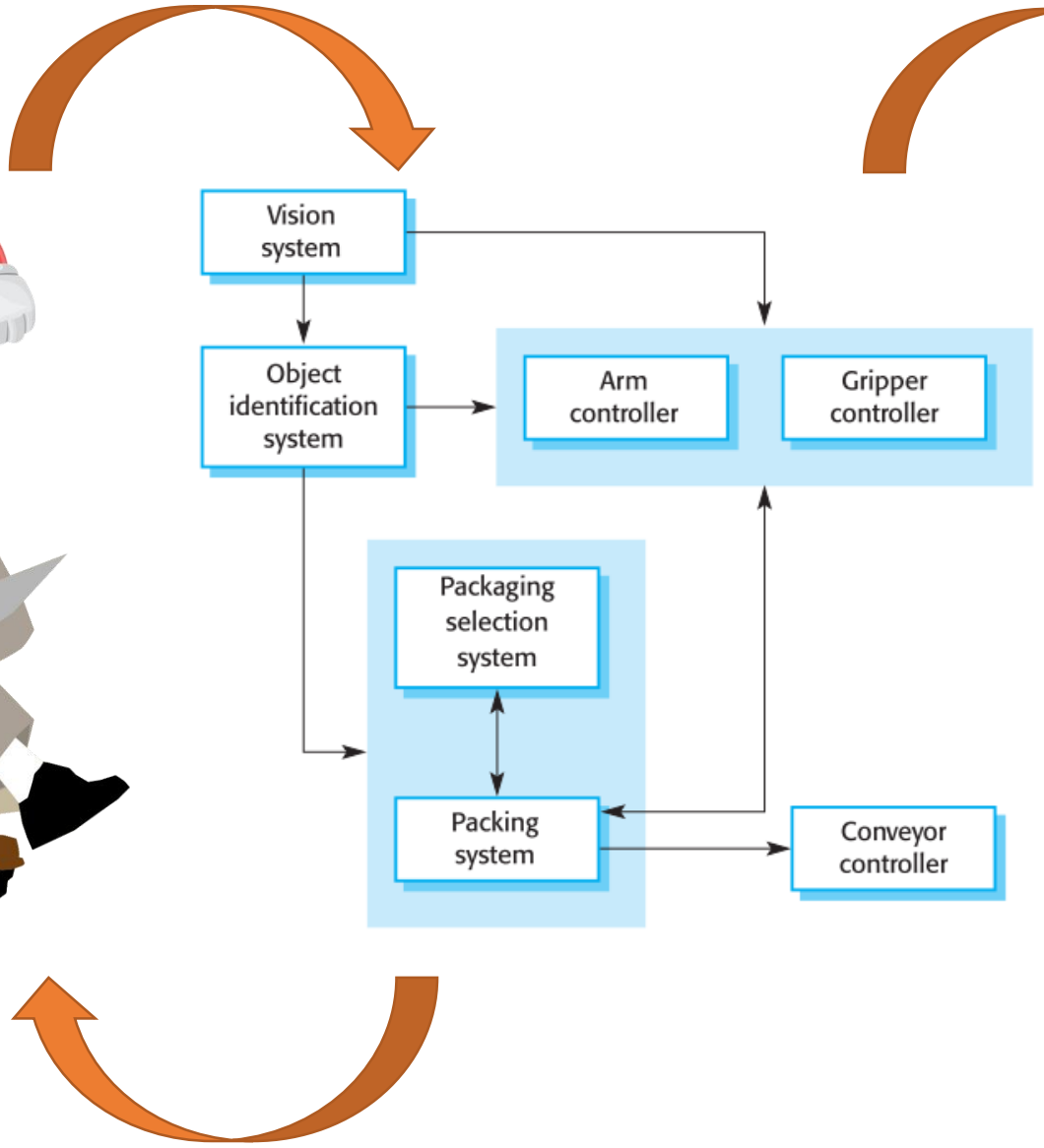


Agile Cycle

https://workingmouse.com.au/innovation/waterfall-vs-agile-choosing-the-right-fit



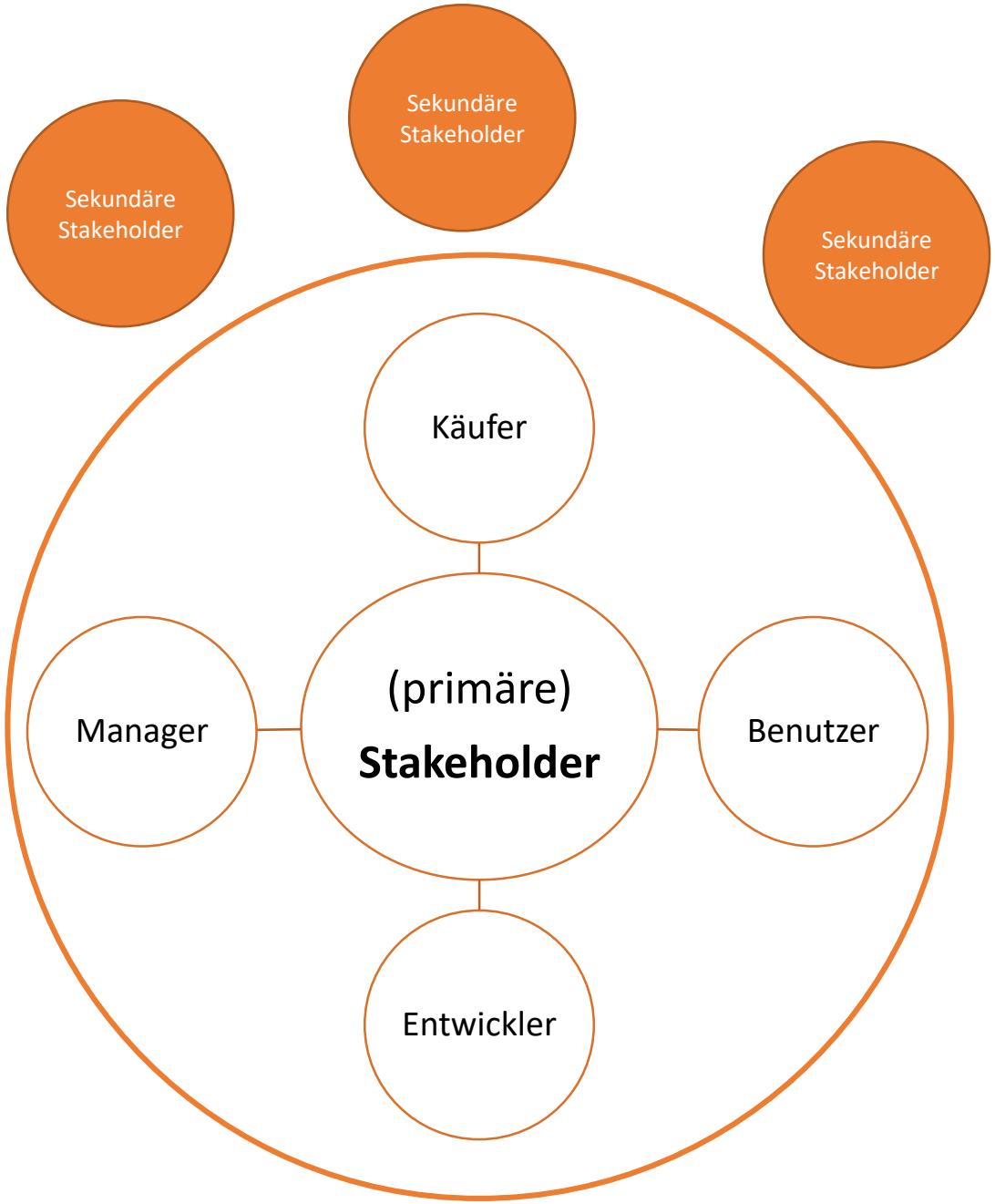
https://images.app.goo.gl/e5iTSFjZVz5svBsT7


Vision system
Object identification system
Arm controller
Gripper controller
Packaging selection system
Packing system
Conveyor controller

Stakeholder

- Stake ~ Interesse am System

- Individuum, Gruppe, Organisation, Partei
- Direktes Interesse
  - Primäre Stakeholder
- Indirektes Interesse
  - Sekundäre Stakeholder



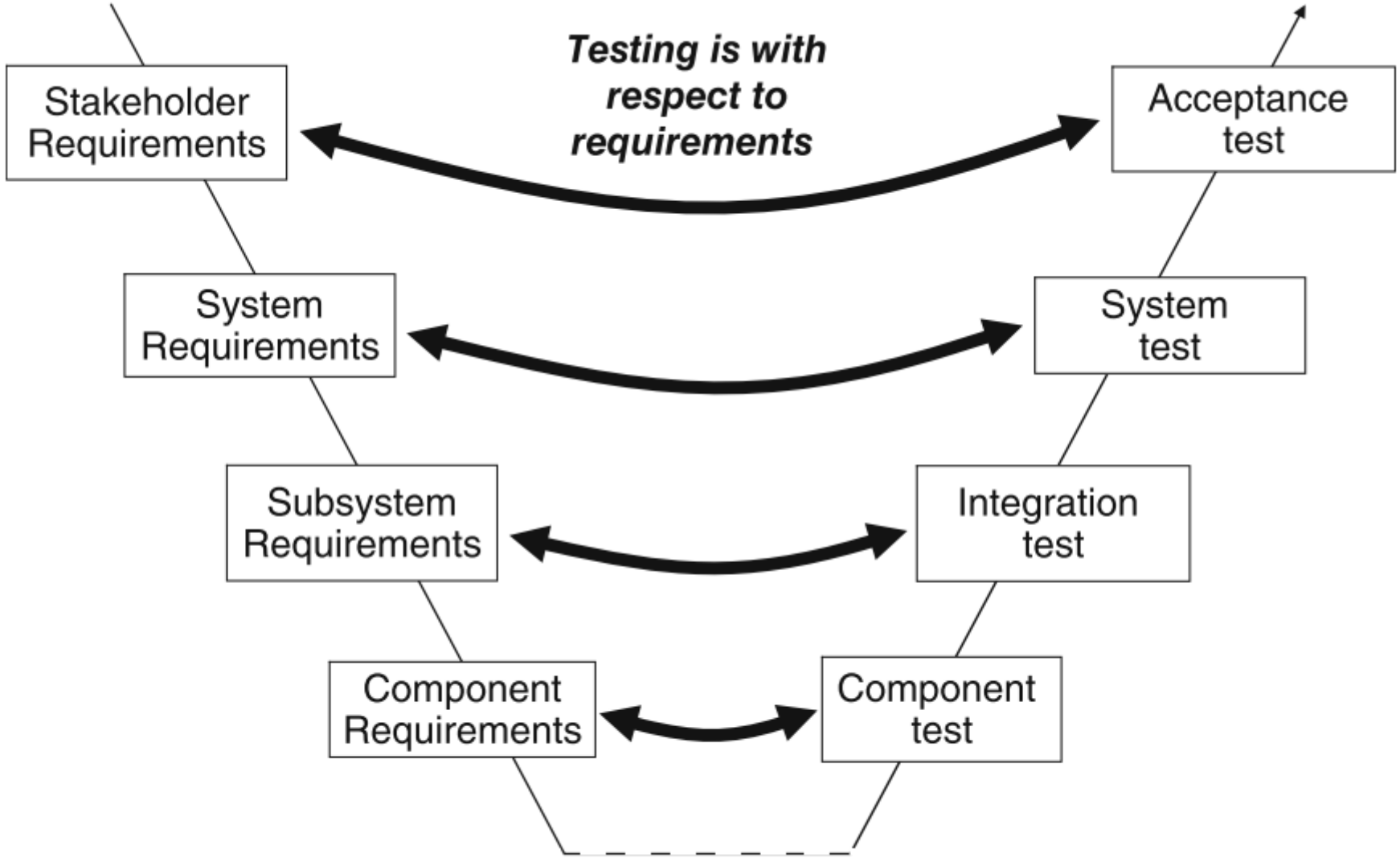


Fig. 1.2 Requirements in the V-Model

Dick, Jeremy, Elizabeth Hull, and Ken Jackson. 2017. 53 *Requirements Engineering*. Cham: Springer International Publishing.

Wie komme ich an Wissen? (Requirements Elicitation)

- Interviews
  - Geschlossen – vorgefertigter Fragebogen
  - Offen – Keine Agenda
  - Mix – fange geschlossen an, werde offener und verfolge Themenstränge
- Workshops
- Ethnographie
  - Software wird in sozio-technologischen Kontexten verwendet
  - => Beobachter nimmt am Alltag teil und sammelt so Requirements

Modell unserer Morgenroutinen!

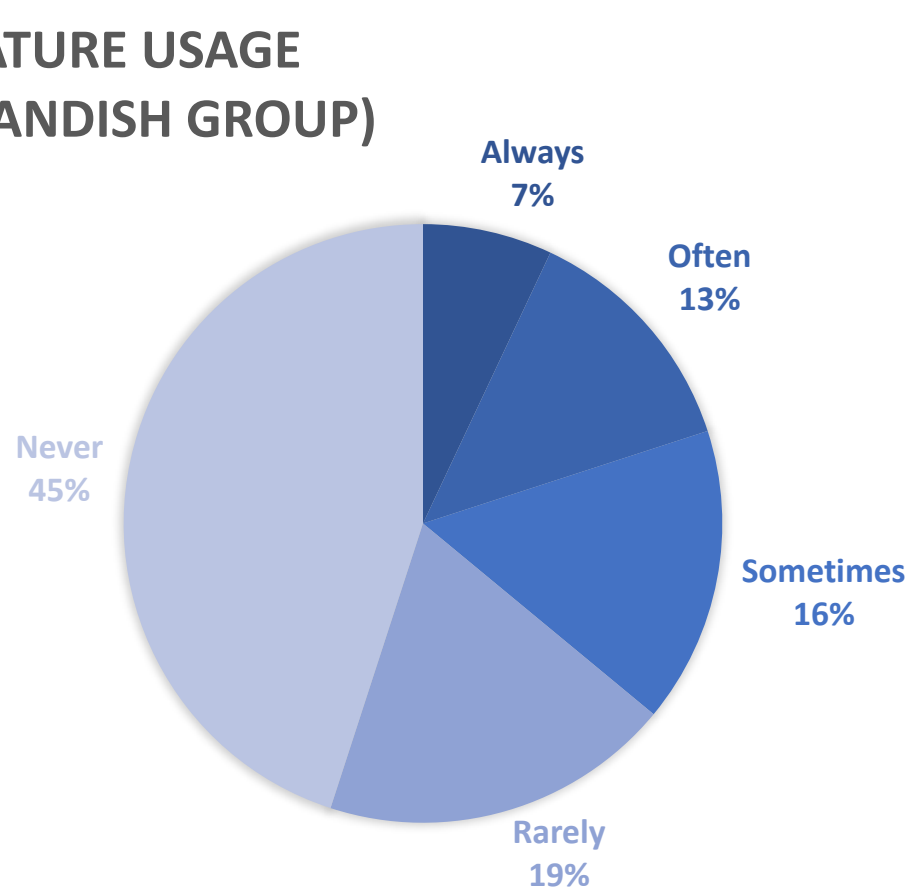
- Übung auf Miro Board

Modell unserer Morgenroutinen!

- Übung auf Miro Board

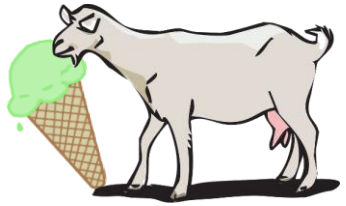
- Jetzt: Entfernen Sie alle Schritte, die nicht unbedingt notwendig sind!

„Minimal Viable Product (MVP)“



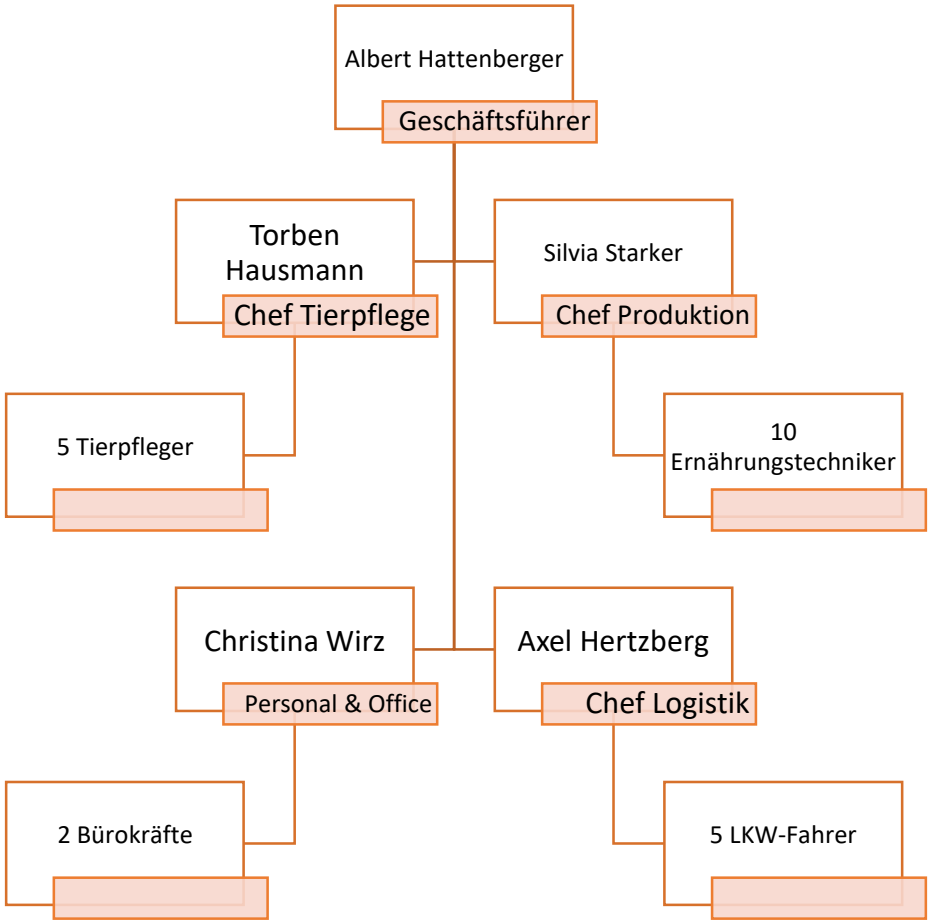
- Nur 1/5 der Features werden regelmäßig genutzt
- 2/3 der Features werden selten bis gar nie benutzt
- => Finde dieses „magische 1/5“!

Fallbeispiel

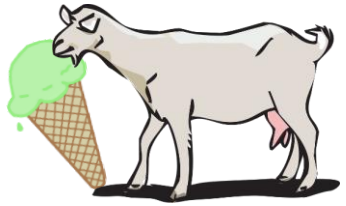


- Firma **Eisziegartig** aus Bad Vigaun stellt Eis aus Ziegenmilch her
- 300 Ziegen
- 5l, 1l und 500ml, 125ml Größen

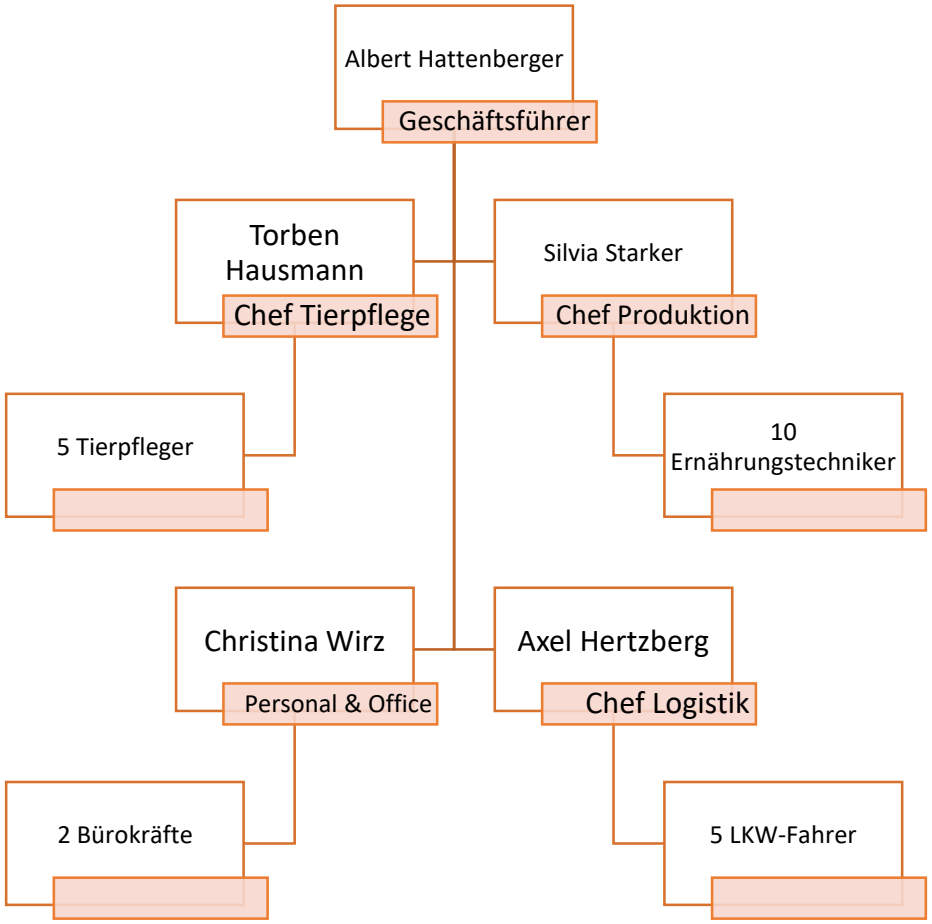
- 15% Expansion im nächsten Jahr
- **Mehr Automatisierung über IT System**



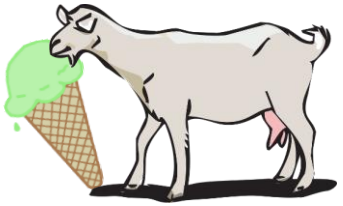
Requirements Elicitation



- Wen muss ich interviewen?
- Workshops?
- „Shadowing“?
- Wem zeige ich Prototypen?

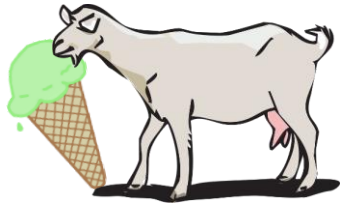


Initiale Requirements der GF



1. Neue Kunden müssen leicht hinzugefügt werden können
2. Pro Tag sollen etwa 50 Bestellungen einlangen können. Diese Bestellungen müssen für 5 Jahre gespeichert werden
3. Das Office und die bestehende Website kann 1. und 2. erledigen
4. Das System soll Rechnungen bei Auftragseingang automatisch generieren (aber nicht versenden!)
5. Das Office und nur das Office darf Rechnungsdetails ändern
6. Ein Wöchentlicher Bericht der unbezahlten Rechnung soll generiert werden
7. Rechnung müssen gedruckt werden können, oder per E-Mail dem Kunden geschickt
8. Produktionsabteilung will die bestellten Mengen ablesen können
9. Zahlungsbedingungen (z.B. Skonto) sollen anpassbar sein
10. Es soll eine Schnittstelle zum bestehenden Rechnungssystem geben

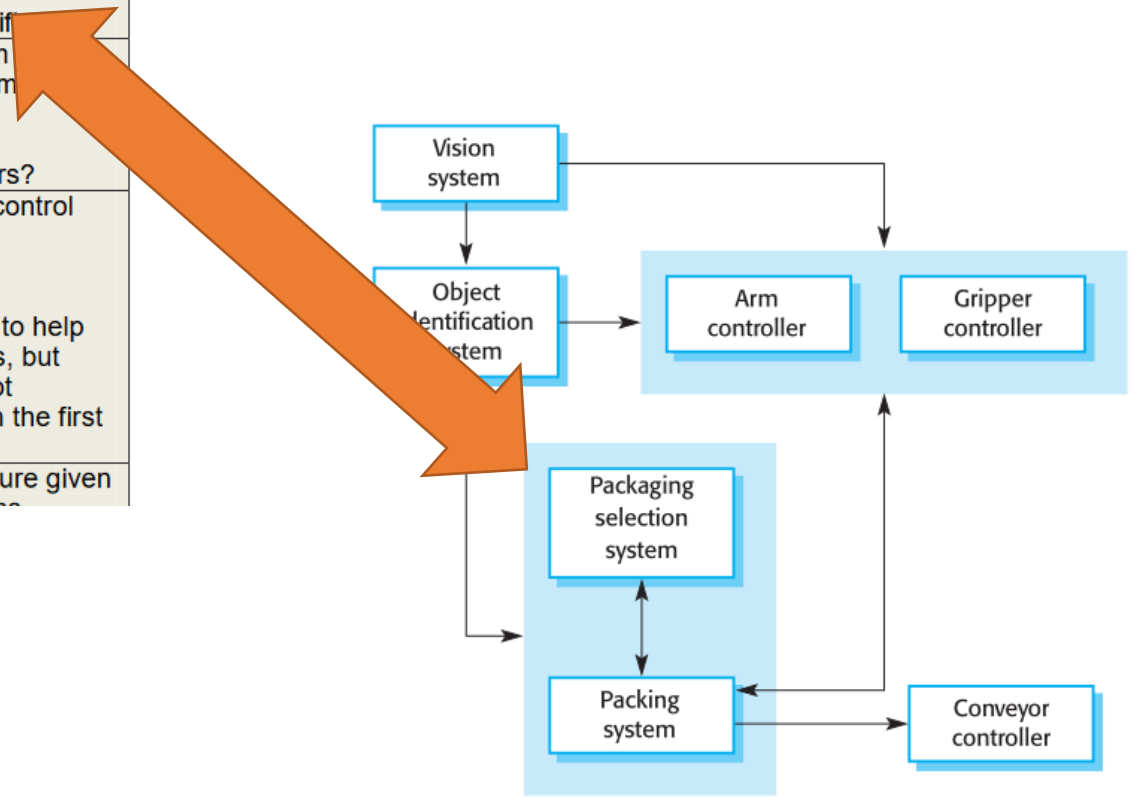
Requirement Specification



- Initiale Anforderungen oft unpräzise oder zu allgemein
  - Z.b.: „Das System soll userfreundlich sein“
- Einteilung in
  - Functional Requirements (FR) – Das System soll **<Objekt><Verb>**
  - Non-functional Requirements (NFR) – Das System soll **<Funktion>** auf diese Art und Weise ausüben
- Miro Übung!

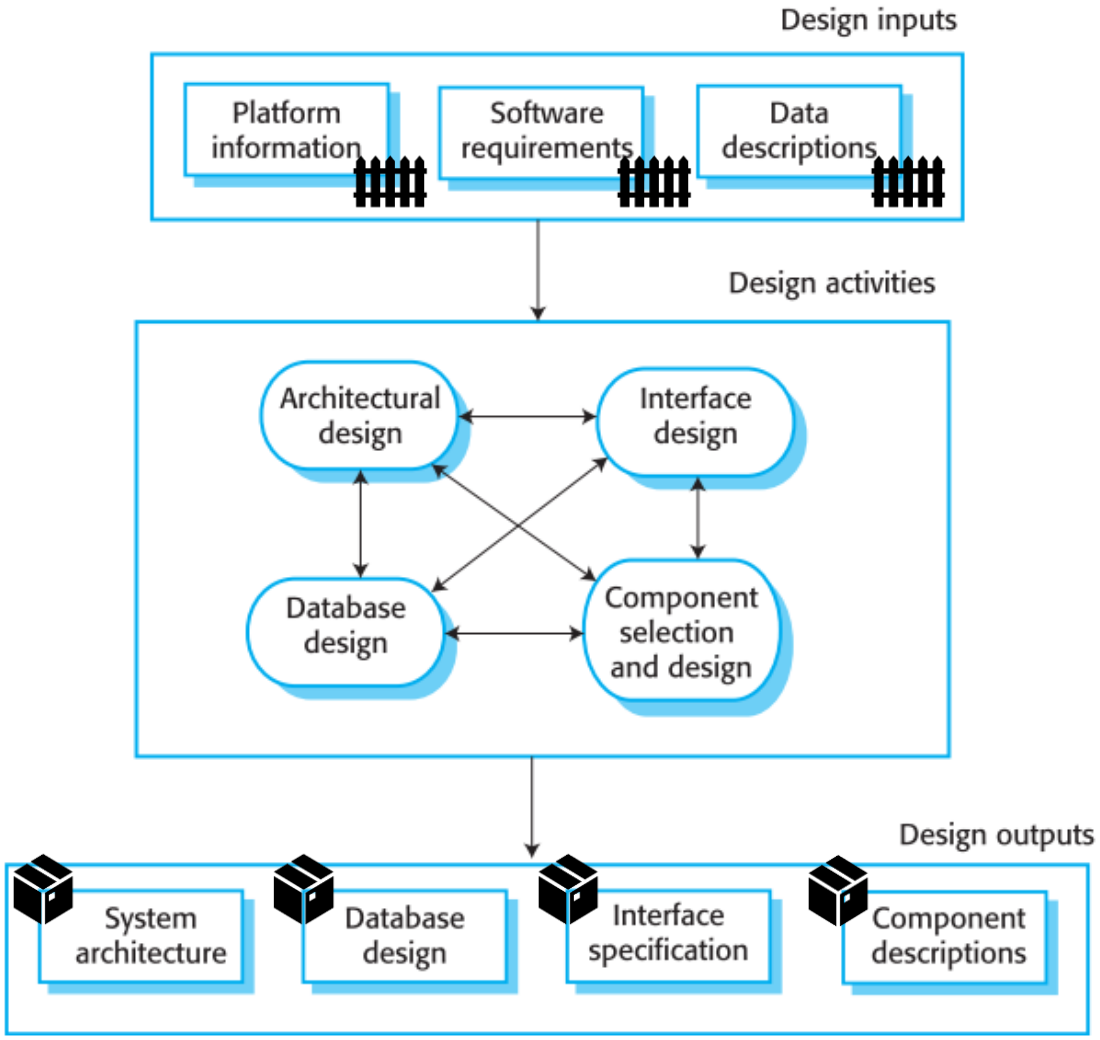
Wir haben Requirements – was nun?

	FR	*Priority*	*Comment*	*NFR*	*Priority*	*Comment*
1	Add Customer	**Must**	Vital to be able to record orders against the customers	Easily	Must	Vital if Customers need to be able to do this. Needs clarif
2	Record Order	**Must**	This is the whole point of the system so it is a Must	50 per day Retain for 5 years	Must Should	Need a firm especially m quantity Why 5 years?
3	This one must be prioritised against the NFR components			Access for Office Staff Access for Customers	Must Should	Important control MD wants customers to help themselves, but perhaps not essential in the first release.
4	Generate	**Must**	Key	Automatically,	Could	Useful feature given



Software Design

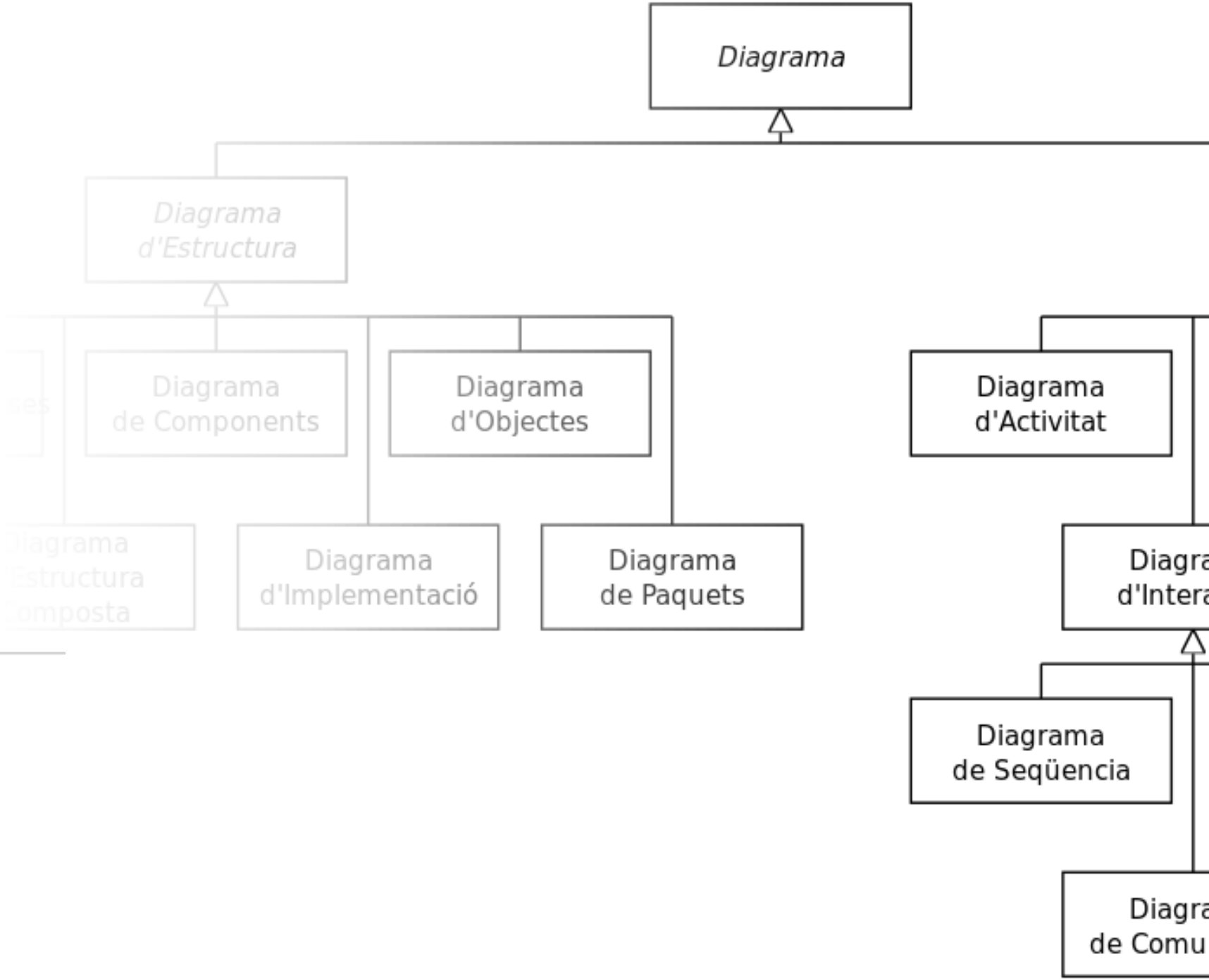
- Beschreibung der Struktur der geplanten Software
  - In der Zukunft
  - Status-quo (wenn schon vorhanden)
- Komponenten werden identifiziert und in Beziehung gesetzt



Entwicklerteam

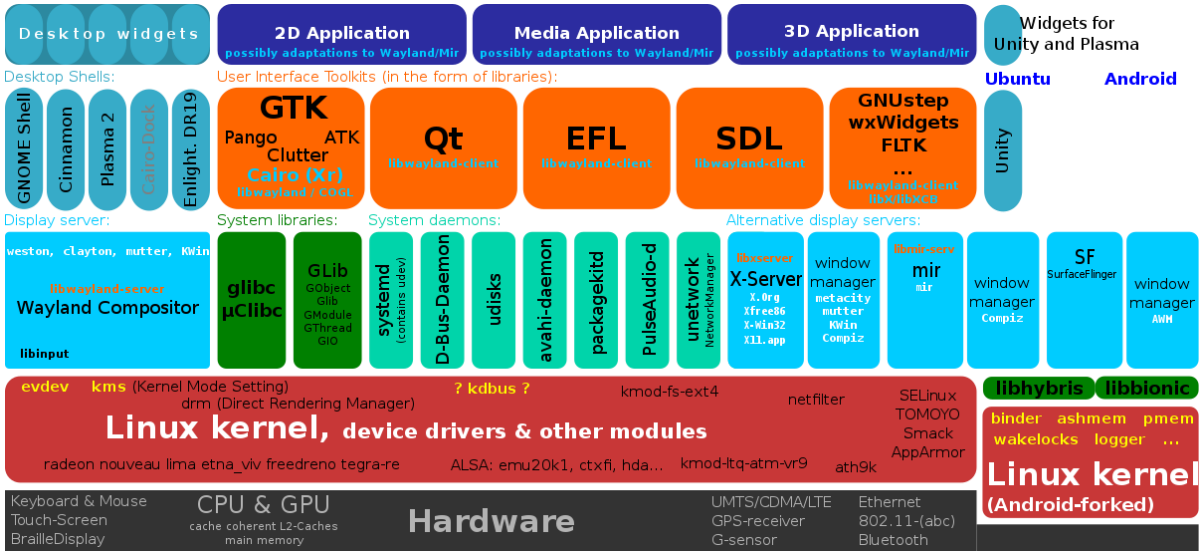
UML

Unified Modeling Language

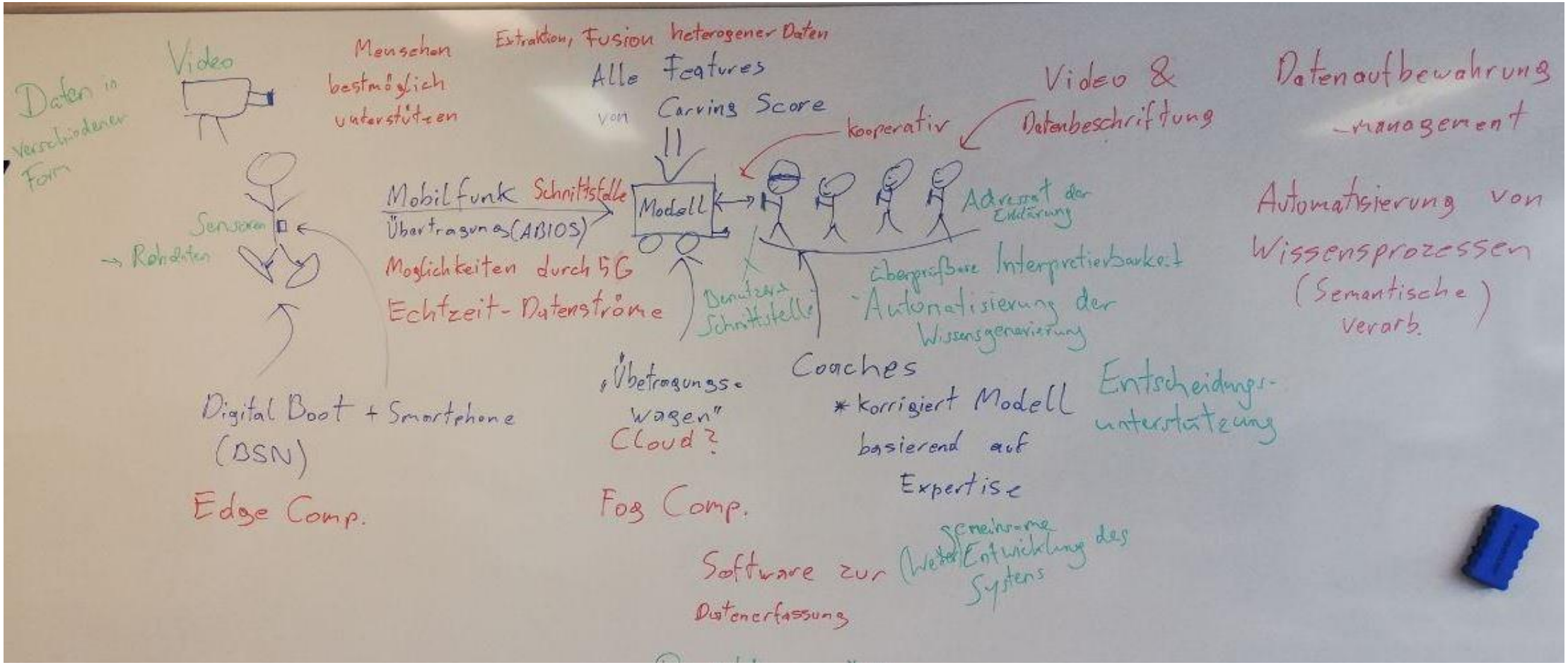


Modelle in der Architektur

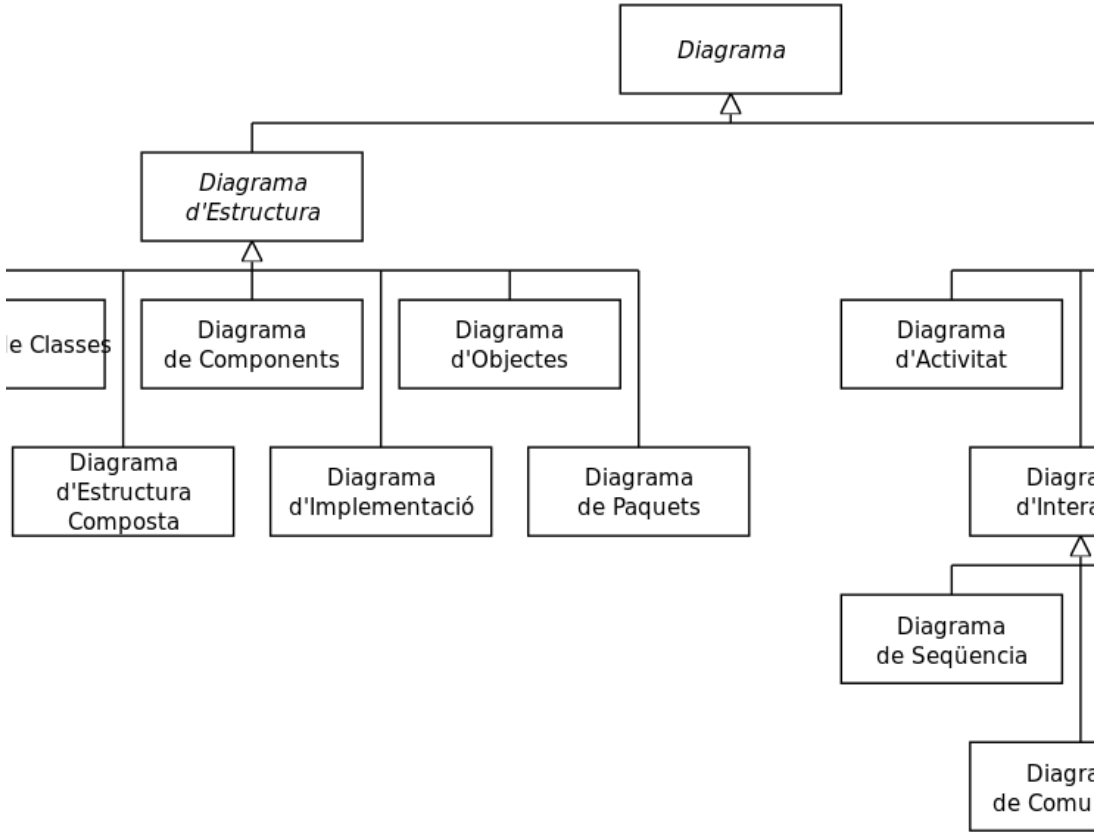
Modelle im Software Engineering



Quick & Dirty


Video
Menschen bestmöglich unterstützen
Extraktion, Fusion heterogener Daten
Alle Features von Carving Score
Video & Datenbeschriftung
kooperativ
Datenaufbewahrung -management
Daten in verschiedener Form
Sensoren
→ Rohdaten
Mobilfunk Schnittstelle Übertragung (ABIOS)
Möglichkeiten durch 5G
Echtzeit-Datenströme
Modell
Adressat der Erklärung
Benutzer Schnittstelle
überprüfbare Interpretierbarkeit
-Automatisierung der Wissensgenerierung
Automatisierung von Wissensprozessen (Semantische) verarb.
Digital Boot + Smartphone (DSN)
Edge Comp.
„Übertragungswagen" Cloud?
Fog Comp.
Coaches
* korrigiert Modell basierend auf Expertise
Entscheidungs-unterstützung
Software zur Datenerfassung
gemeinsame (Weiter-)Entwicklung des Systems

UML als Werkzeug für Software-Modelle

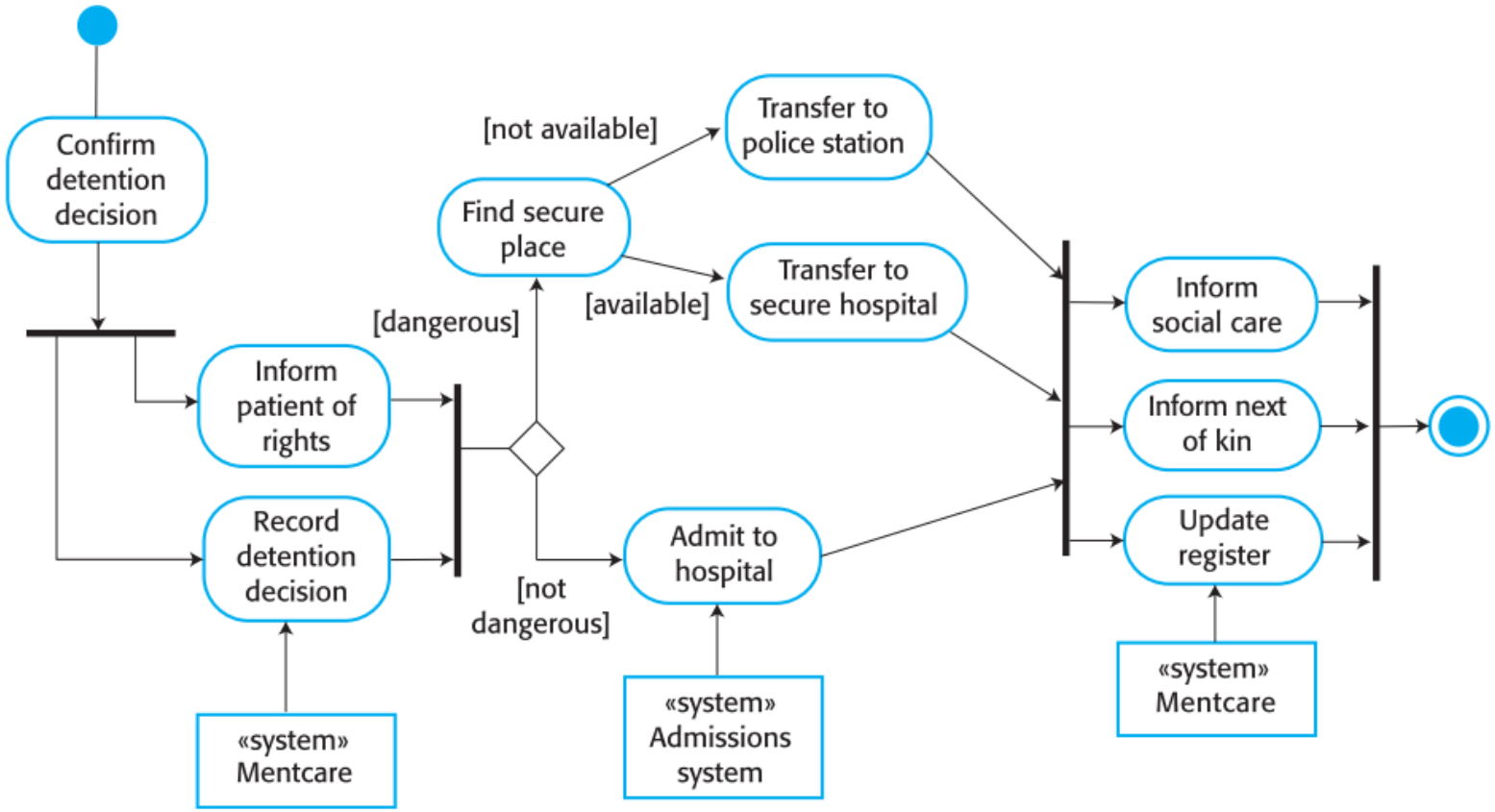


Unified Modeling Language (UML 2)

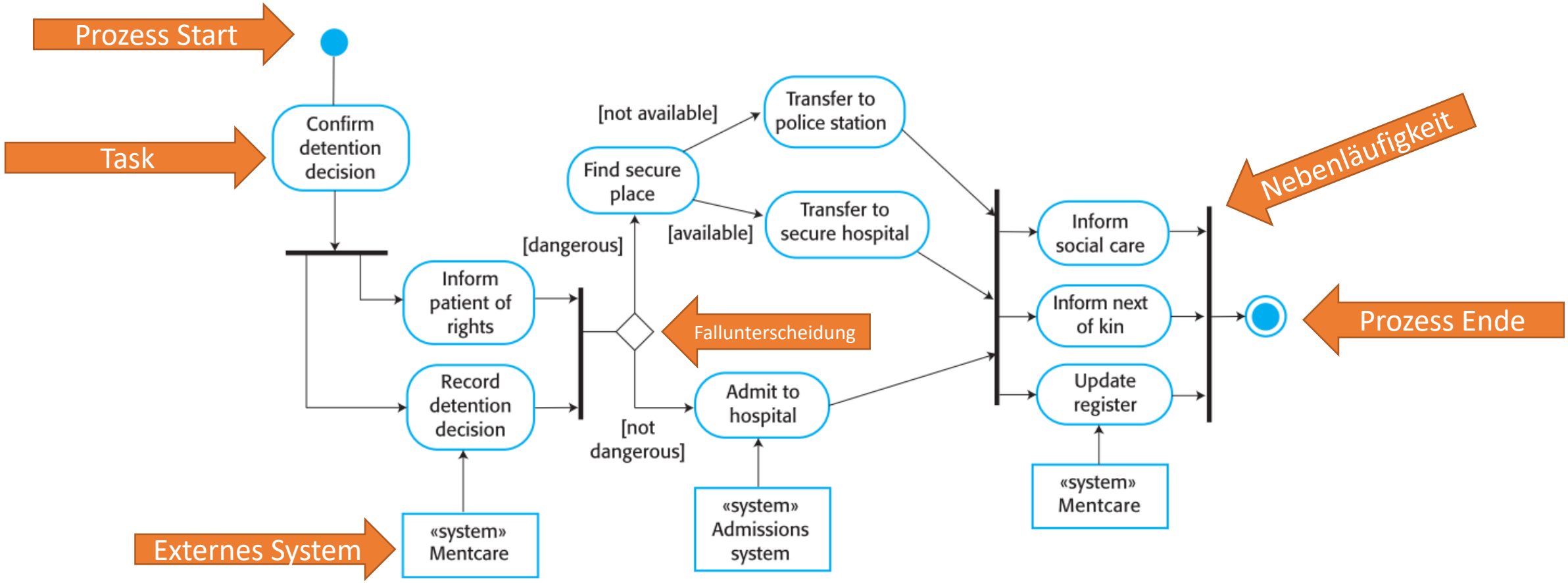
- 13 Diagrammtypen
  - davon reichen 5 Typen meist aus (Erickson & Siau, 2007)

- Activity Diagram
  Aktivitäten in einem Prozess
- Use Case Diagram
  Interaktionen zwischen System und Context
- Sequence Diagram
  Interaktionen zwischen Akteuren, dem System, und den System Komponenten
- Class Diagram
  Objektklassen und deren Beziehungen
- State Diagram
  Reaktion des Systems auf interne und externe Events

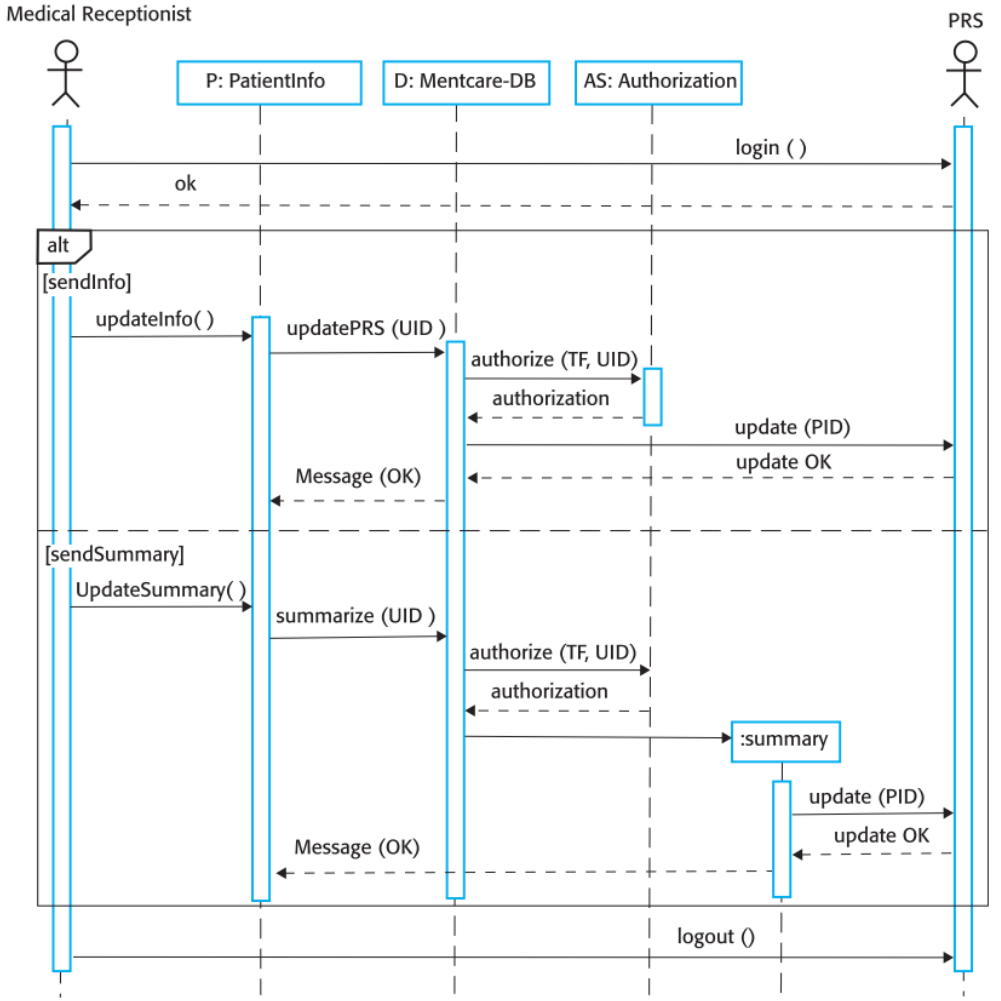
Prozessmodelle (UML Activity Diagram)



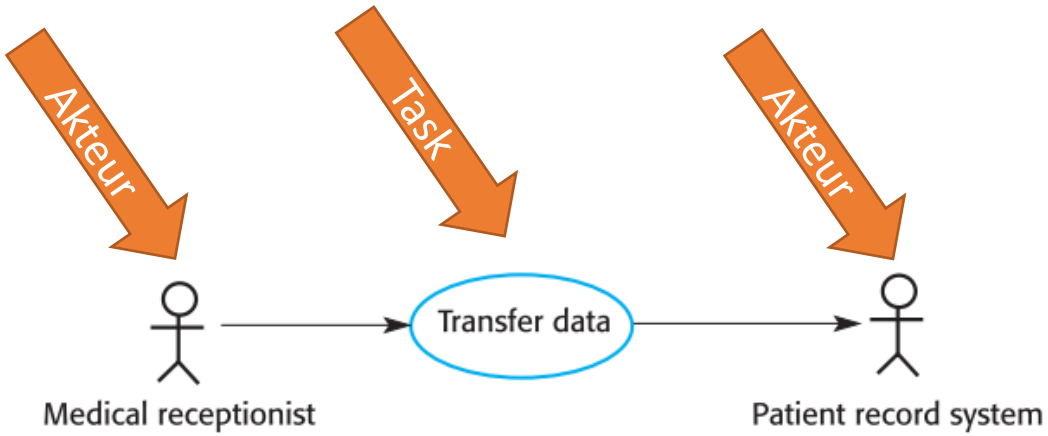
Prozessmodelle (UML Activity Diagram)

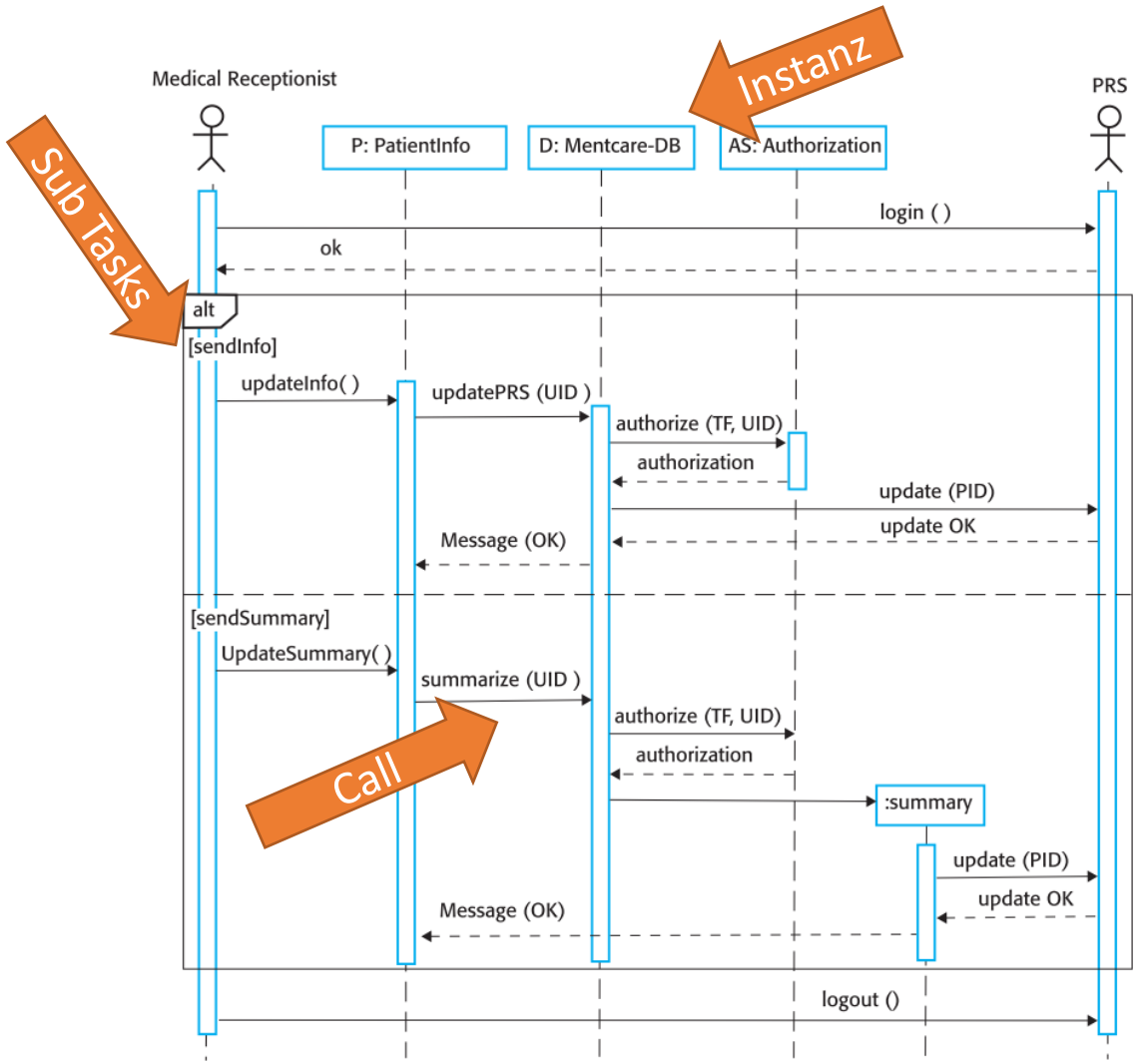


Interaktionsmodelle (Use Case / Sequence)

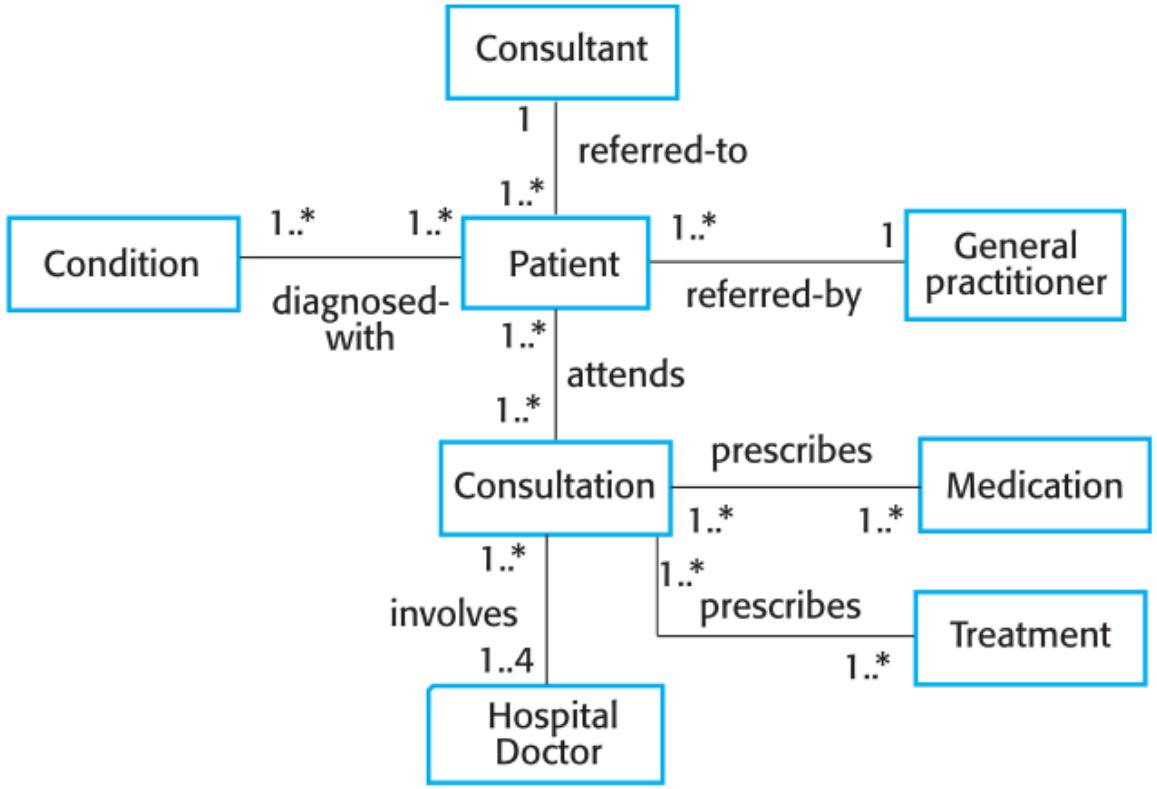


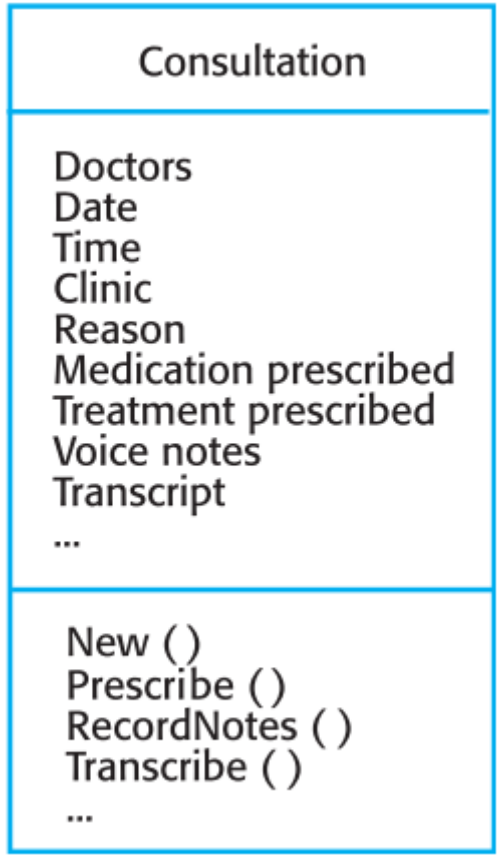
Interaktionsmodelle (Use Case / Sequence)



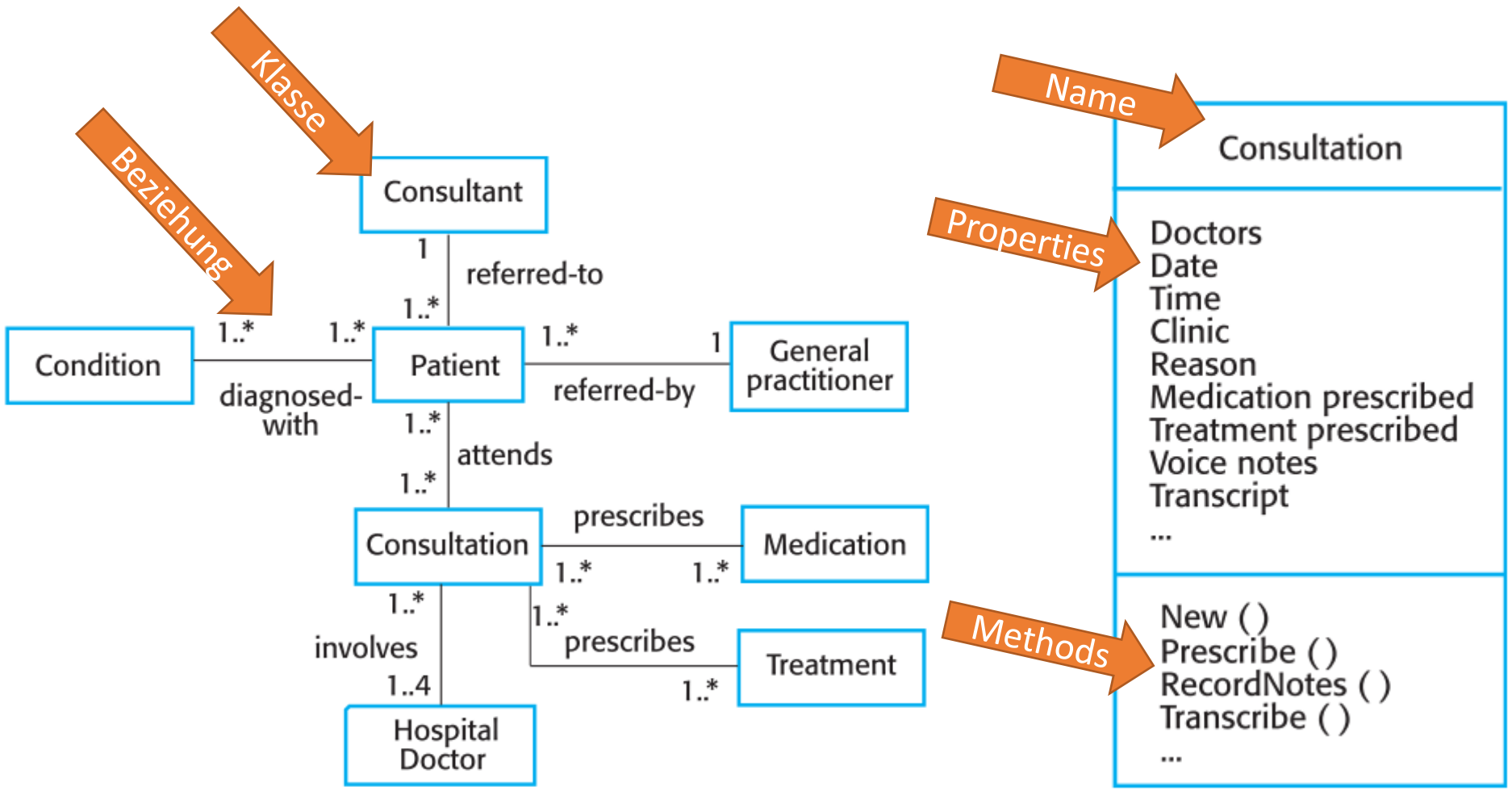


Strukturmodelle (Class Diagram)





Strukturmodelle (Class Diagram)



Verhaltensmodelle (State Diagram)

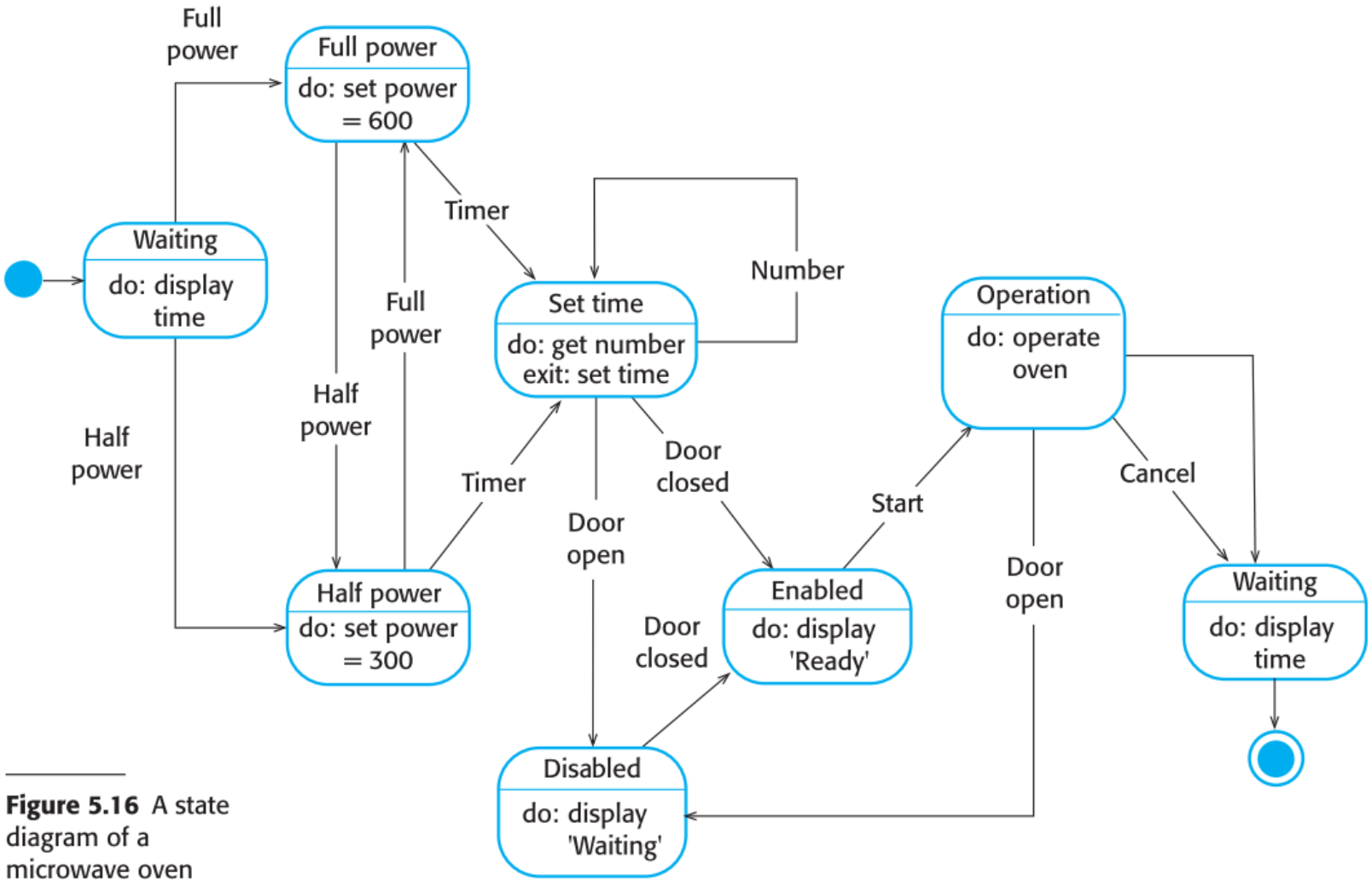


Figure 5.16 A state diagram of a microwave oven

Verhaltensmodelle (State Diagram)

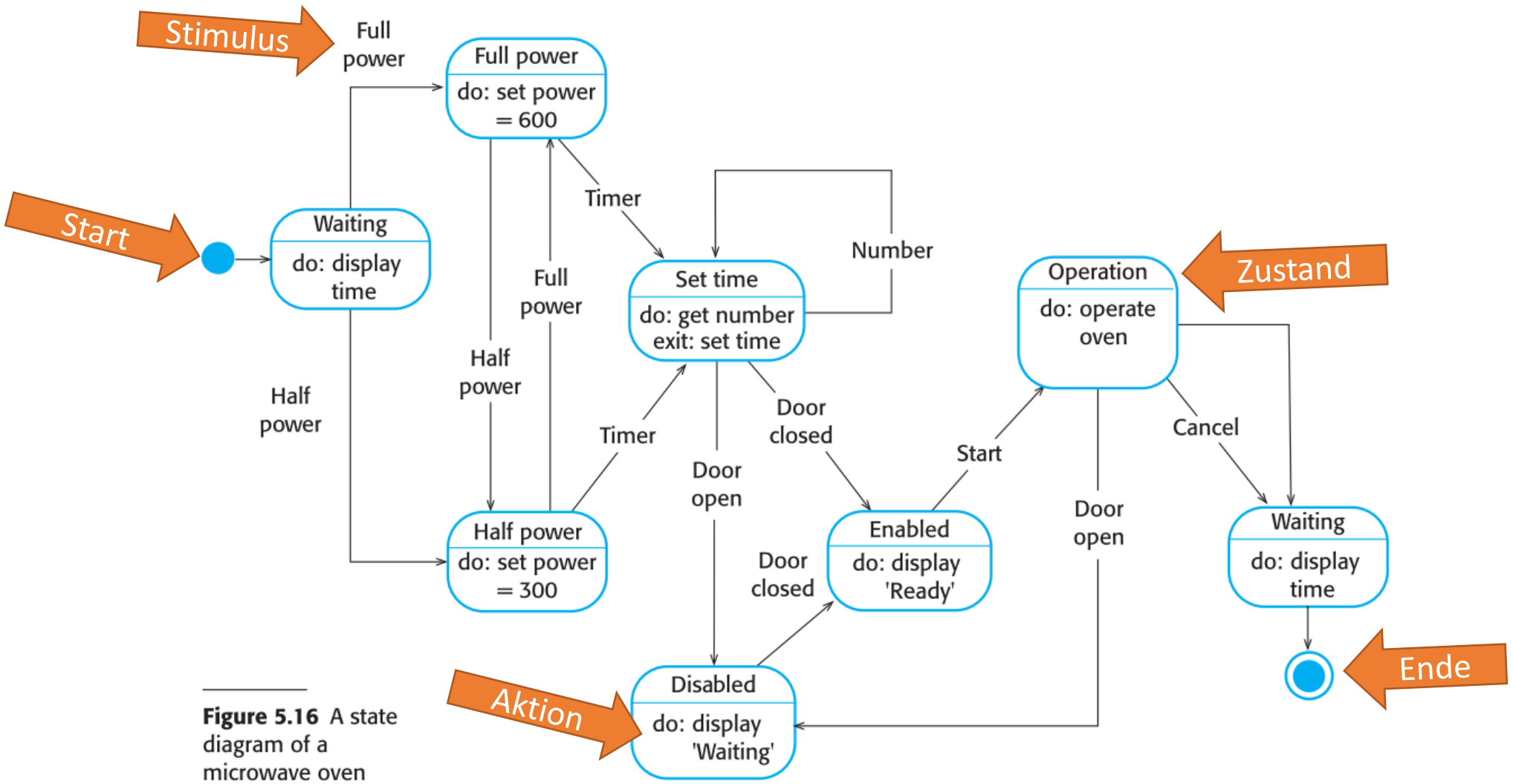
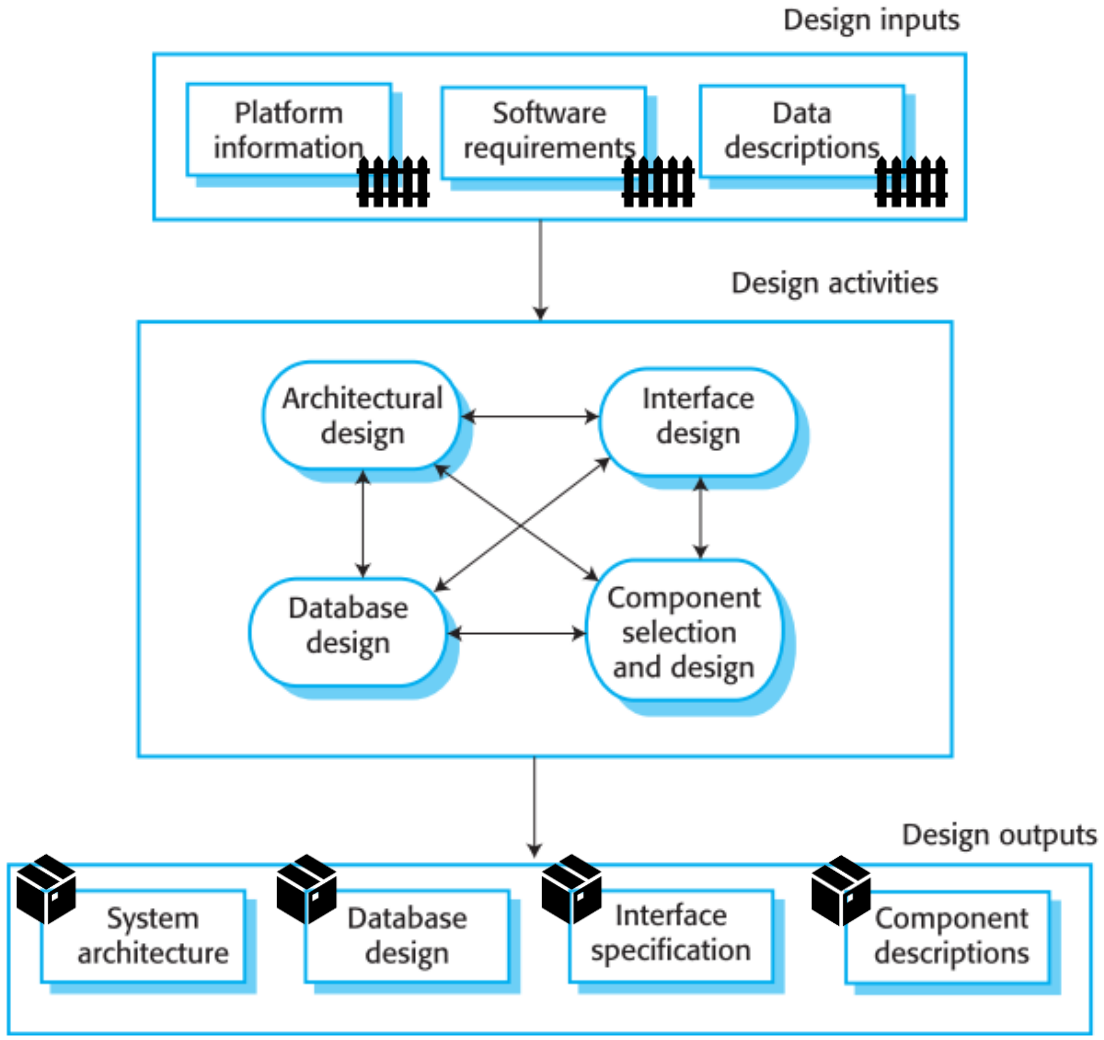


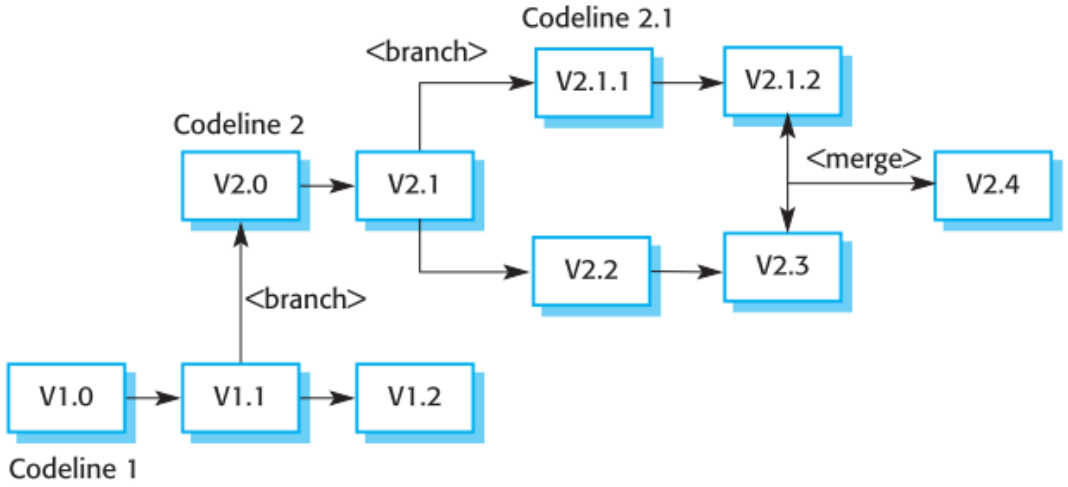
Figure 5.16 A state diagram of a microwave oven

Software Design & Implementierung

Entwicklerteam



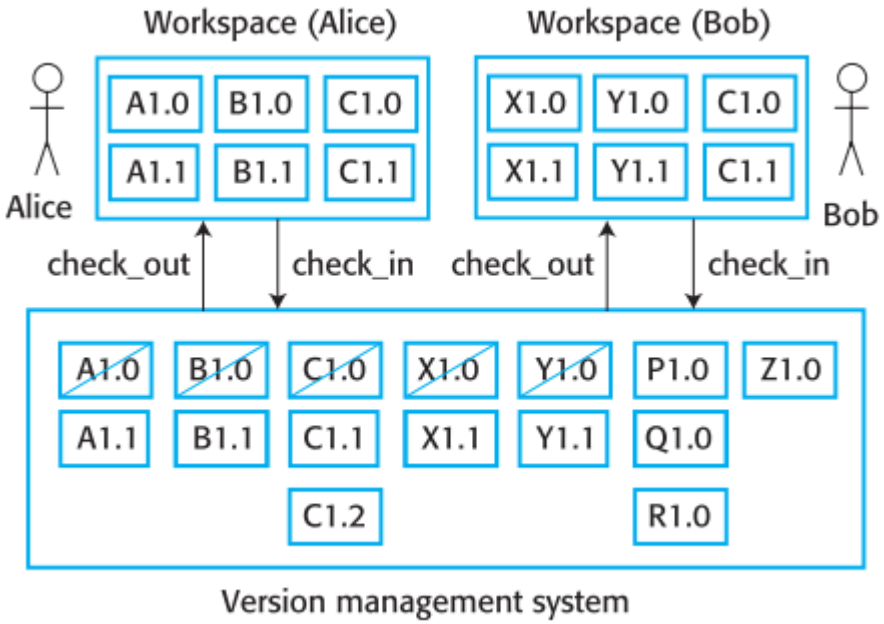
Software Versionierung



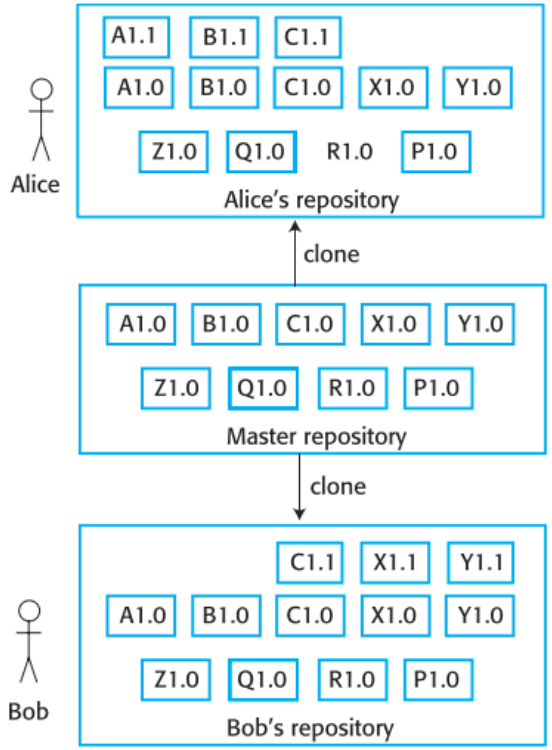
- Problem: Software entwickelt sich täglich…
  - … Nachvollziehen von Änderungen?
  - … wo liegt die aktuellste Version?
  - … Kollaboration?
  - … Archivierung alter Versionen?
  - … Konfliktauflösung?

Tool: Version Control Systems (VCS)

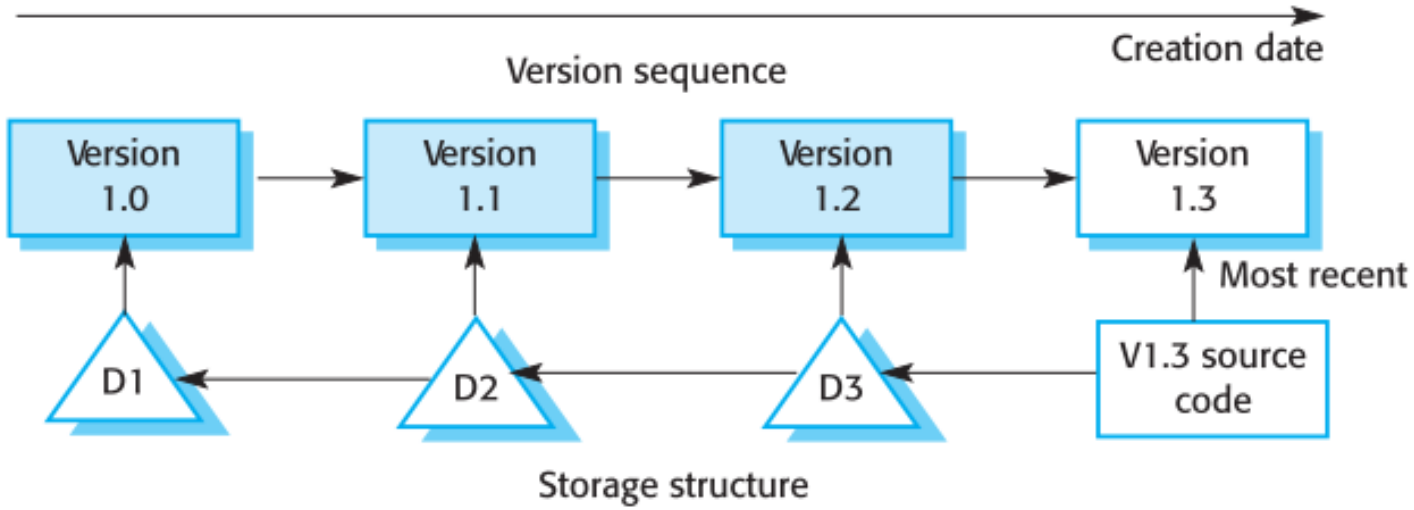
Zentralisierte VCS



Dezentralisierte VCS

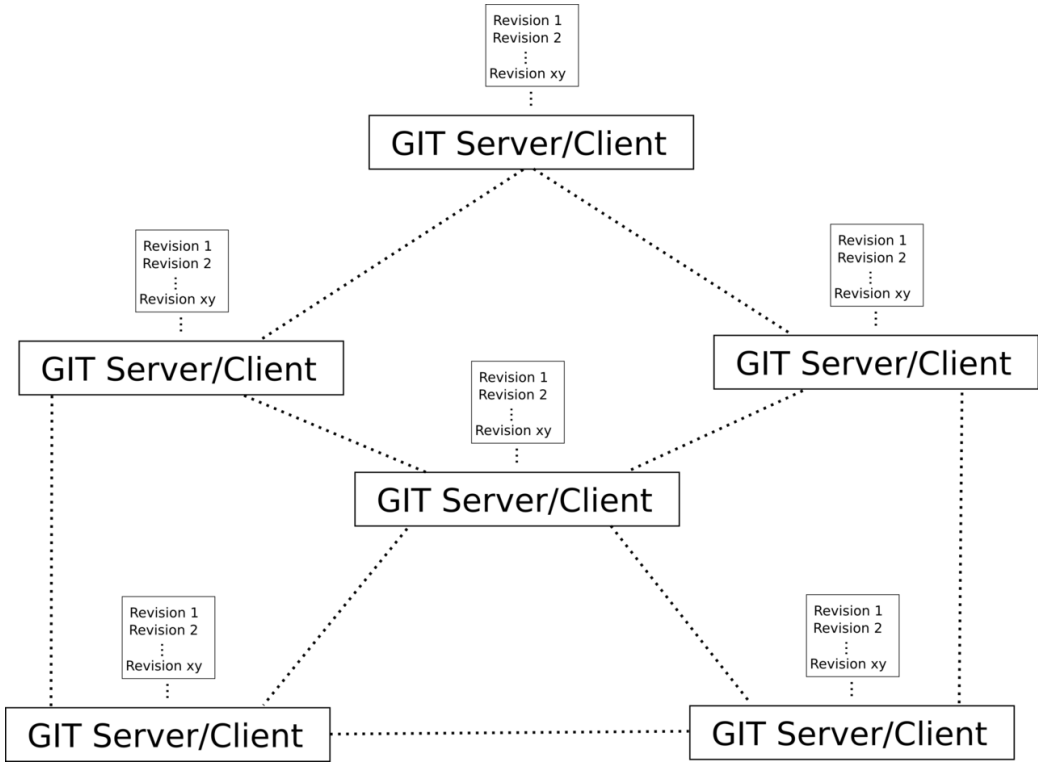


Speicheroptimierung (Delta Storage)

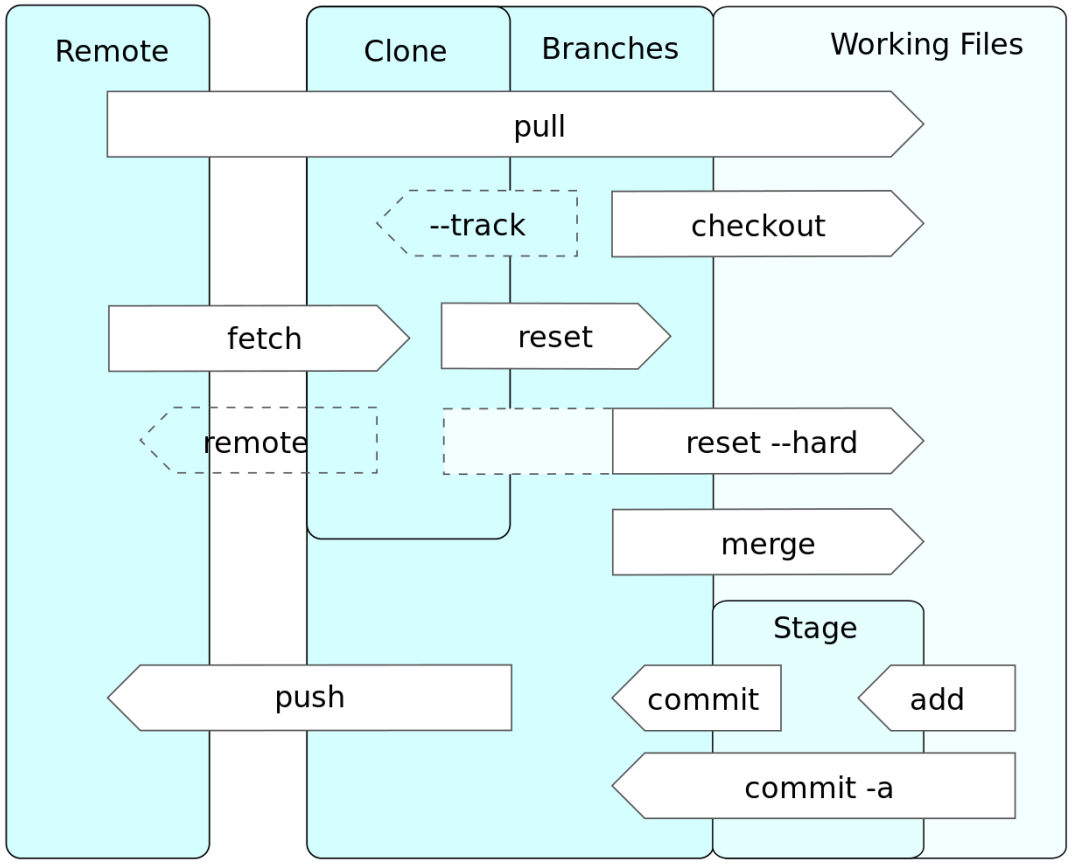


- Open Source
- 2005 von Linus Torvalds
- Heute Junio Harmano

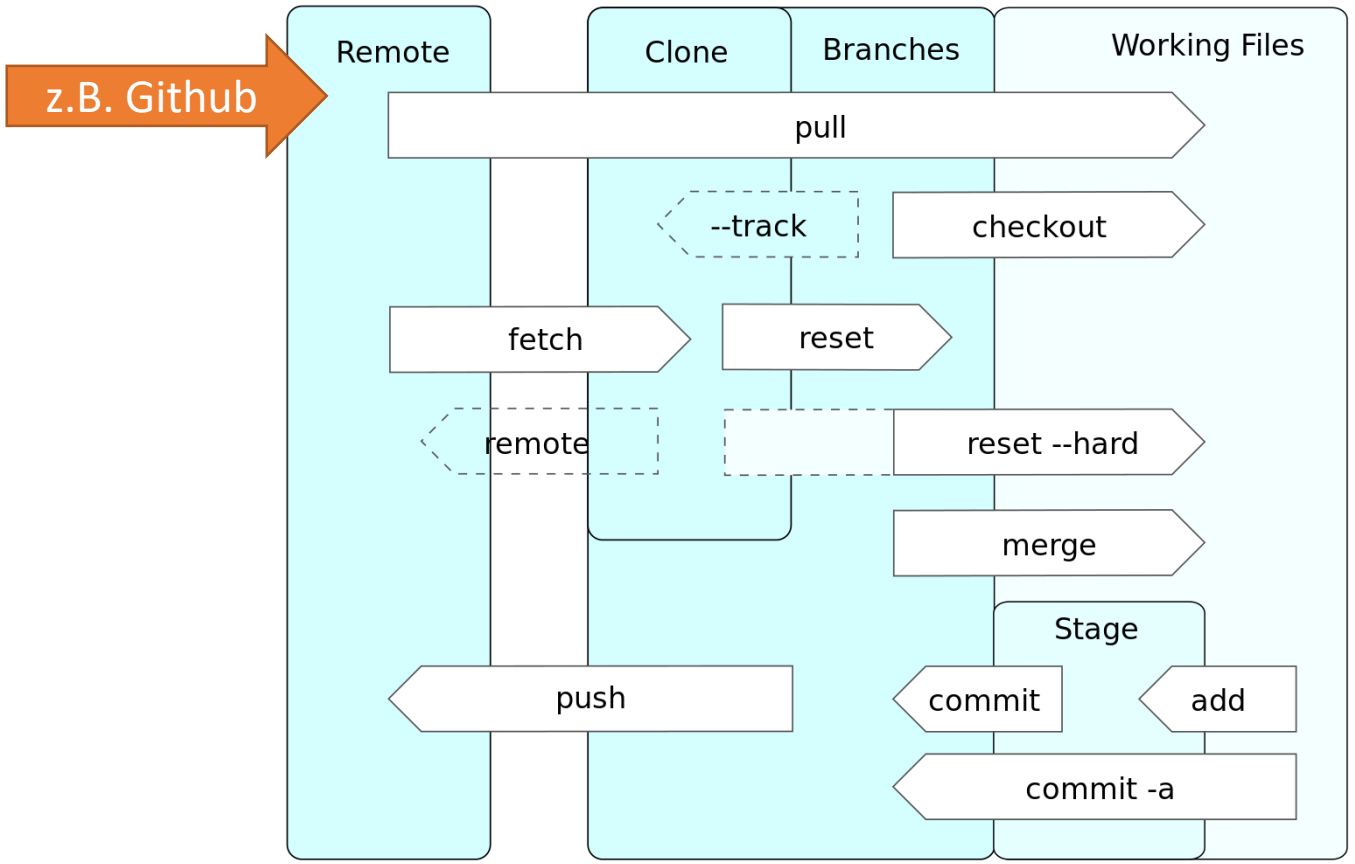
- 69% der registrierten Softwareprojekte auf Open Hub (2019)
- Unixartige & Windows



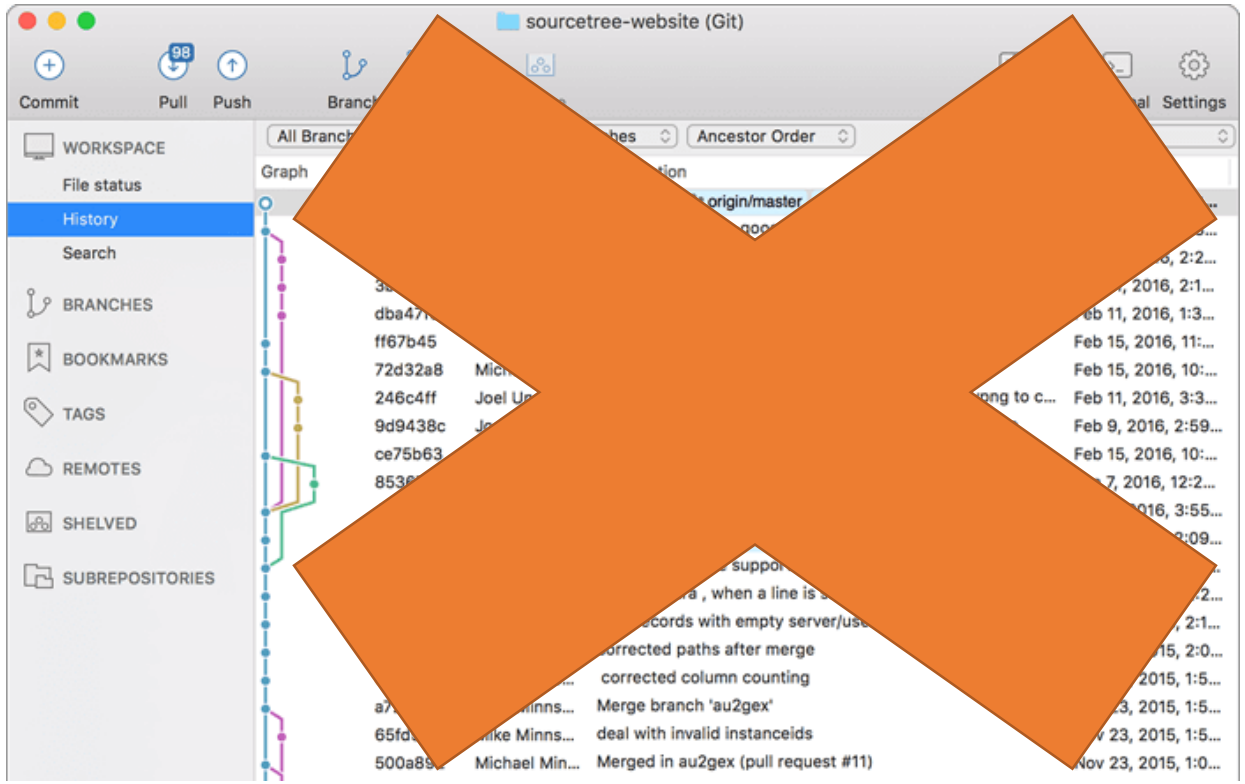
Datenfluss


Remote
Clone
Branches
Working Files
pull
--track
checkout
fetch
reset
remote
reset --hard
merge
Stage
push
commit
add
commit -a

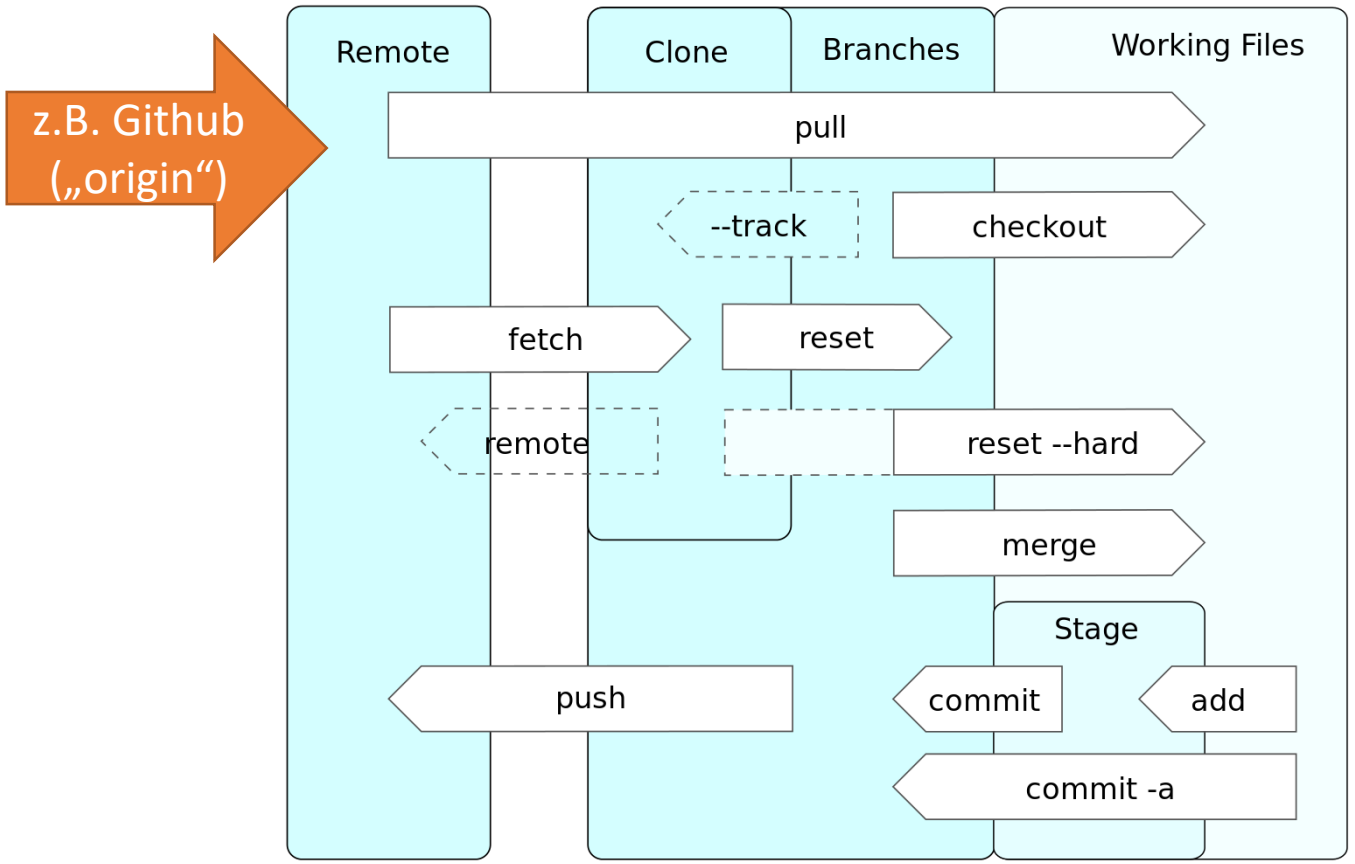
Datenfluss


z.B. Github
Remote
Clone
Branches
Working Files
pull
--track
checkout
fetch
reset
remote
reset --hard
merge
Stage
push
commit
add
commit -a

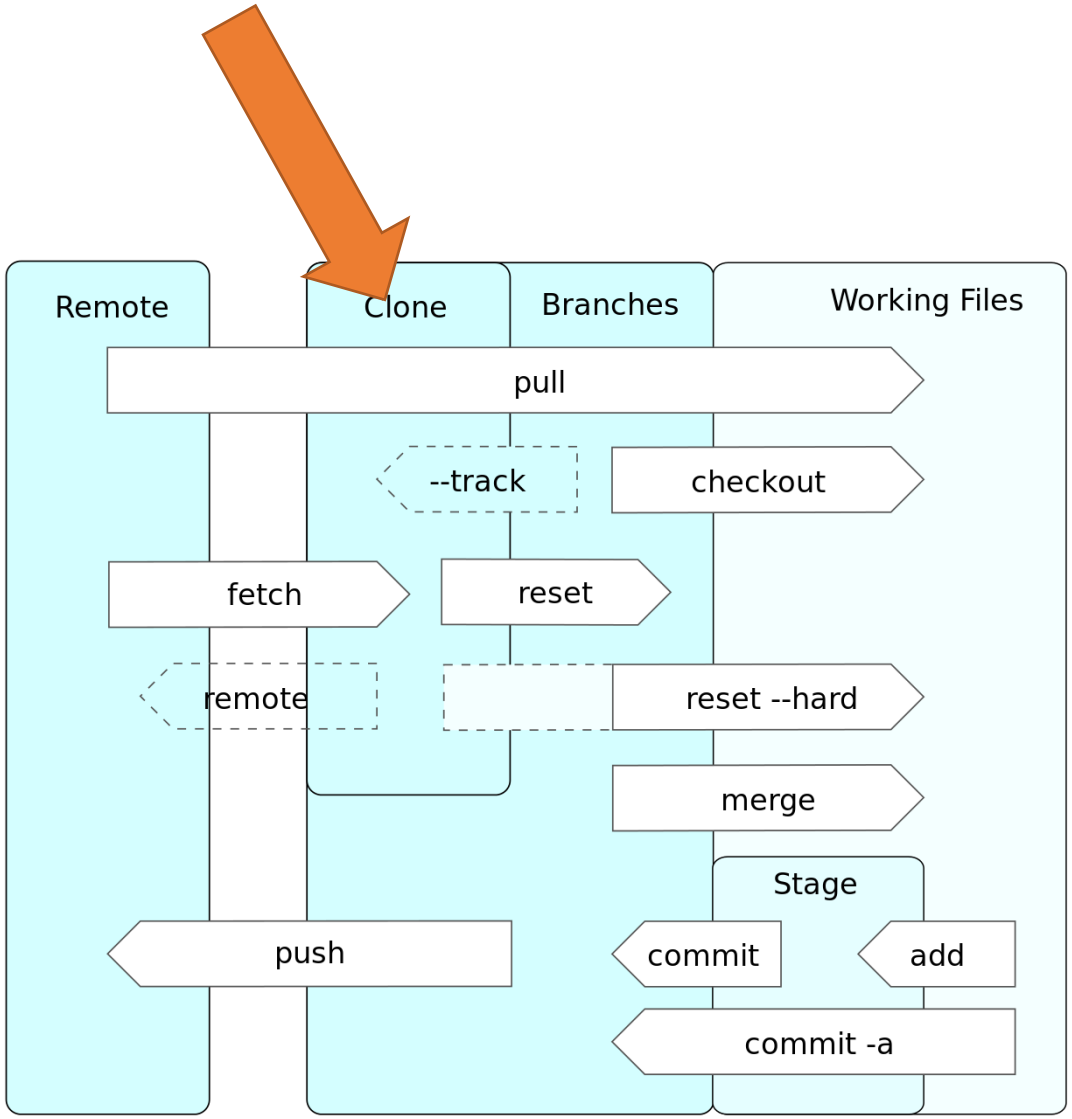
GUI? Brauchen wir nicht ;)



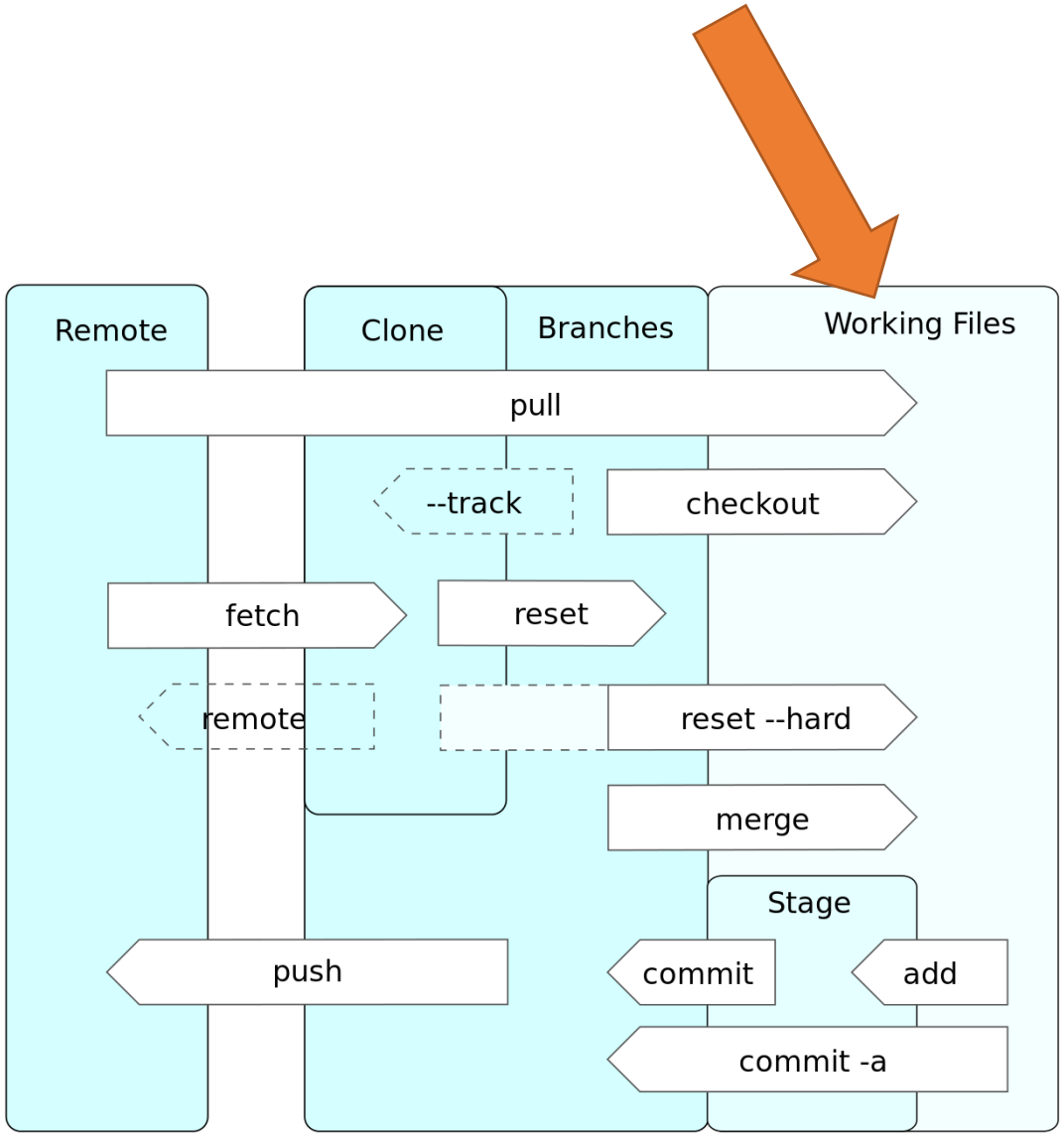
Datenfluss


z.B. Github
(„origin“)
Remote
Clone
Branches
Working Files
pull
--track
checkout
fetch
reset
remote
reset --hard
merge
Stage
push
commit
add
commit -a

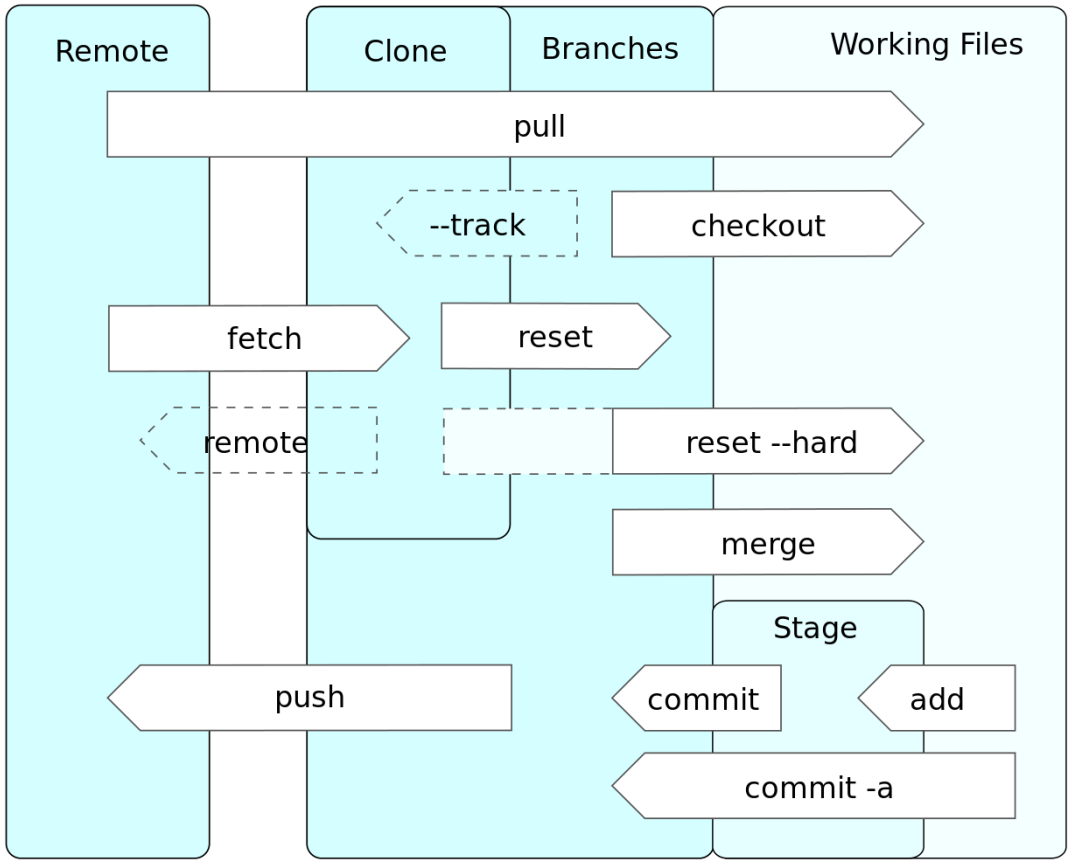
Datenfluss


Remote
Clone
Branches
Working Files
pull
--track
checkout
fetch
reset
remote
reset --hard
merge
Stage
push
commit
add
commit -a

Datenfluss

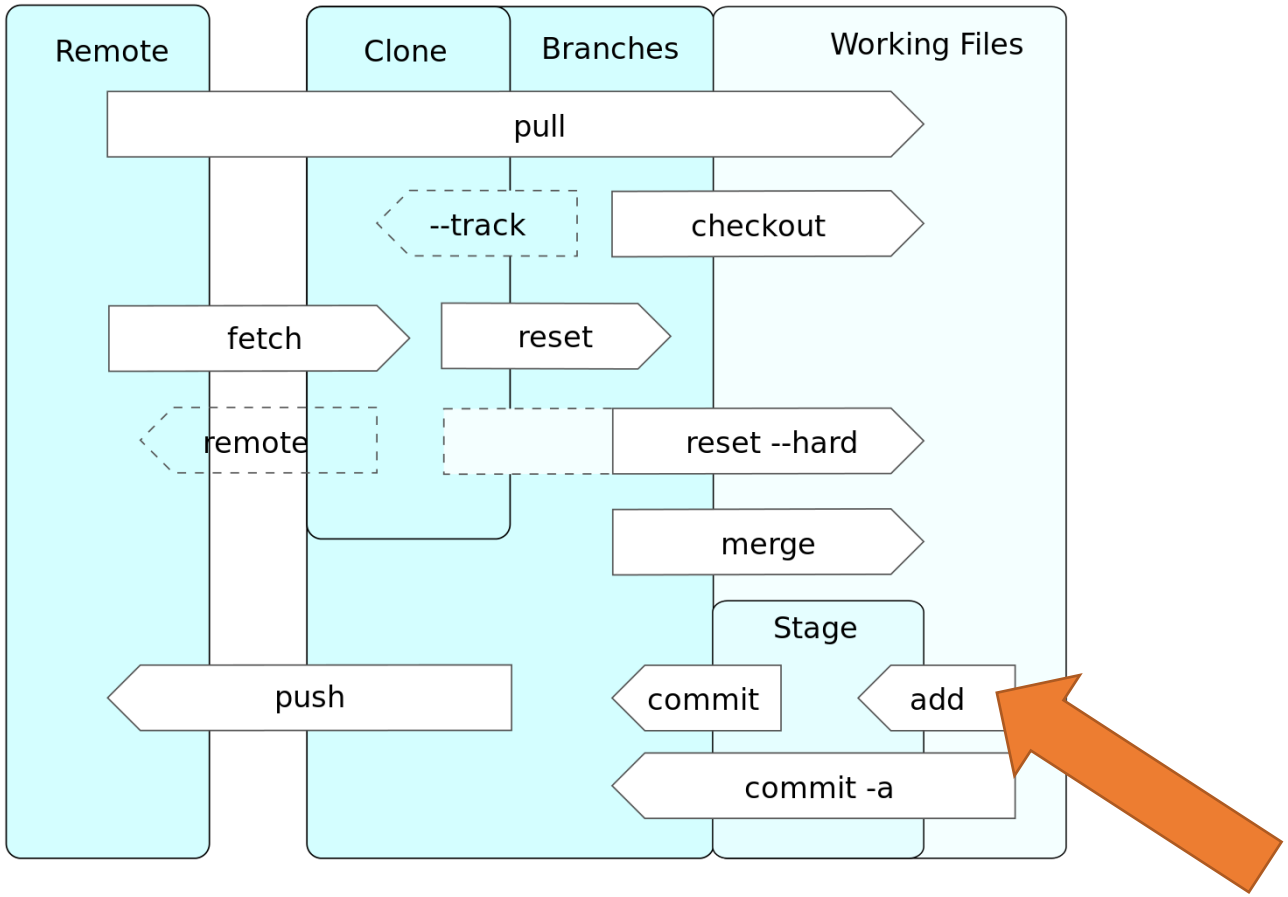

Remote
Clone
Branches
Working Files
pull
--track
checkout
fetch
reset
remote
reset --hard
merge
Stage
push
commit
add
commit -a

Datenfluss

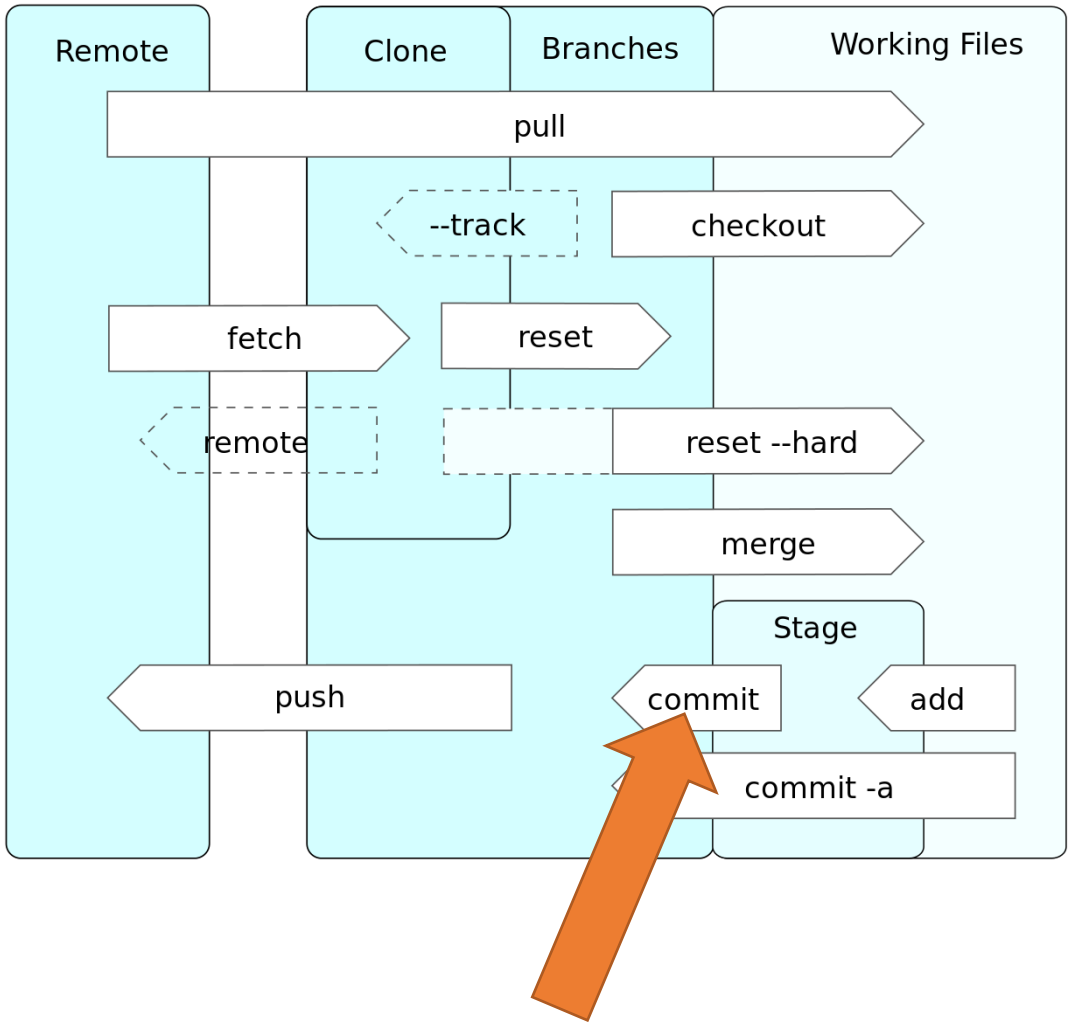

Remote
Clone
Branches
Working Files
pull
--track
checkout
fetch
reset
remote
reset --hard
merge
Stage
push
commit
add
commit -a


pre {
margin: 5
line-heigh
padding:
overflow:
</>

Datenfluss


Remote
Clone
Branches
Working Files
pull
--track
checkout
fetch
reset
remote
reset --hard
merge
Stage
push
commit
add
commit -a

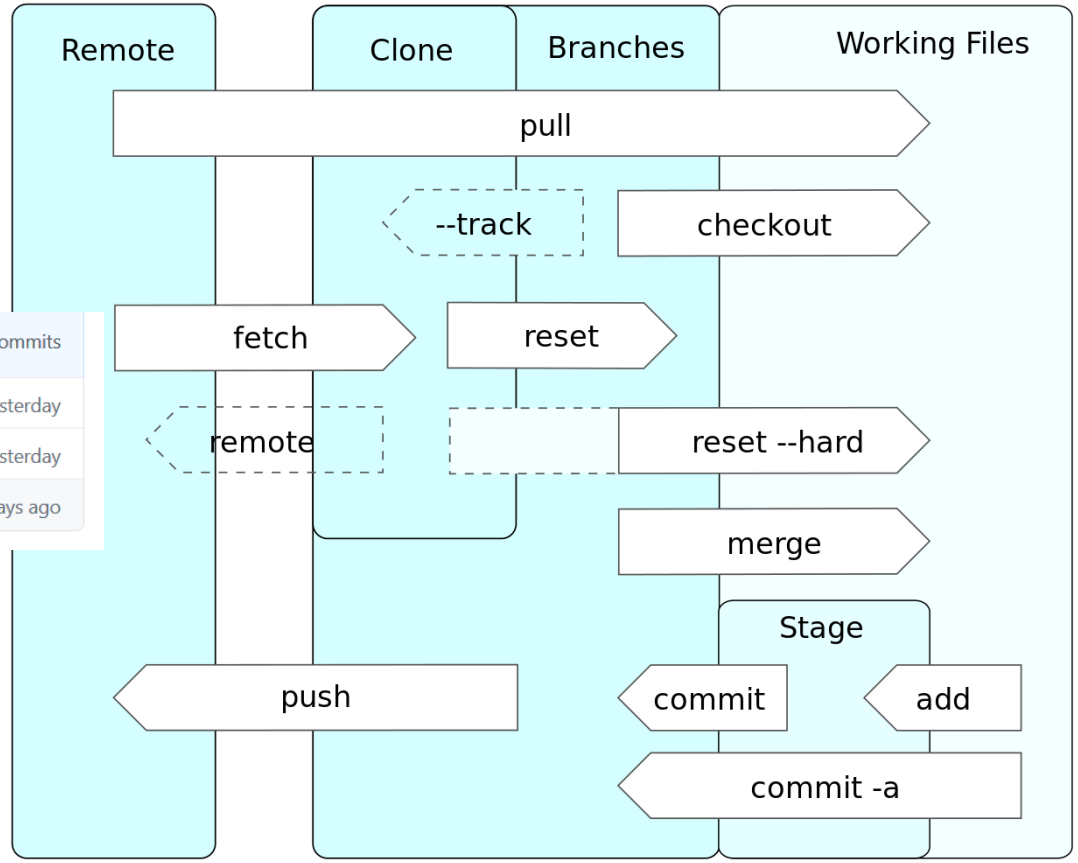
Datenfluss


Remote
Clone
Branches
Working Files
pull
--track
checkout
fetch
reset
remote
reset --hard
merge
Stage
push
commit
add
commit -a

Datenfluss

wokremse pattern examples — 9dd2da2 yesterday — 3 commits

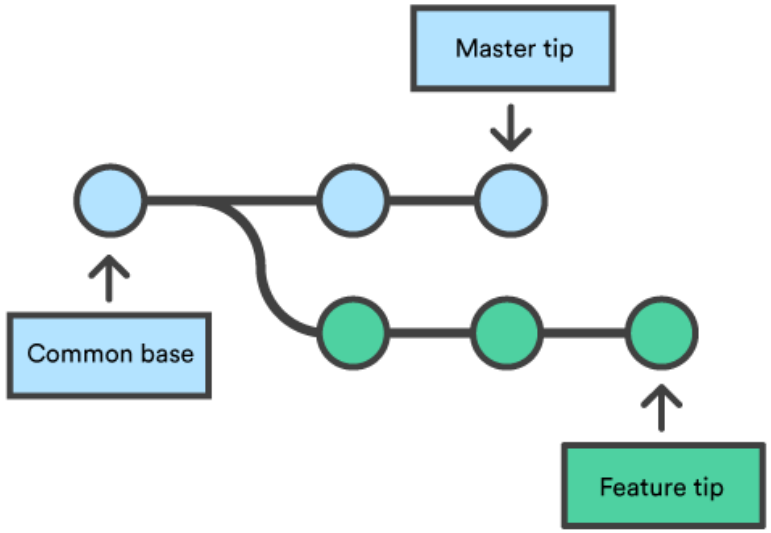
src/patterns	pattern examples	yesterday
.gitignore	added git ignore	yesterday
README.md	Initial commit	10 days ago

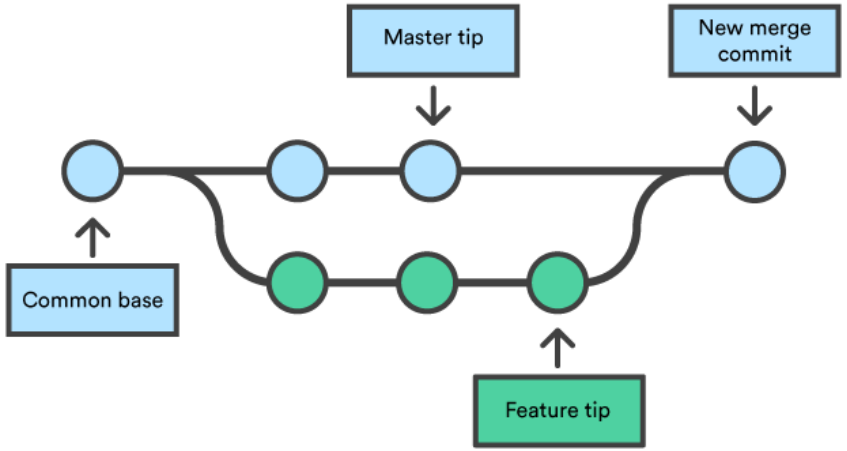


Git Übung

1. Klont das swe-se-2020 Repository
   git clone https://github.com/kremserwo/swe-se-2020
2. Erstellt einen Branch <nachname>_hello_world
   git branch <branch>
3. Pusht den neuen Branch auf Origin
   git push –u origin <branch>
4. Erstellt in eurem lokalen Repository eine Datei „hello_world.txt“
5. Fügt die neue Datei eurer Staging-Umgebung hinzu
   git add ./hello_world.txt
6. Commitet euer Repository
   git commit .
7. Pusht euer Repository auf Origin
   git push

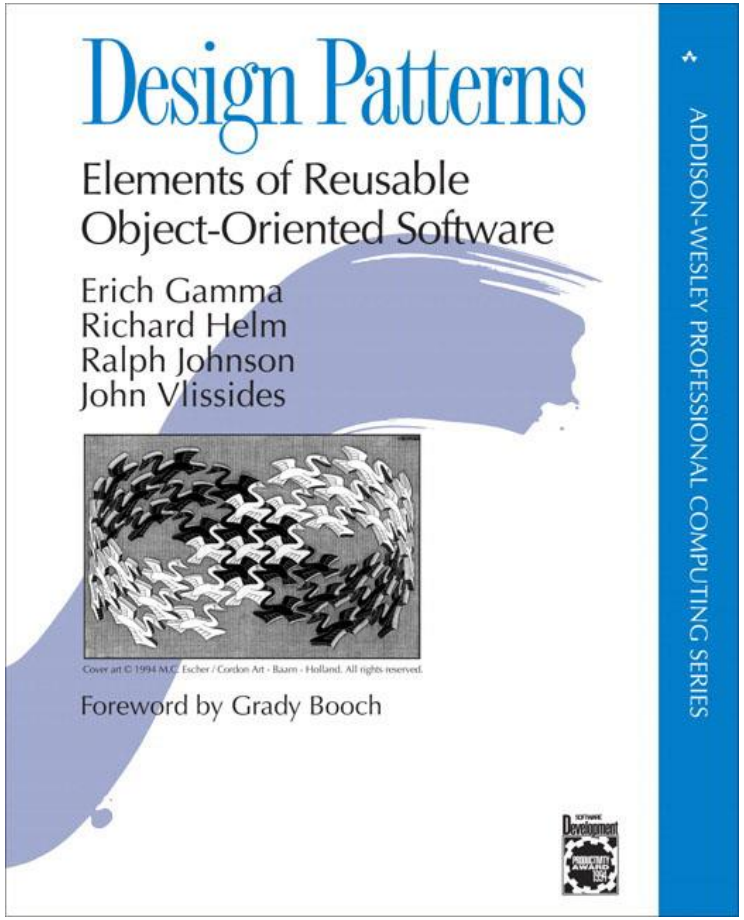
Merge & Konflikte





Entwurfsmuster (Design Patterns)

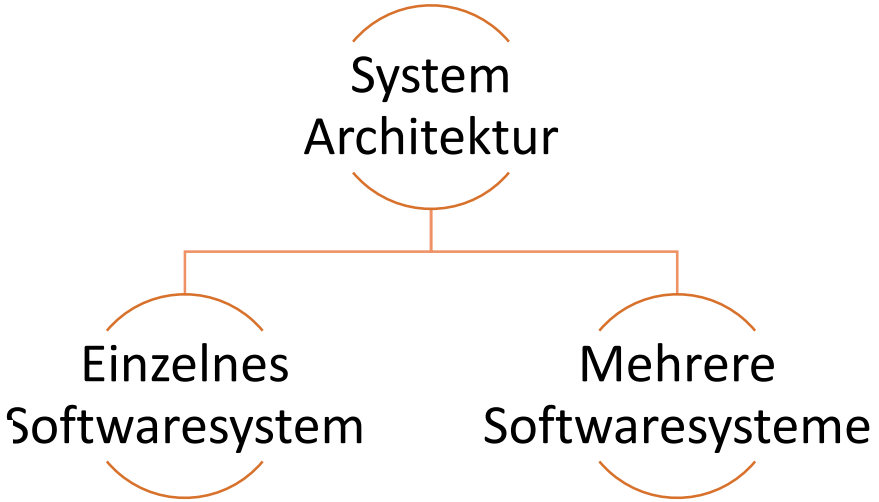
- „Gang of Four (GoF)“
- Best Practice für wiederkehrende Probleme im Software Design
- Gedankliche Abkürzung
- „Gibt es da vielleicht schon ein Pattern?“
- Frameworks implementieren diese Patterns
- https://springframework.guru/gang-of-four-design-patterns/

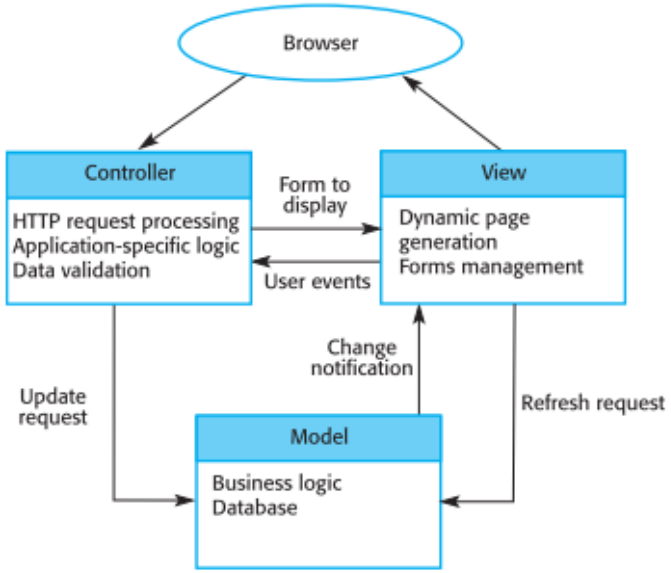


„Fertige“ Problemlösungen

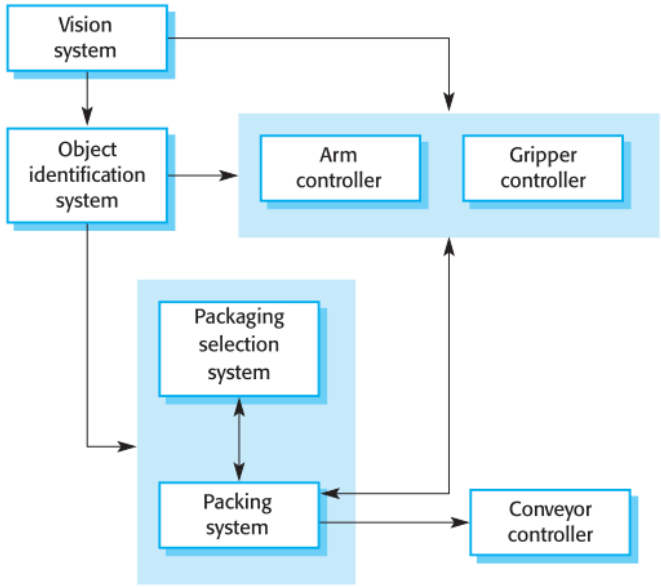
- Mehrere Objekte müssen über die Statusänderung eins anderen Objekts informiert werden?
  - Observer Pattern!
- Interfaces „aufräumen“?
  - Facade Pattern!
- Elemente in einer Collection nacheinander durchgehen, egal welche Datenstruktur dahinter liegt?
  - Iterator Pattern!
- Die Funktionalität einer Klasse in Runtime ändern?
  - Decorator Pattern!

Zwei Abstraktionsebenen





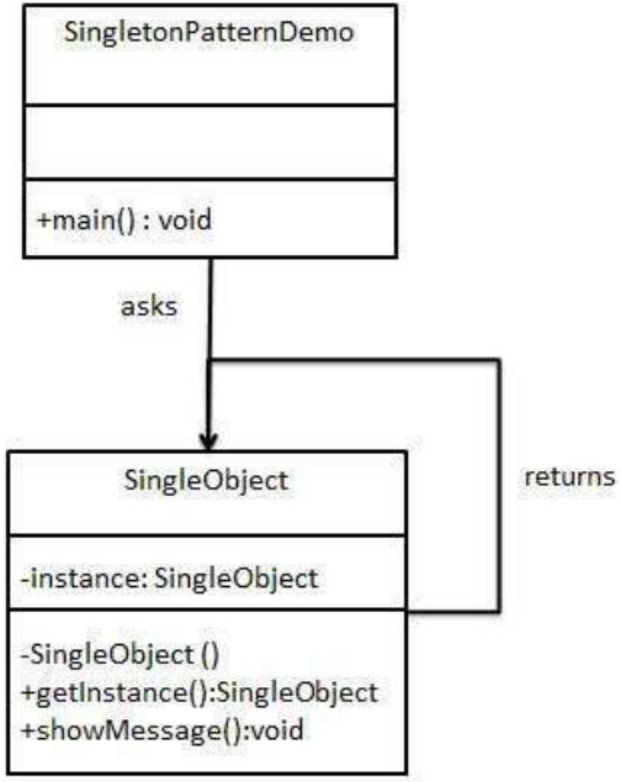
Beispiele



UML von: https://www.tutorialspoint.com/design_pattern/

Singleton Pattern

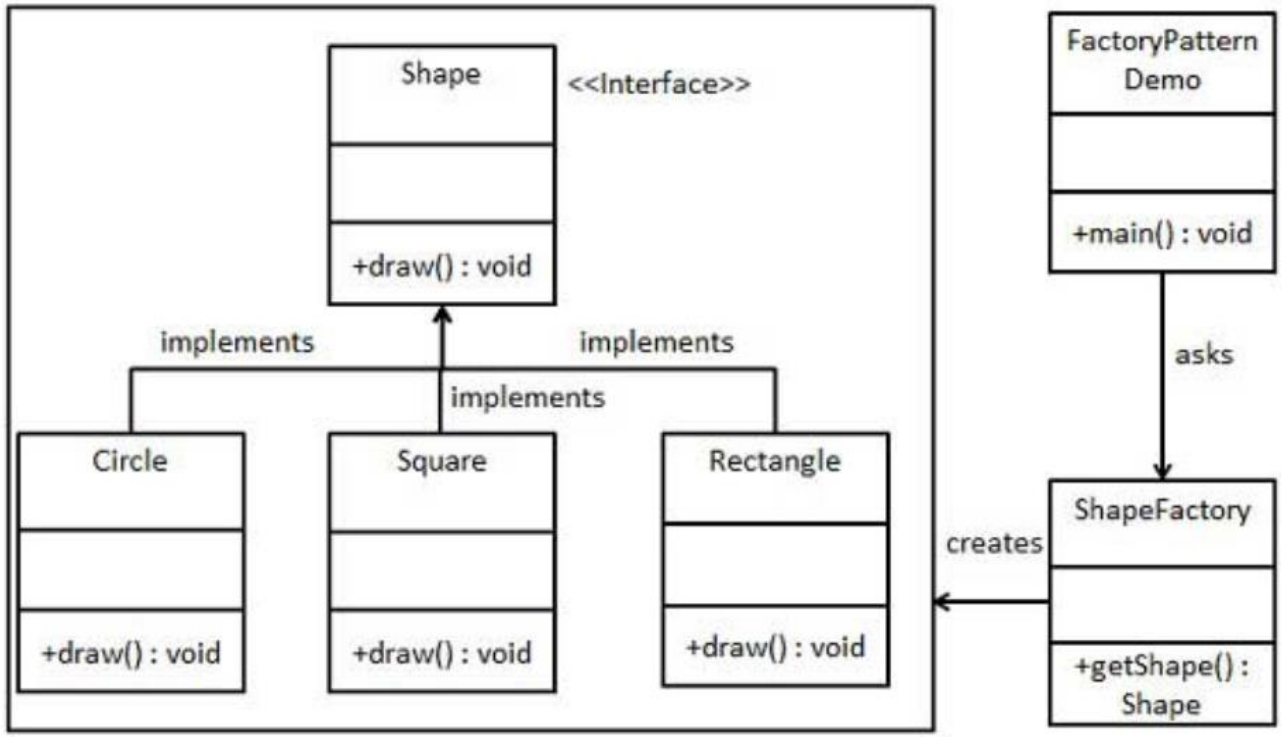
- Problem: Ich brauche eine, und nur eine, Instanz einer Klasse!



UML von: https://www.tutorialspoint.com/design_pattern/

Factory Pattern

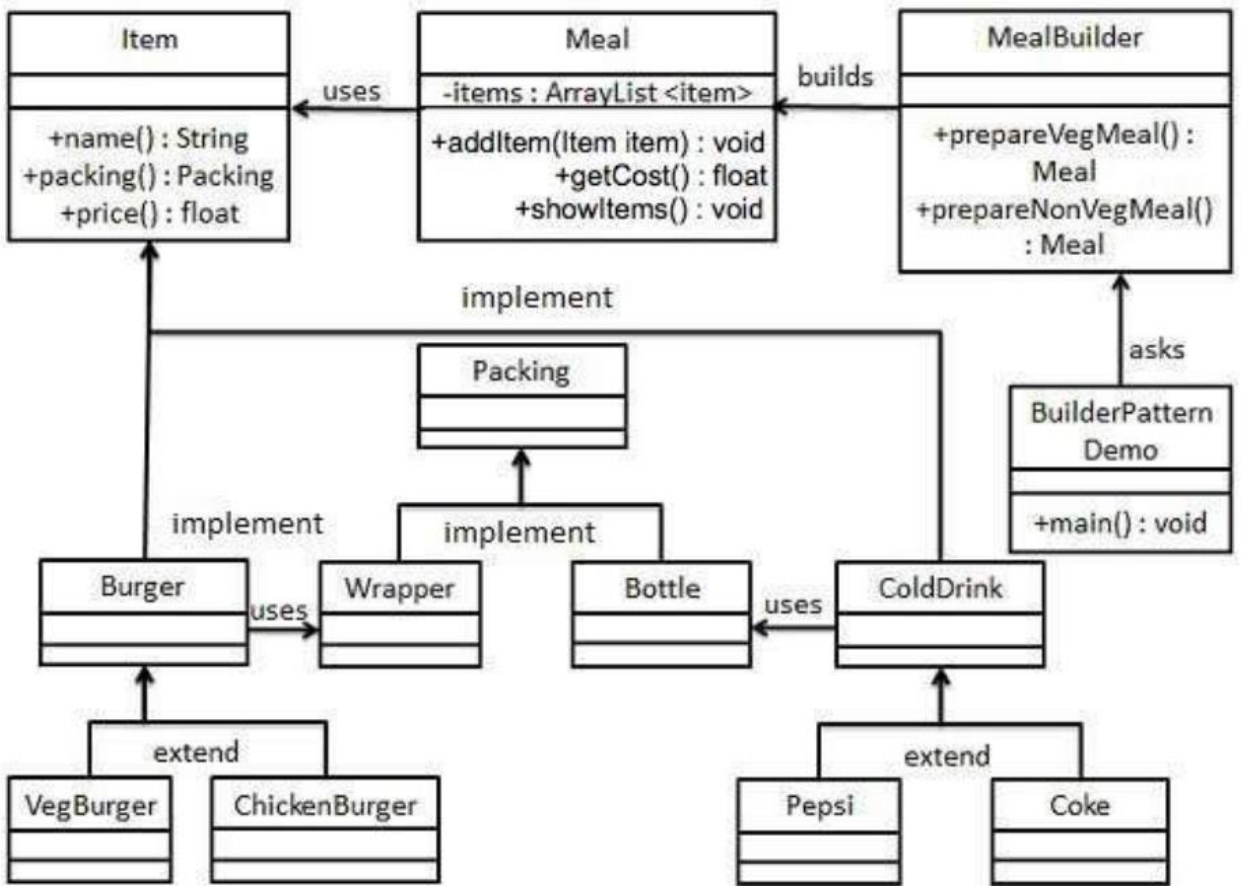
- Problem: Erzeuger braucht Klassenname, um Objekte zu erzeugen!



UML von: https://www.tutorialspoint.com/design_pattern/

Builder Pattern

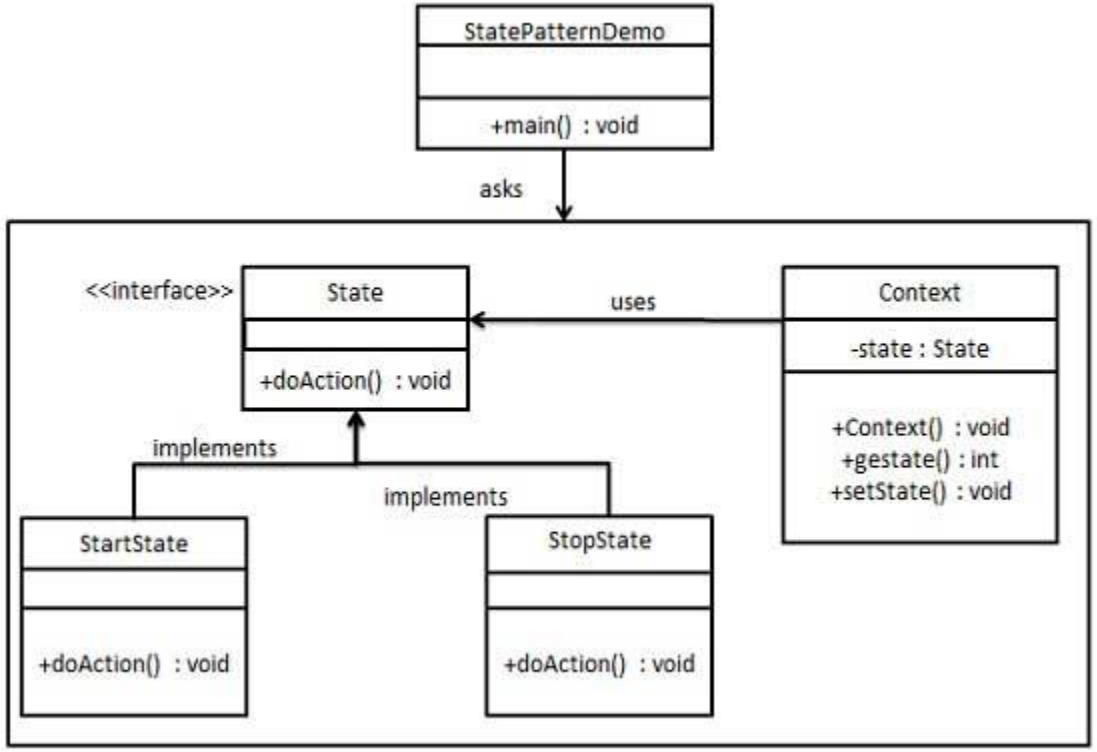
- Problem: Wir wollen einen komplexen Erzeugungsprozess über Subklassen hinweg teilen!



UML von: https://www.tutorialspoint.com/design_pattern/

State Pattern

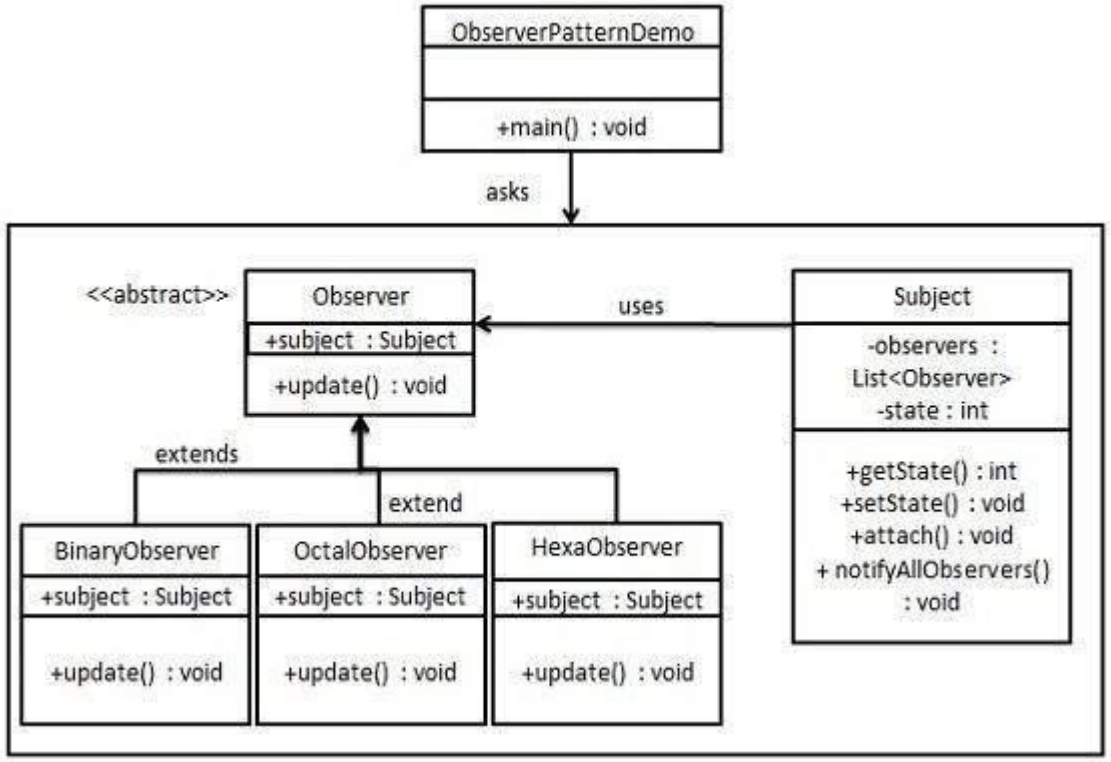
- Problem: Ändere Objektverhalten basierend auf seinem internen State!



UML von: https://www.tutorialspoint.com/design_pattern/

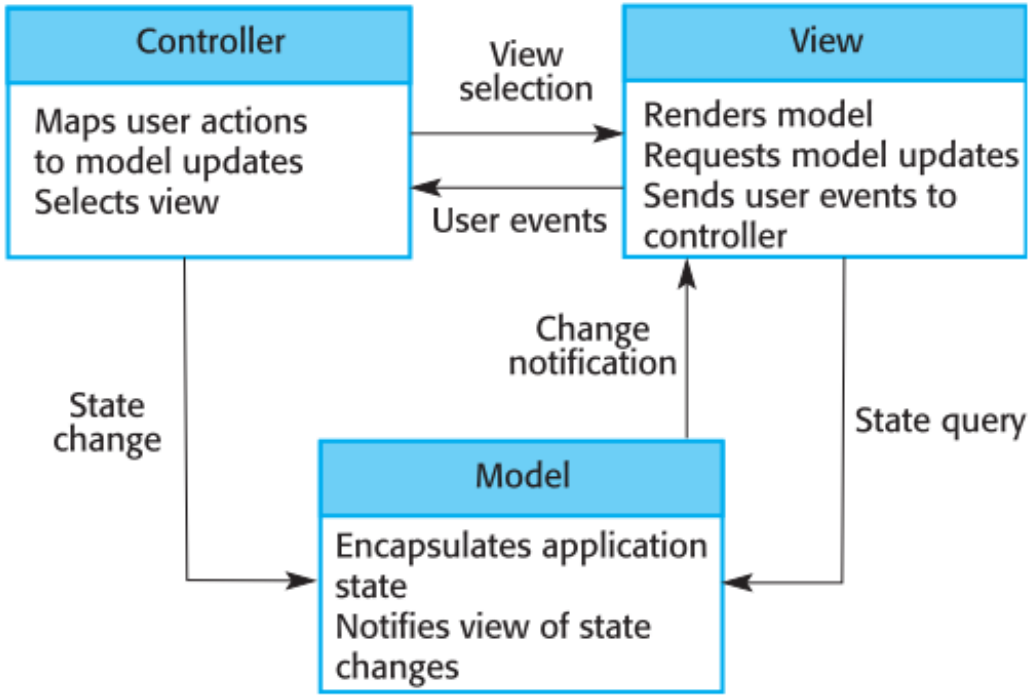
Observer Pattern

- Problem: Wir wollen über die Änderungen eines Objekts informiert werden!



Model-View-Controller (MVC) Architecture

- Beliebt in Web-Anwendungen
- „Separation of Concerns“
  - Modularität
  - Module haben alle Informationen, die sie brauchen, aber nicht mehr
- Frameworks in allen möglichen Sprachen
  - (ASP.NET Core MVC für C#)



Model-View-Controller (MVC) Architecture

Vorteile

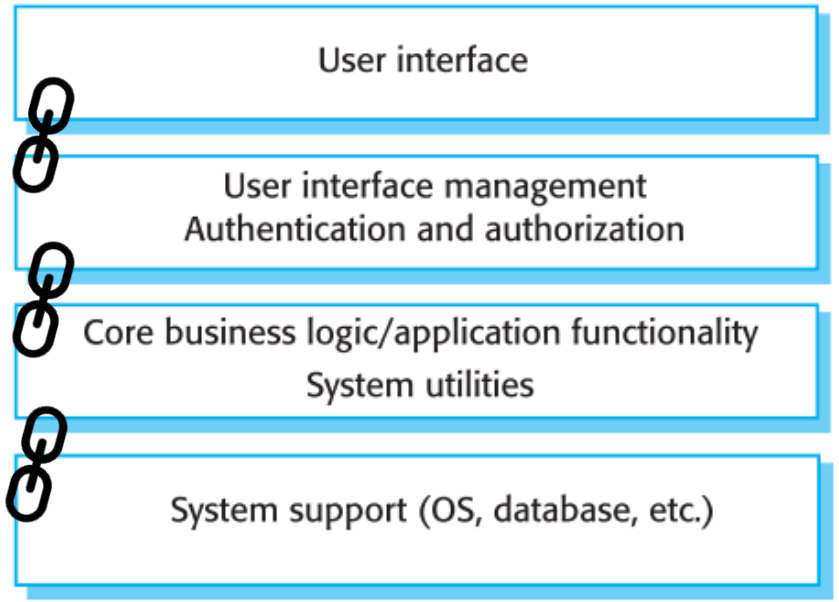
- Daten und Darstellung der Daten sind getrennt
  - => Datenmodell frei wählbar
  - => Darstellung der Daten frei wählbar
  - => Interaktionsmodalität frei wählbar
  - => Leicht erweiterbar

Nachteile

- Overhead durch Kommunikation zwischen den Komponenten
  - => Nur bei Projekten mittlerer & hoher Komplexität

Layered Architecture

- Jeder Layer ist abhängig vom Layer davor
- Jeder Layer darf nur auf die Daten des Layers davor zugreifen
- Datenaustausch über definierte Interfaces



Layered Architecture

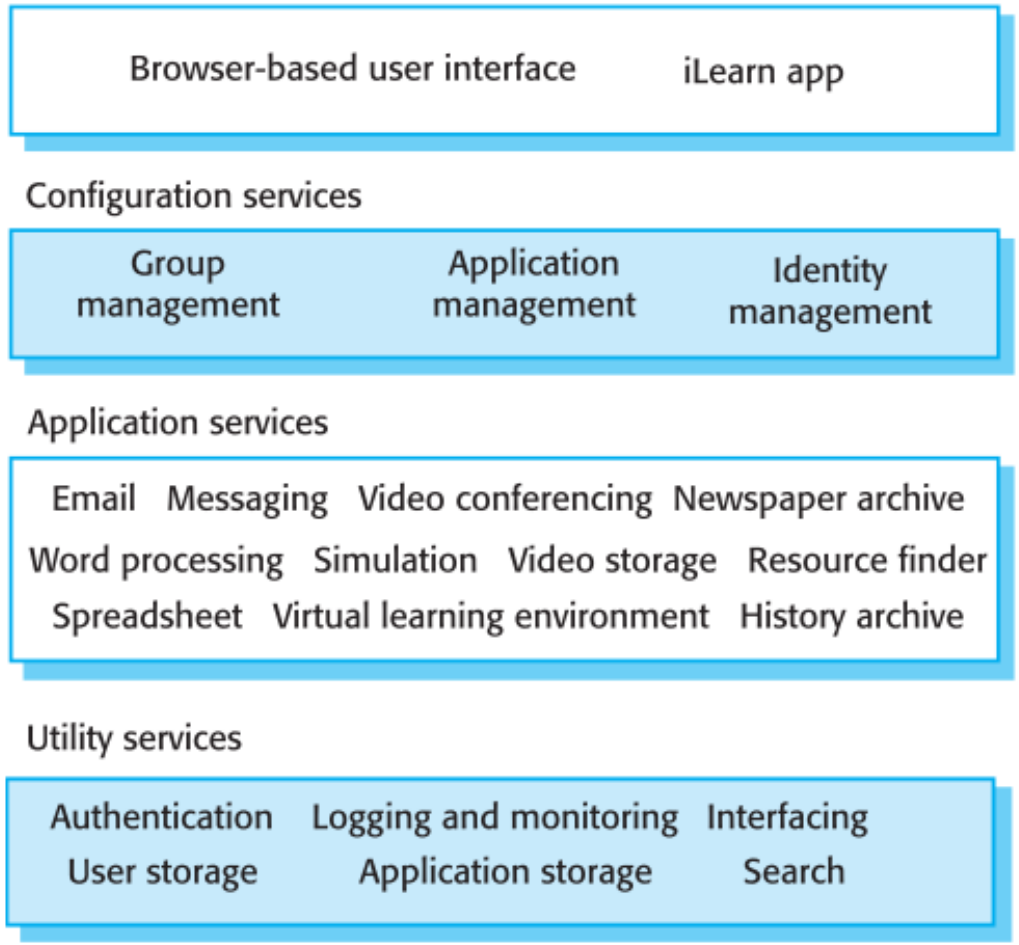
Vorteile

- Kommunikation über Interfaces
  => Layer können ausgetauscht / hinzugefügt werden
- Layer bilden eigene Einheit
  => Multi-Layer Security
  => Aufteilen von Entwicklungsaufgaben

Nachteile

- Interfaces müssen klar definiert sein
  => Oft praktisch nicht sauber umsetzbar
- Daten müssen auf jedem Layer wiederinterpretiert werden
  => Performanceverlust

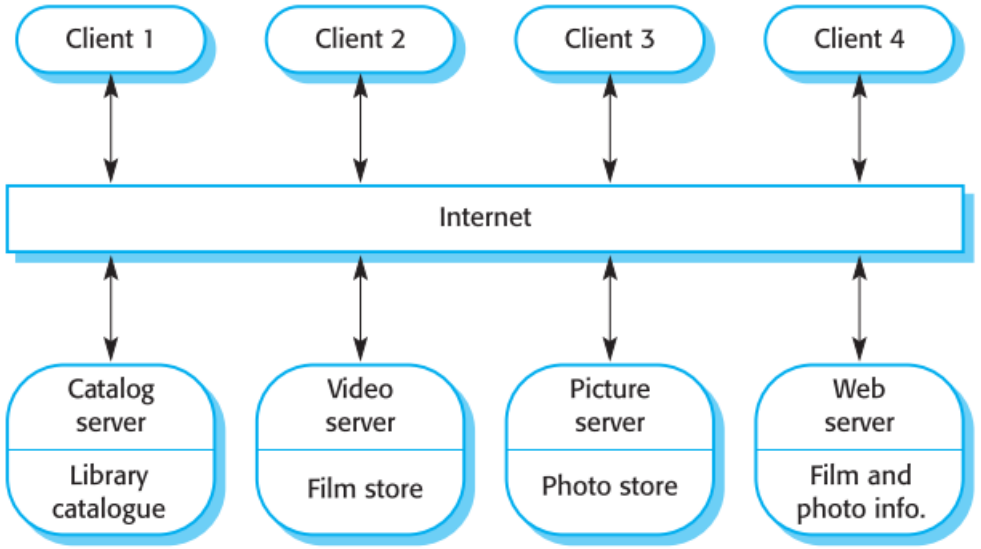
Layer Architecture (Beispiel)



Client-Server Architektur

- Menge an Servern, die Services anbieten
- Menge an Clients, die Services konsumieren
- Ein Netzwerk verbindet Clients und Server

- Kann auch auf einzelnem Computer laufen!
  (z.B. Multi-Threaded Software)



Client-Server Architektur

Vorteile

- Verteilbar auf mehrere Maschinen

  => Aufteilen der Workload („Load-balancing“)

  => Client muss nur Interface kennen, Logik auf Serverseite

Nachteile

- Netzwerk

  => Performance abhängig von Netzwerkperformance („Bottleneck“)

- Concurrency-Control

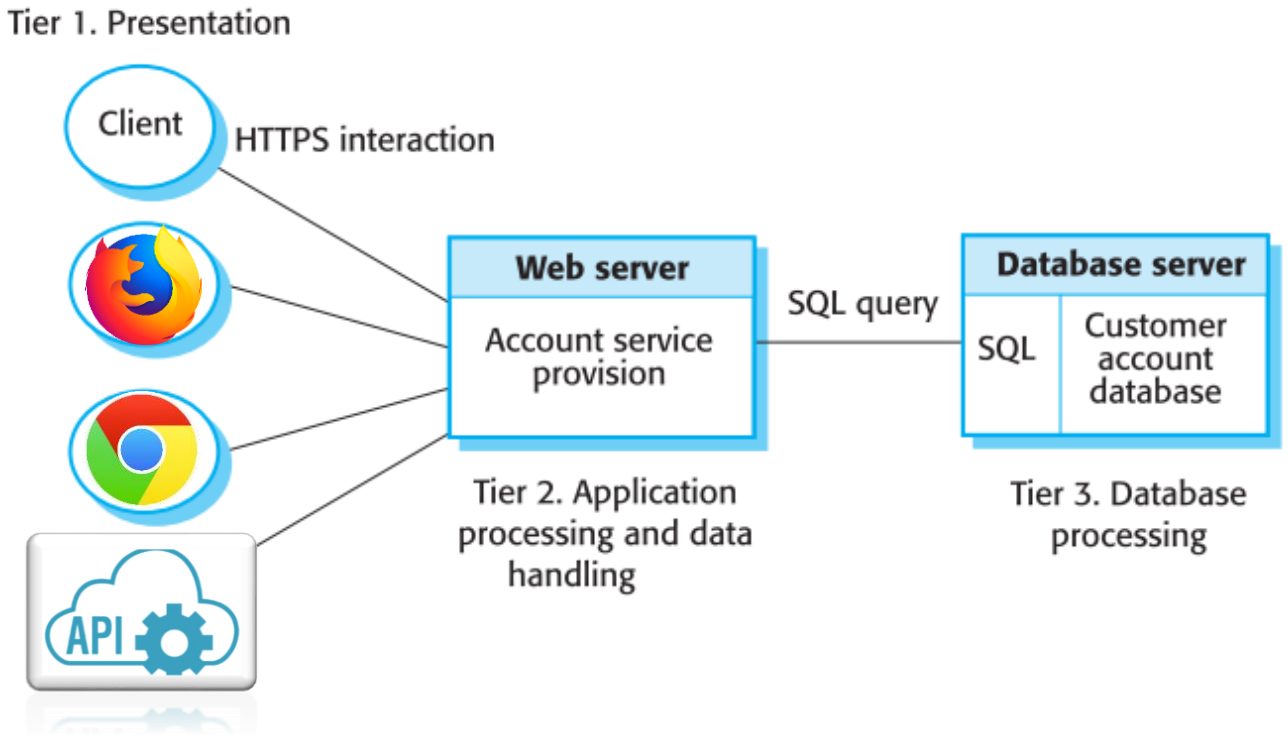
  => Wie behandle ich Konflikte?

- Kontrollverlust

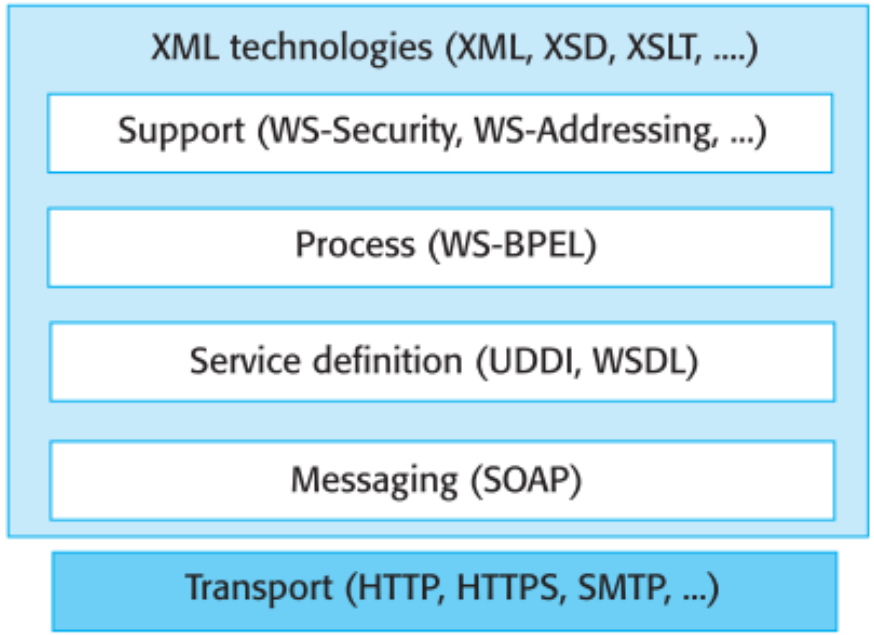
  => Weiß nicht, wie Anfrage verarbeitet wird

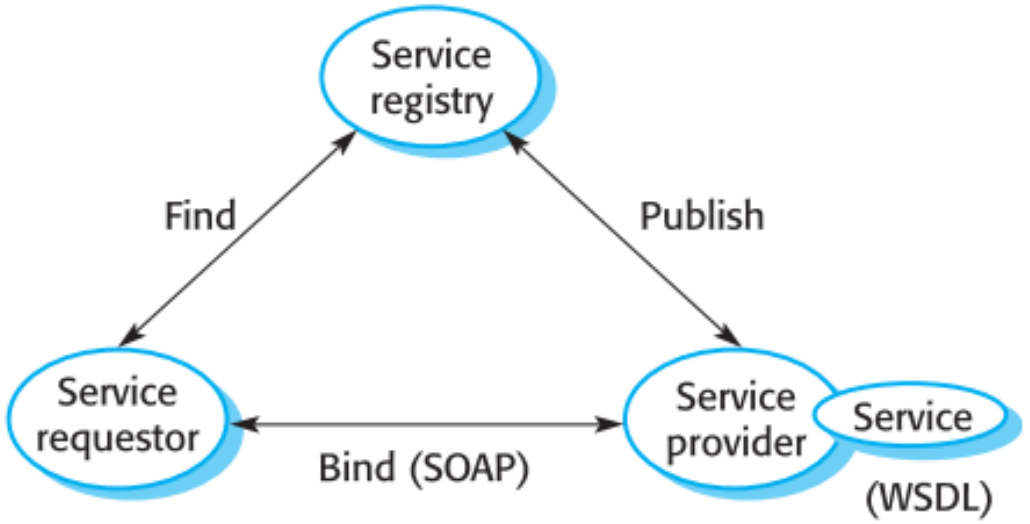
  => 3rd Party Services?

Beispiel Client-Server System



Service-Oriented Architecture (SOA)





Service-Oriented Architecture (SOA)

Vorteile

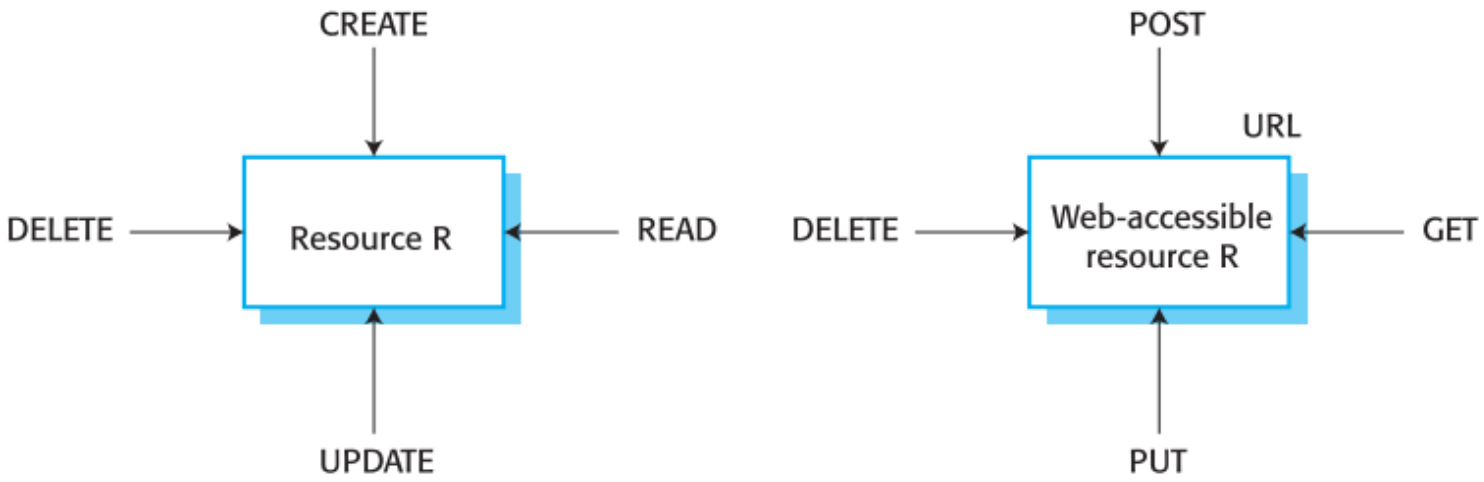
- Interfaces zu Services sind definiert und verfügbar
  => Service Requestor kann zwischen Kandidaten wählen
- Internationale Standards
  => Zukunftssicher
  => Über Organisationen hinweg

Nachteile

- WSDL beschreibt nur Interface
  => Semantik dahinter muss dokumentiert sein
  => Performance des Service ist nicht sicher
- SOAP ist XML basiert
  => ausgetauschte Nachrichten sind „riesig“

RESTful Services

- Representational State Transfer (Fielding 2000)



RESTful Services

Vorteile

- Message Format frei wählbar
  - => SOAP kann um REST erweitert werden (z.B. Cloud Provider)
- Kleine Messages

  => Format / Inhalt der Nachrichten frei wählbar (z.B. JSON)

  => Leichteres parsing, gut für Mobile

Nachteile

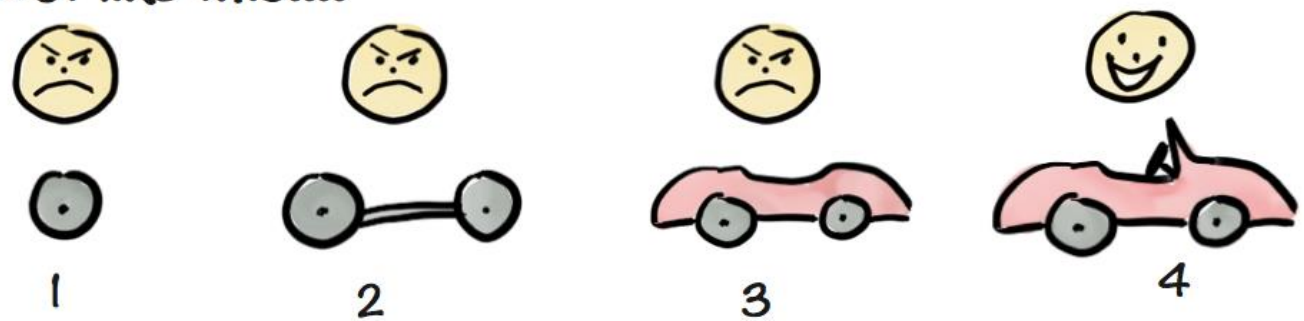
- Komplexe Service Interfaces können schwer umsetzbar sein
- Es gibt keinerlei Standards

REST API Demo

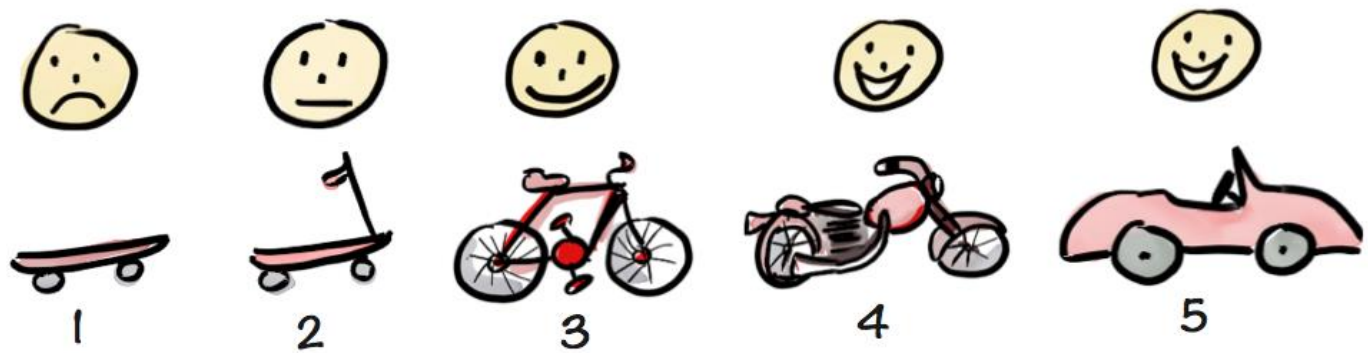
- https://jsonplaceholder.typicode.com/
- Firefox Plugin RESTClient

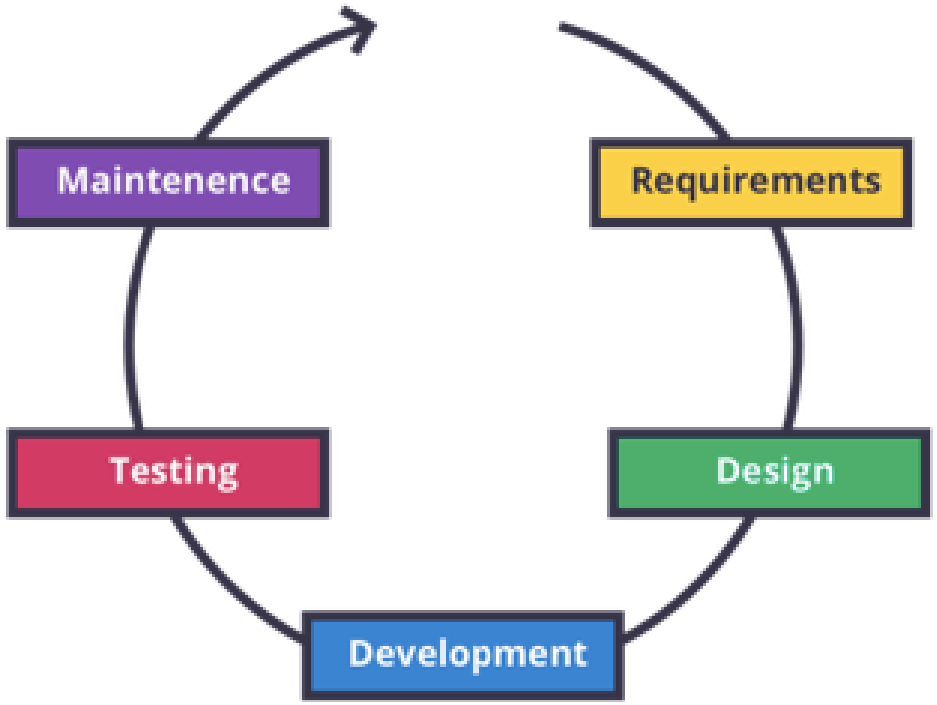
Agile Softwareentwicklung

Not like this....



Like this!



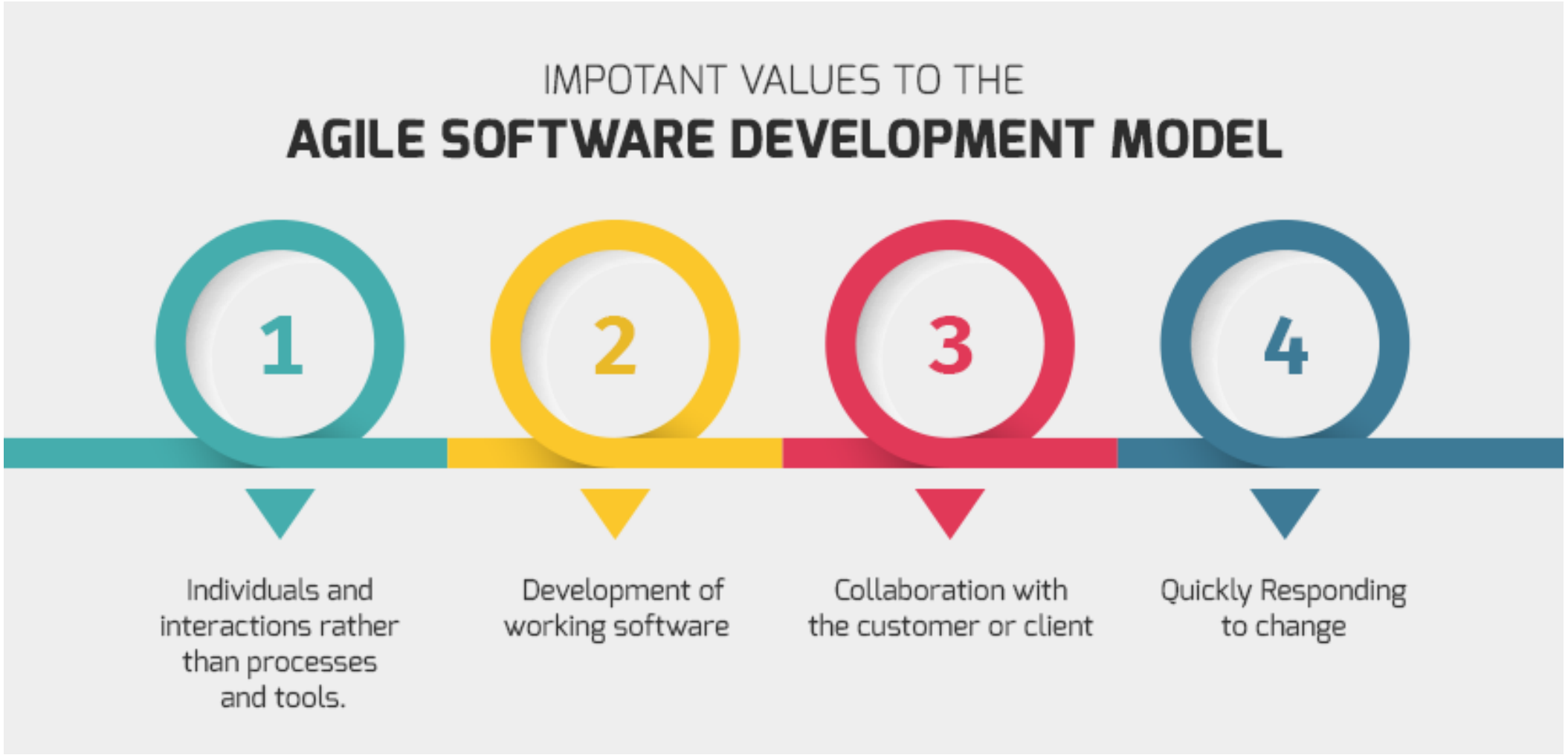


Agile Cycle

https://agilemanifesto.org/

12 Principles (extract)

- 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**.
- At regular intervals, **the team reflects** on how to become more effective, then tunes and adjusts its behavior accordingly


IMPOTANT VALUES TO THE
AGILE SOFTWARE DEVELOPMENT MODEL
1
2
3
4
Individuals and interactions rather than processes and tools.
Development of working software
Collaboration with the customer or client
Quickly Responding to change

Scrum

- Management Stil, der Agile implementiert


The Scrum Team
Product Owner
The person with the project vision, who breaks down the project into tasks and then prioritizes them, so everything meets the objectives and goals of the customer, market and organization.
ScrumMaster
This is the expert who helps to make sure the project is following the Scrum framework. While they guide and coach the team, their responsibility is to the process rather than the team.
Development Team
These are the people who execute the project. They're autonomous, self-directed and each has a unique set of skills that address the needs and the project and will work together with the whole team.
PROJECTMANAGER

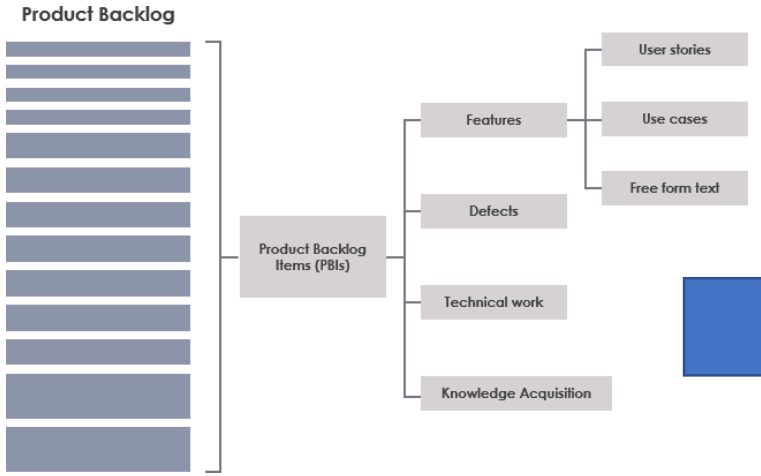
Die Rollen in Scrum

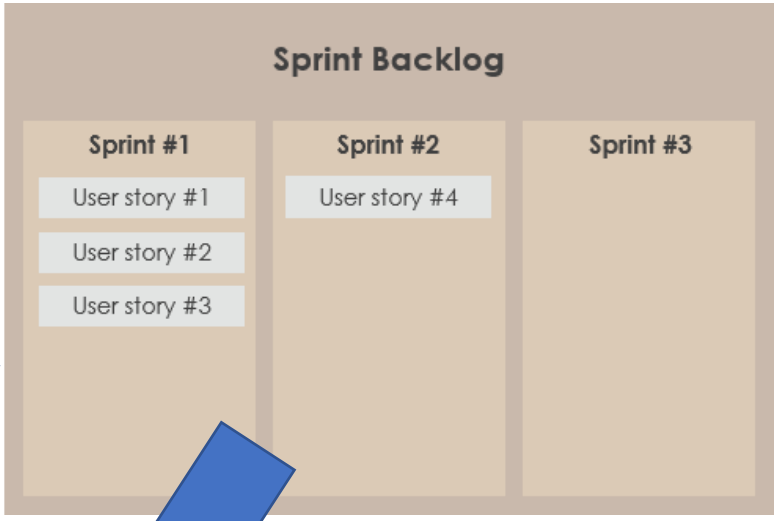
- Product Owner:
  - 1 Person, 100%
  - Business Value

- Development Team
  - Selbstorganisiert, interdisziplinär, 3-9 Personen
  - Fertigstellung von Increments
  - Alle Mitglieder 100%

- Scrum Master
  - 1 Person, 100%
  - Coaching und Einhaltung der Scrum Regeln

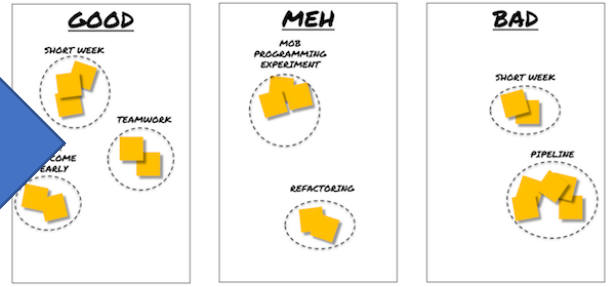
Artifacts



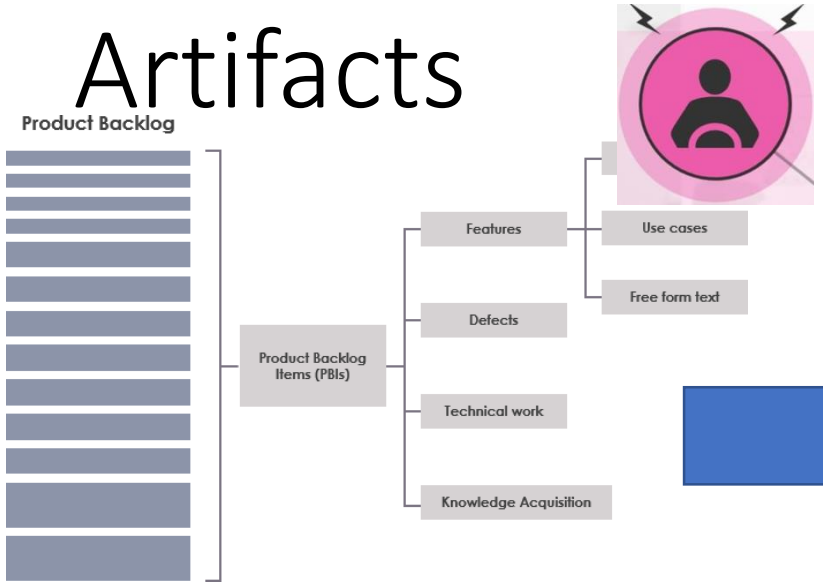


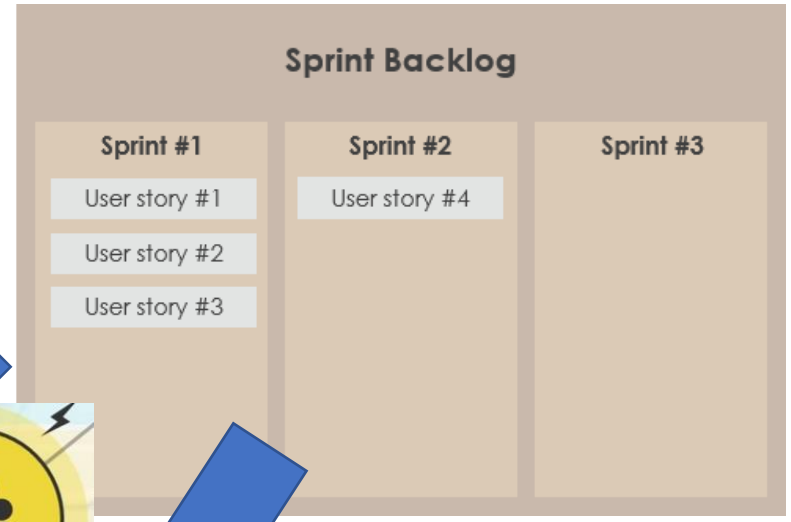
Increment

Retrospective



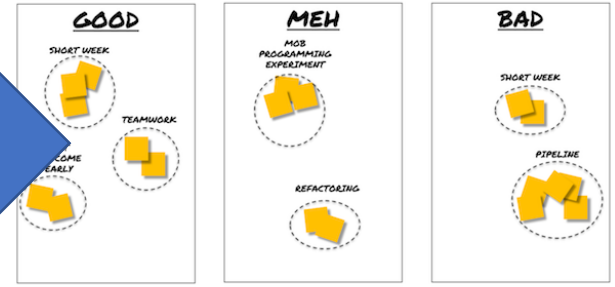
Artifacts



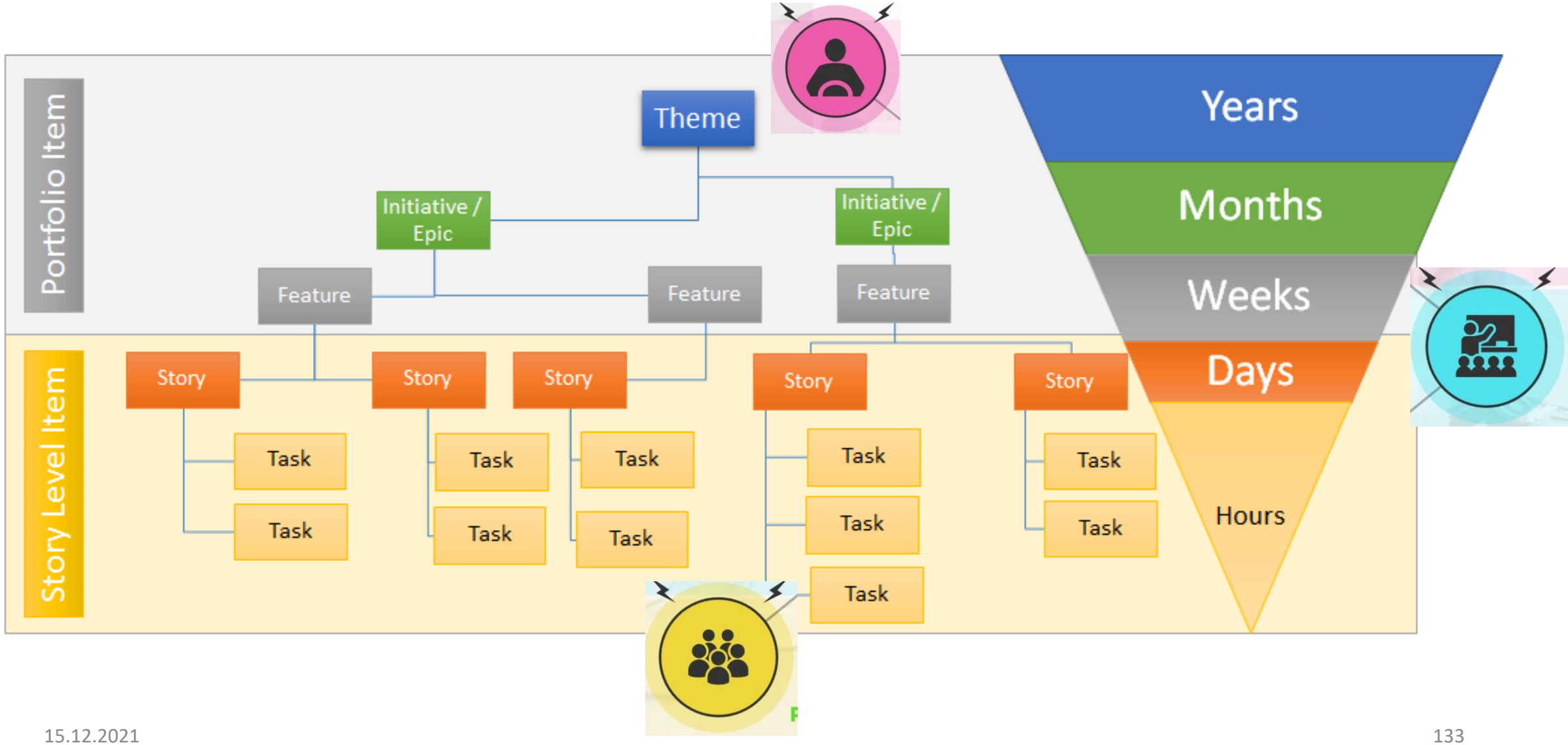


Increment

Retrospective



From Theme to Task



Userstories

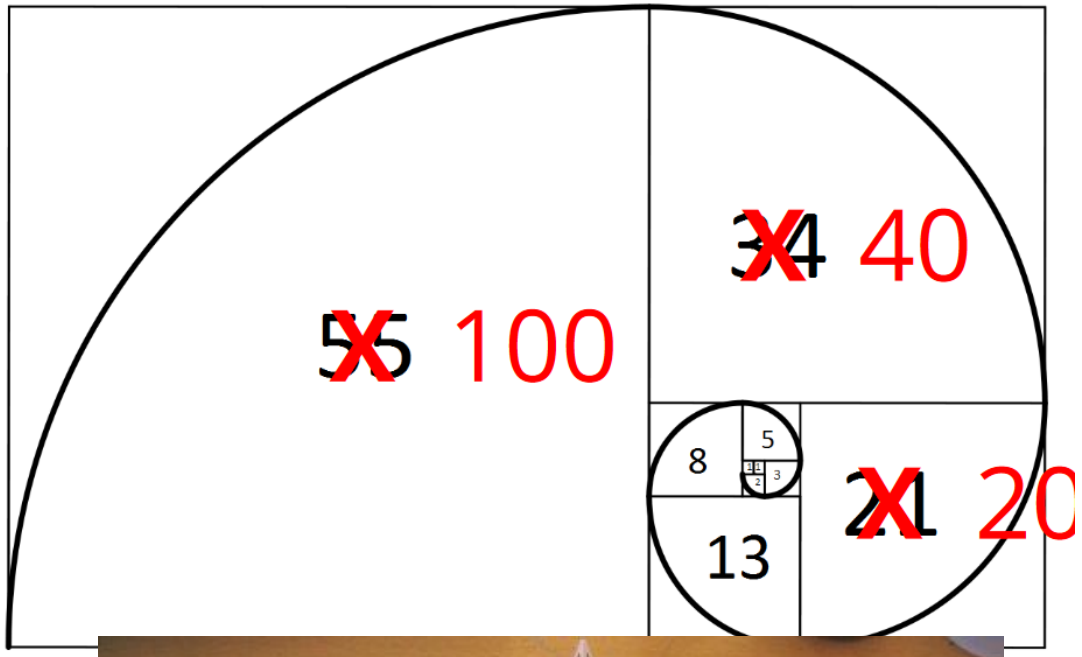
- Requirements werden als Userstories behandelt
- Vorlage:

„As **<role>**, I want **<function>** so that **<reason>**“

- Akzeptanzkriterien auf der Rückseite

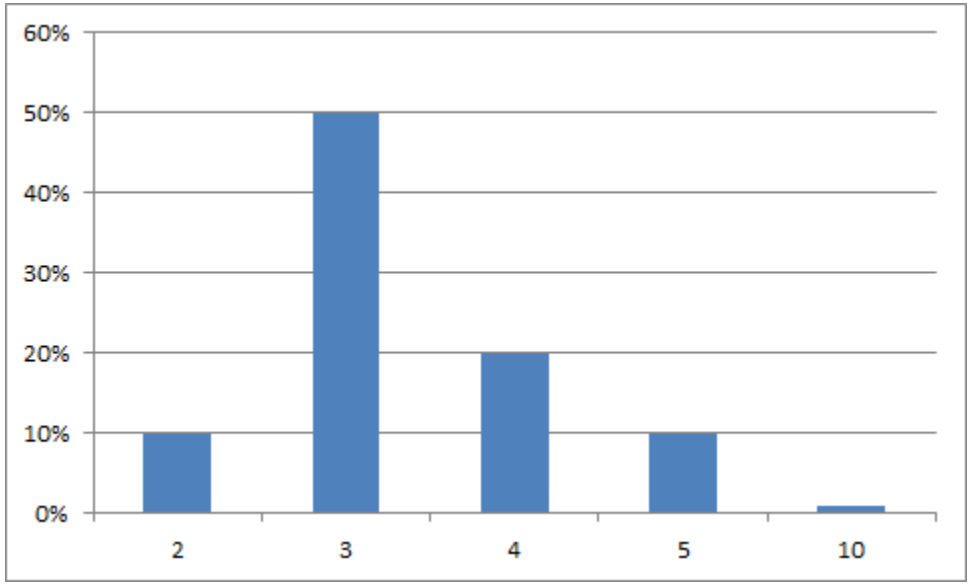
Abschätzung Aufwand

- Keine Zeit- oder Geld-Dimensionen!
- Immer relativ zu anderen Stories
  - T-Shirt Estimate: S, M, L, XL, XXL
- Beliebt: (Pseudo-) Fibonacci

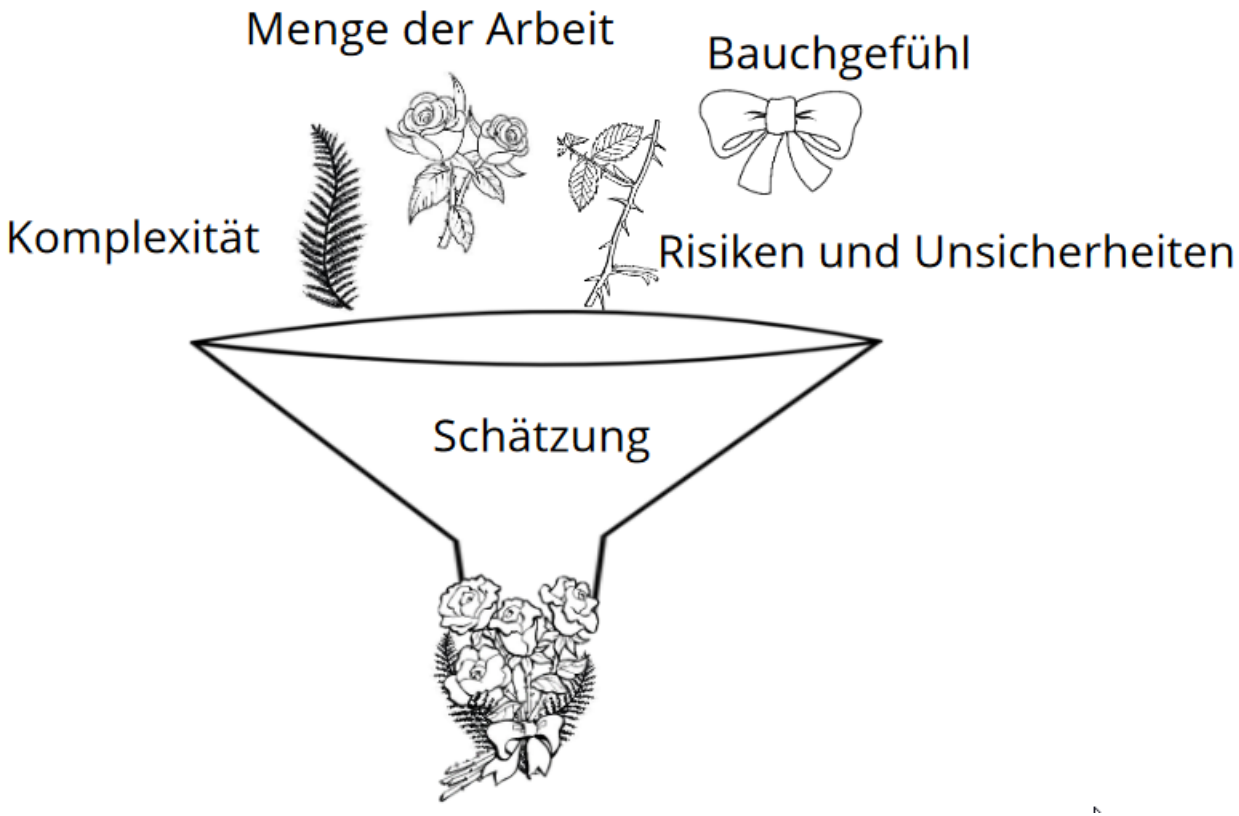


3-Zahlen Schätzung (aus „Clean Coder“)

- Optimistische Einschätzung
  - Alles geht nach Plan, nichts unerwartetes passiert
- Nominale Einschätzung
  - Üblicherweise dauert das X Stunden
- Pessimistische Einschätzung
  - Alles geht schief



Was ist alles Teil der Schätzung?



Scrum Events

- Der Sprint (eine komplette Iteration)
  - Dauer 2 Wochen – 1 Monat, aber immer gleich lang
  - Ziel: Ausliefern eines Potentially Shippable Increment, welches Sprintziel erfüllt
- Sprint Planning
  - Am Beginn des Sprints
  - Dauer: ~8h für 4-wöchigen Sprint
  - Ziel: Was soll im nächsten Inkrement drinnen sein? Wie kommen wir dahin?
- Daily Scrum
  - Tägliches Meeting
  - Dauer: 15 min, im Stehen!
  - Kurzer Bericht, was geschehen ist, was geschehen wird, Hindernisse
  - Keine Problemlösung!

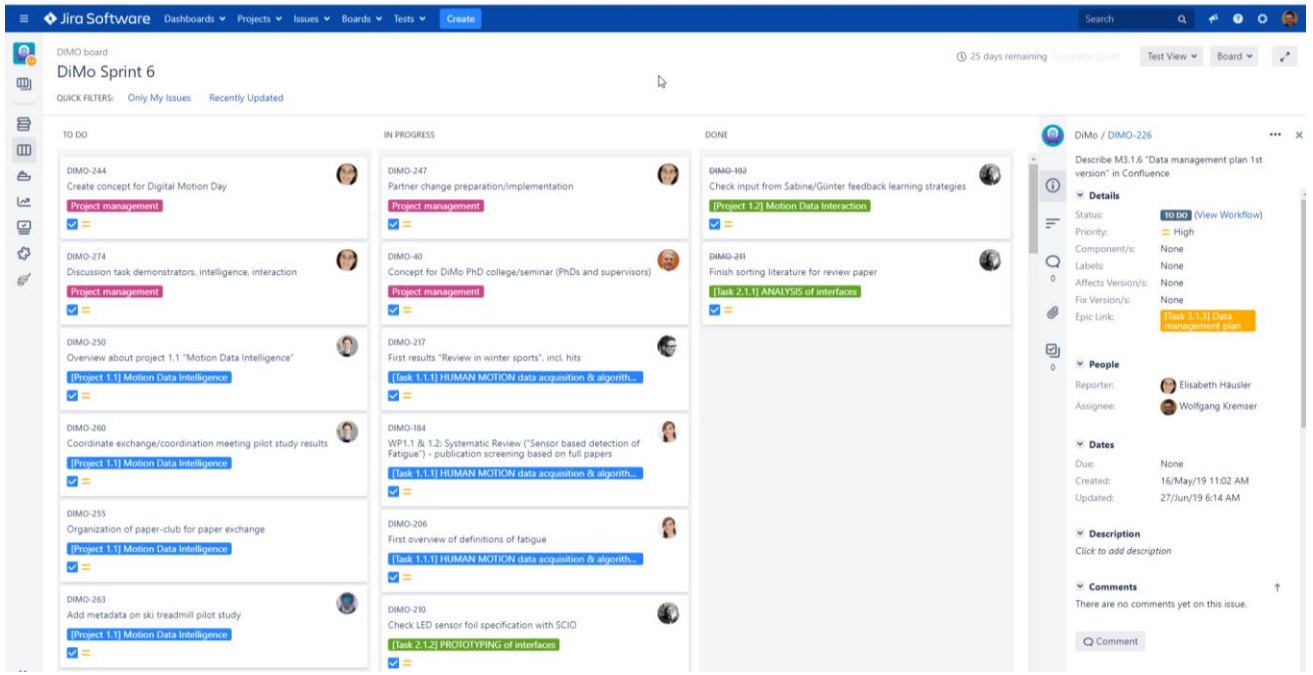
Scrum Events

- Review Meeting
  - Ende des Sprints
  - Dauer: 4h bei 1-monatigen Sprint
  - Ziel: Product Owner präsentiert Fortschritt + neuen Backlog
  - Dev Team präsentiert Inkrement, reüssiert Sprint

- Retrospective
  - Zwischen Sprints
  - Dauer: 3 Stunden für 1-monatigen Sprint
  - Ziel: Lernen aus Erfahrungen

Tools for Agile

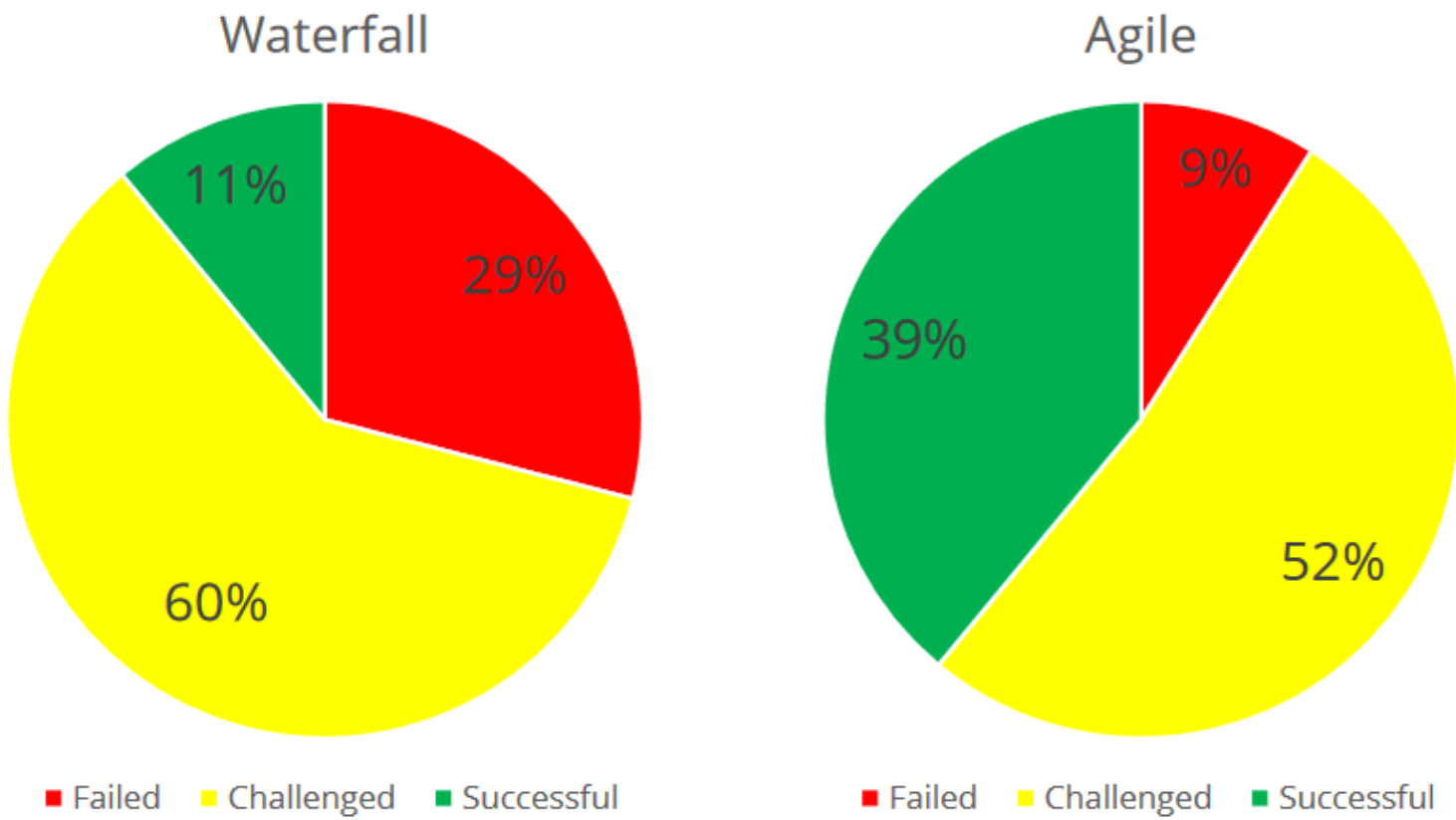
- Version Control (Git, SVN)
- Jira
- Confluence



Analoges Sprintboard

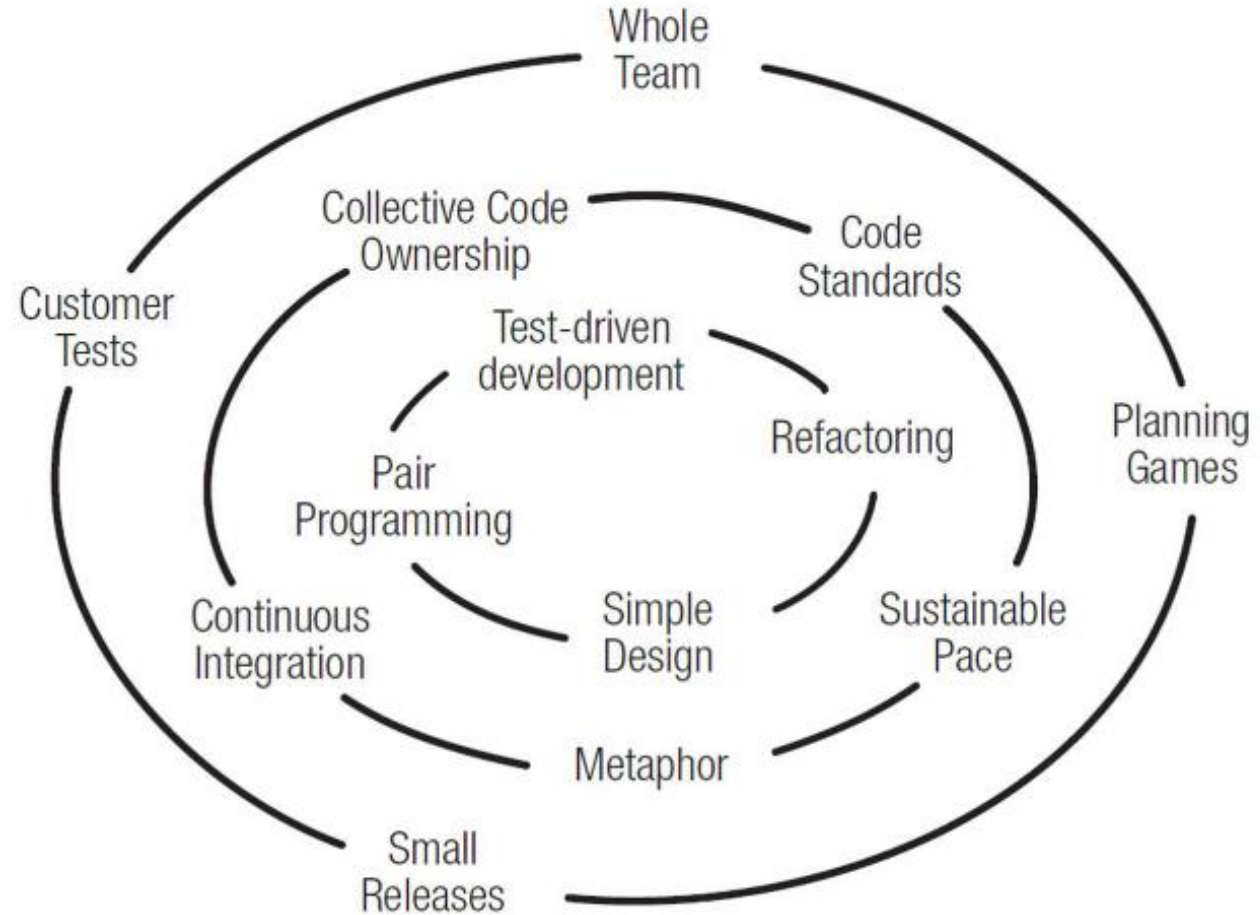
Why Agile Fails

*Respondents were able to make multiple selections.



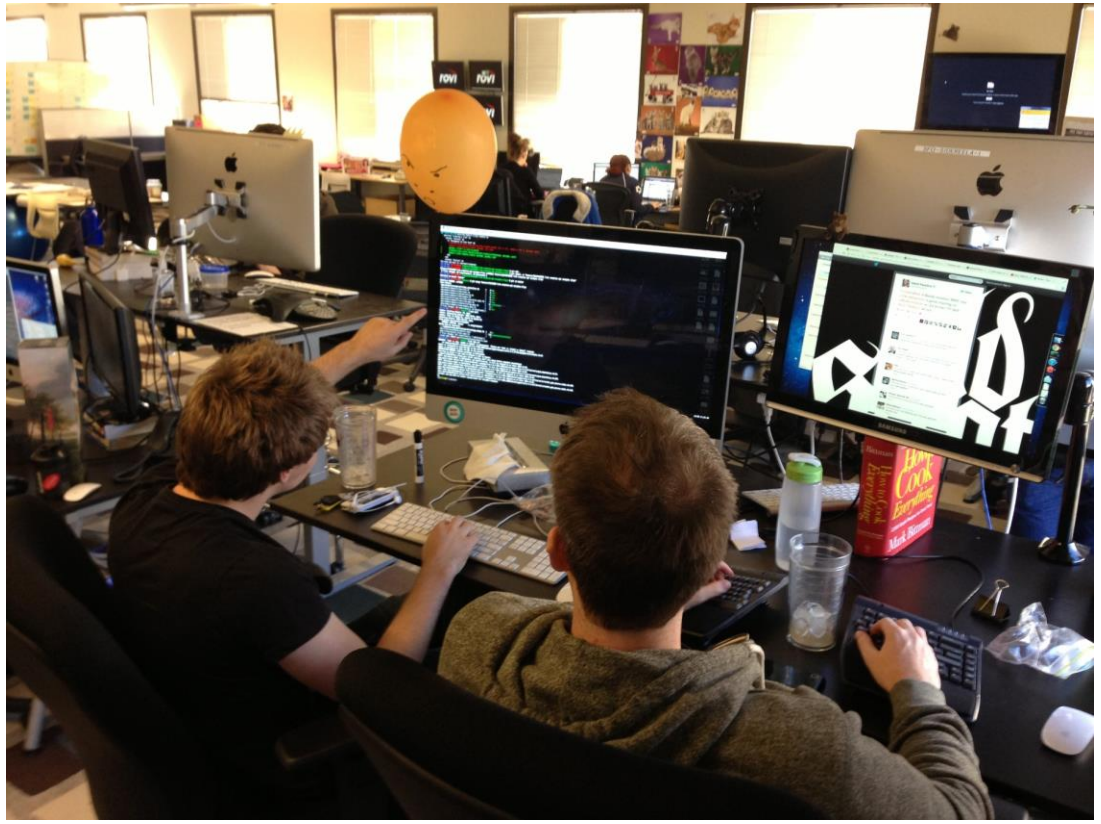
Aus: Standish Group – CHAOS Report 2011-2015 – >10.000 Software Projects

Xtreme Programming



XP - Pair Programming

- Driver – schreibt den Code
- Navigator – steuert die Entwicklung
- Regelmäßiger Wechsel
- Nie alleine



XP - Refactoring

- „Lasse den Code sauberer zurück als du ihn gefunden hast!“

- Ständige Anpassung von Architektur, Design und Code
- Achtung: Refactoring ändert nur die **Innenwahrnehmung** der Software, nach Außen hin bleibt sie gleich.

Software Validation

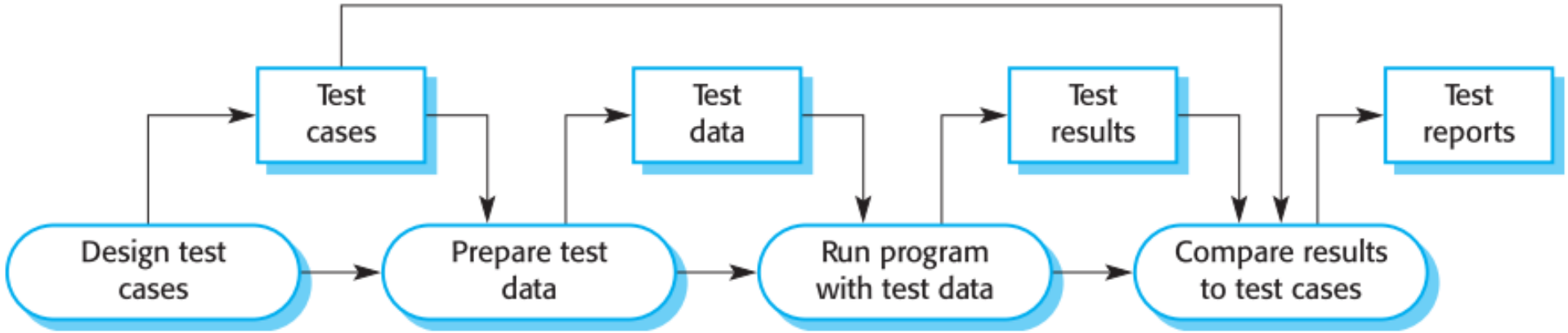
Zwei Ziele des Testing

- Validation Testing
  - Stakeholdern beweisen, dass Requirements erfüllt werden

- Defect Testing
  - Bugs finden und beheben

Bug = Software Output ist nicht korrekt, ungewünschtes verhalten, Spezifikationen werden nicht eingehalten.

Testaublauf



Development Testing

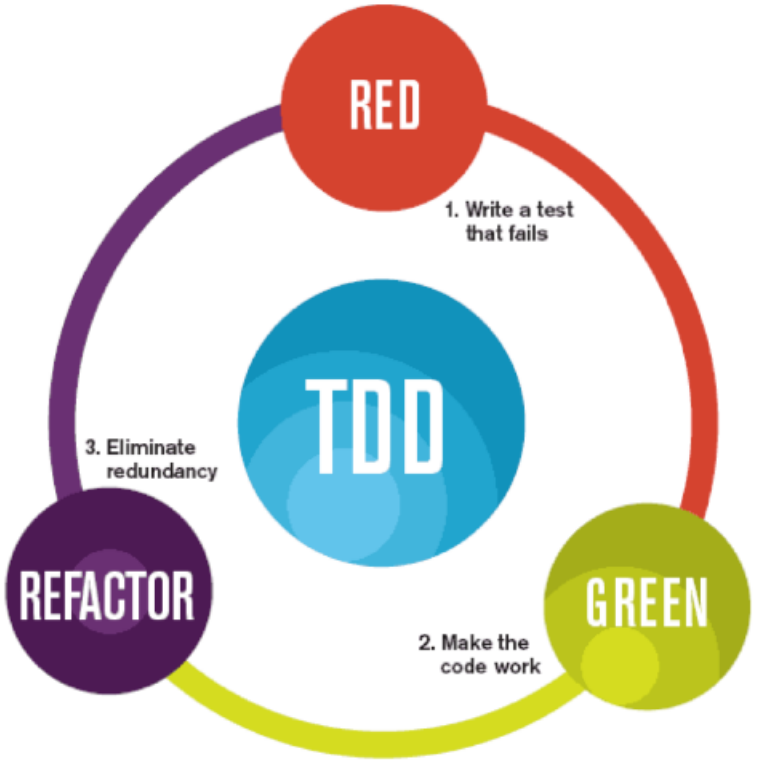
- Unit Tests
  - Einzelne Programmeinheiten oder Klassen werden getestet
- Component Tests
  - Teste, ob mehrere Programmeinheiten gut zusammenarbeiten
- System Tests
  - Das ganze System wird getestet. Interagieren die Komponenten wie gewollt?

Unit Tests

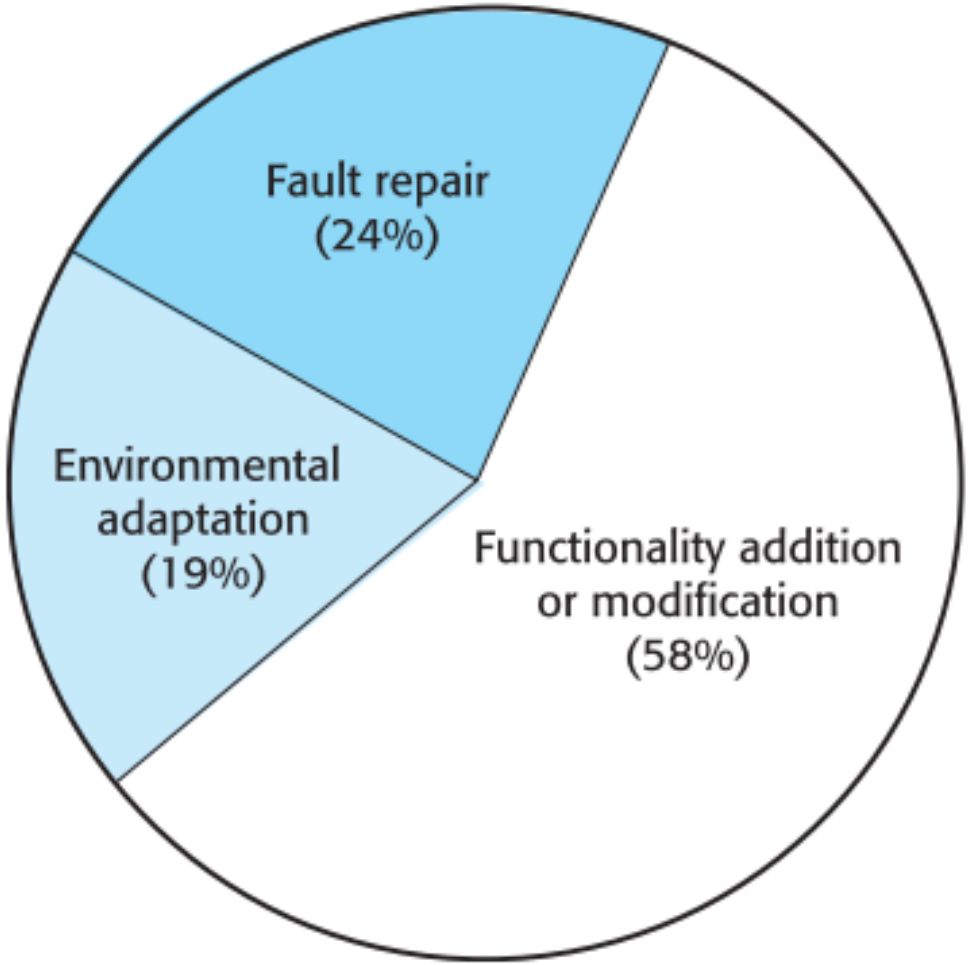
- Teste einzelne Methoden mit verschiedenen Input Parametern
- Versuche, Objekt in alle möglichen Zustände zu versetzen
  - Normaler Input
  - Randfälle
  - Falscher Input

1. Setup
2. Call
3. Assertion

Test Driven Development

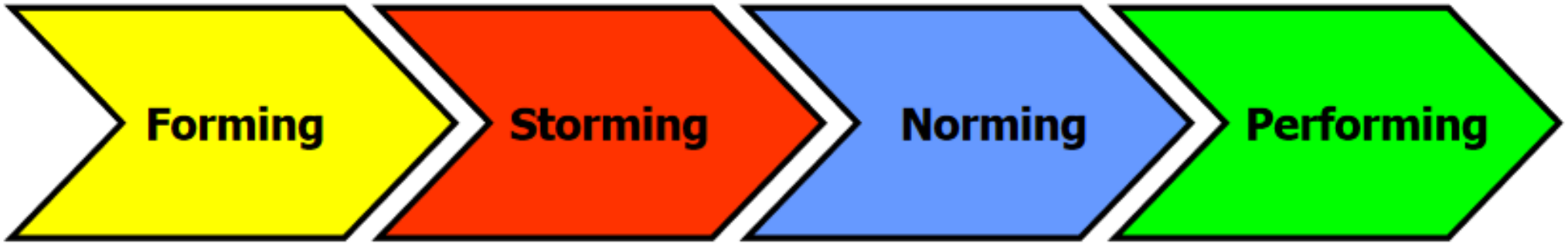


Software Evolution

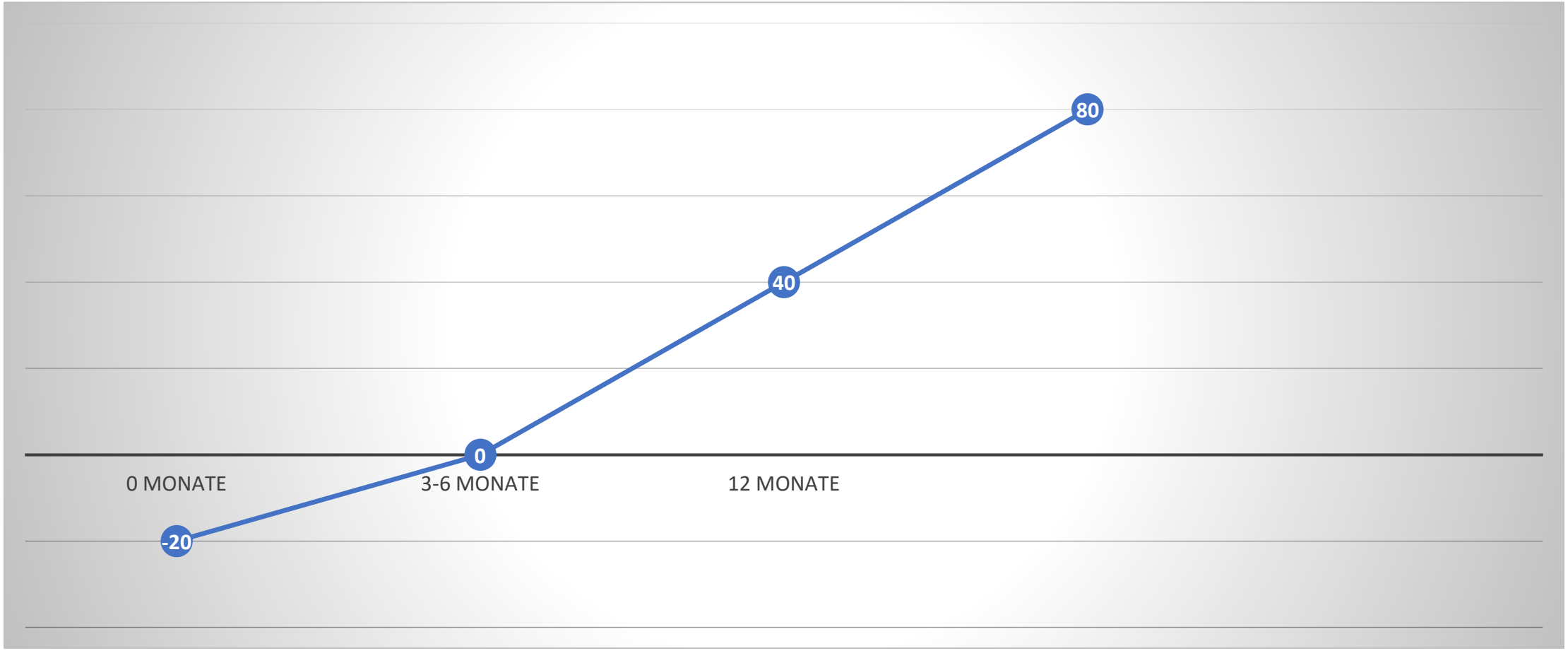


Teambildung

Jeder neue Mitarbeiter verursacht…



Jeder neue Mitarbeiter verursacht…



Brooks Law (1975)

„Adding manpower to a late software project makes it later“

Psychological Safety

- If I make a mistake in this team, it is held against me.
- Members of this team are able to bring up problems and tough issues.
- People on this team sometimes reject others for being different.
- It is safe to take a risk in this team.
- It is difficult to ask other members of this team for help.
- No one on this team would deliberately act in a way that undermines my efforts.
- Working with members of this team, my unique skills and talents are valued and utilized.