V05 - Entwurf

Softwaretechnik 1

Wintersemester 2021/2022

Prof. Dr.-Ing. Jasminka Matevska

(jasminka.matevska@hs-bremen.de)

Hochschule Bremen, Fakultät Elektrotechnik und Informatik, TI, ISTI, DSI & ATMEI

Rechtliche Hinweise

- Die Rechte an geschützten Marken liegen bei den jeweiligen Markeninhabern
- Die Rechte an referenzierte Literatur, Folien, Notizen und sonstigen Materialien liegen bei den jeweiligen Autoren
- Diese Veranstaltung benutzt Materialien von Prof. Dr. Spillner (HSB in Ruhestand) und Prof. Dr. Hasselbring (Universität Kiel)
- Alle Rechte an den Materialien zu dieser Veranstaltung liegen bei ihrem Autor, Prof. Dr.-Ing. Jasminka Matevska
- Jede Form der teilweisen oder vollständigen Weitergabe, Speicherung auf Servern oder Nutzung in Lehrveranstaltungen, die nicht von dem Autor selbst durchgeführt werden, erfordert seine schriftliche Zustimmung. Eine schriftliche Zustimmung ist darüber hinaus für jede kommerzielle Nutzung erforderlich
- Für inhaltliche Fehler kann keine Haftung übernommen werden

Der **rote** Faden

- Einführung
- Entwicklungsprozess
- Vorgehensmodelle
- Anforderungsanalyse
- **Entwurf**
- Objektorientierter Entwurf mit UML
- Qualitätssicherung
- Test
- Konfigurationsmanagement
- Implementierung
- Projektmanagement

Frage

- Was ist nach der Erstellung des Pflichtenheftes zu tun?
- Was meinen Sie?

Entwurf/Design - Gliederung

- Allgemeines
- Entwurfskriterien
- Entwurfsprinzipien
- Sichtweisen / Vorgehensweisen zum Entwurf
- Heuristiken

Entwurf

- Festlegung der Architektur des Systems
- Spezifikation der Komponenten des Systems und deren Schnittstellen

  → aus dem **WAS** das System leisten soll, nach und nach das **WIE** erarbeiten
- Kreativer Prozess, der erfordert
  - umfassendes Verständnis der Problemstellung
  - Fähigkeit der Umsetzung von softwaretechnischen Methoden und Prinzipien
  - Fantasie und Kreativität
- **Interaktiver** und **iterativer** Prozess

  → vom ersten Entwurf bis zur Feinstrukturierung des Systems

Entwurf

- Qualität des Systems stark von der Qualität des Entwurfs abhängig
- Guter Entwurf?
  - lässt sich nicht definieren
  - abhängig vom Anwendungsbereich
  - gute Architektur kann vorliegen bei
    - kompaktem System, wenig Ressourcen oder
    - einfach zu verstehen, leicht erweiterbar
- Wichtige Kriterien
  - Wartbarkeit und Erweiterbarkeit
  - Ressourcenverbrauch
  - Angemessenheit
  - Korrektheit
  - ...

Entwurf - Tätigkeiten

1. Problemadäquate Zerlegung des Gesamtsystems in Teilsysteme und Spezifikation der Wechselwirkungen zwischen den Teilsystemen
2. Zerlegung der Teilsysteme in Komponenten (Module) und Spezifikation der Anforderungen an diese Komponenten, d.h. Festlegung der Schnittstellen der Komponenten
3. Entwurf der Algorithmen zur Bereitstellung der durch die Komponenten-schnittstelle definierten Funktionalität

Entwurf - Wissensbasis

- Anforderungsdefinition / Pflichtenheft
- Zusätzliche vom Kunden zur Verfügung gestellte Dokumente
- Allgemeine und spezielle Literatur zum Problembereich
- Vorwissen der Analytiker
- Wissen und Erfahrungen von Fachleuten
- Wissen und Erfahrungen von Personen, die in dem Bereich arbeiten
- Erfahrungen mit früheren / anderen Systemen

Entwurfskriterien und -prinzipien

- Kriterien
  - Kopplung
  - Kohäsion
  - Zulänglichkeit
  - Vollständigkeit
  - Einfachheit
- Prinzipien
  - Abstraktion
  - Strukturierung
  - Modularisierung
  - Hierarchie
  - Kapselung
  - Separation of Concerns
  - Gründlichkeit und Formalität

Was ist ein Entwurfskriterium?

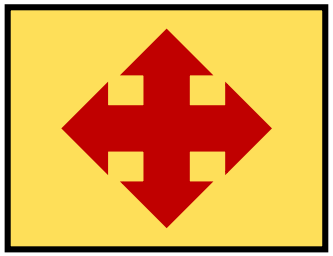
- Maß für die interne Beschaffenheit eines Entwurfs

Entwurfskriterium Kopplung

- Maß für die Abhängigkeit der Bausteine untereinander
- Kopplung über die Schnittstelle
- **Ziel:** geringe/lose Kopplung
  - Änderungen und Austausch eines Bausteins leicht möglich
  - Fehler in einem Baustein wirkt sich nicht auf andere aus

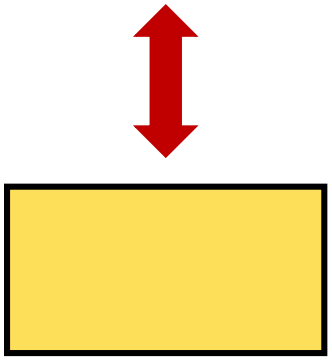
Entwurfskriterium Kohäsion

- Maß für die (funktionale) Bindung der Elemente innerhalb eines Bausteins
- Kohäsion bewertet den inneren Zusammenhalt
- **Ziel**: starke/hohe Kohäsion
  - alle benötigten Informationen und Operationen zur Lösung einer (Teil-) Aufgabe sind in einem Baustein vorhanden (aber auch nicht mehr!)



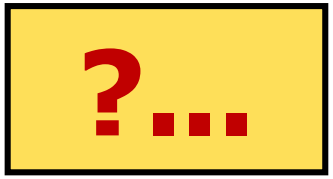
Entwurfskriterium Zulänglichkeit

- Maß für Erfüllung der Anforderungen
- Verantwortlichkeit muss über die Schnittstelle eines Bausteins in minimaler Form erreichbar sein
- Zulänglichkeit bewertet die "passende" Umsetzung der Anforderungen an einen Baustein
- **Ziel**: überschaubare, einsichtige Zulänglichkeit



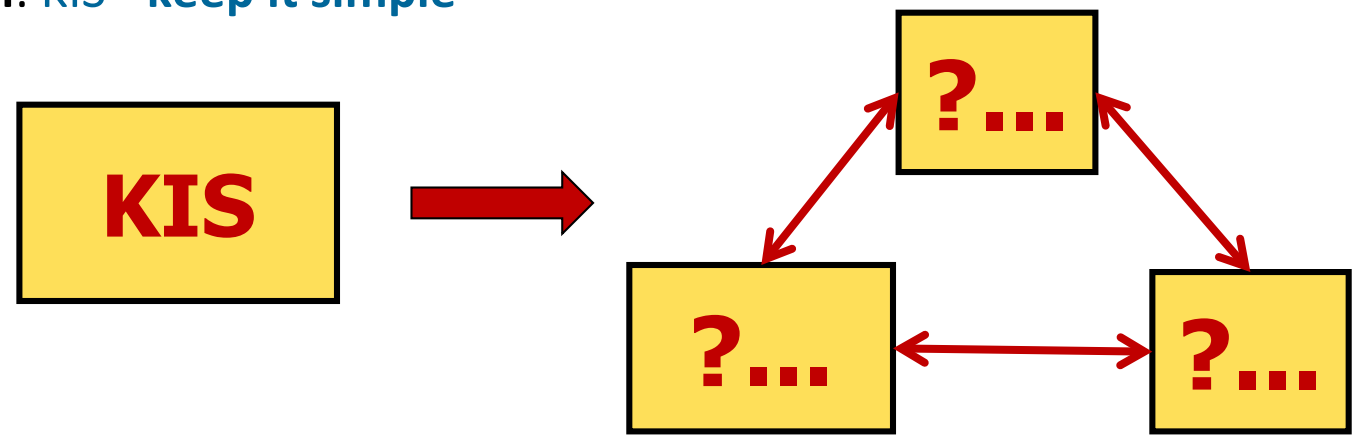
Entwurfskriterium Vollständigkeit

- Maß für vollständige Umsetzung der Anforderungen
- Verantwortungsbereich eines Bausteins soll die geforderte Funktionalität umfassend anbieten
- Vollständigkeit bewertet die "komplette" Umsetzung der Anforderungen an einen Baustein
- **Ziel**: Vollständigkeit in Bezug zur Aufgabe



Entwurfskriterium Einfachheit

- Maß für die Umsetzung von hinreichender Funktionalität mit angemessener Schnittstelle des Bausteins
- Gradwanderung zwischen Zulänglichkeit und Vollständigkeit
- Einfachheit durch das "Runterbrechen" komplexer Funktionalität in einzelne, einfache Funktionalitäten
- **Ziel**: KIS - **keep it simple**



Was ist ein Entwurfsprinzip?

- Grundsatz, dem man seinem Handeln zugrunde legt
- Noch unabhängig von konkreten Methoden und Techniken

Entwurfsprinzip Abstraktion (Verallgemeinerung)

- Behalten der Übersicht durch Ignorieren von (zunächst unwichtigen) Details
  - Zerlegung in Hinblick auf die Identifizierung wichtiger und (zunächst) weniger wichtiger Aspekte
  - Wertung (wichtig/unwichtig) Voraussetzung (ist je nach beteiligter Personengruppe meist unterschiedlich)
- Lösung eines allgemeines Problems als Grundlage zur Lösung eines speziellen Problems

Entwurfsprinzip Abstraktion (Verallgemeinerung)

- Abstraktion ist notwendig, um Komplexität meistern zu können
  - Oft lassen sich erst durch Abstraktion Gemeinsamkeiten (von Objekten, Situationen, Prozessen) erkennen
  - Gemeinsamkeiten zur Strukturierung des Systems nutzen
  - Abstraktionsebenen (grobe, mittlere, detaillierte Sicht) tragen zur Übersichtlichkeit bei
- Vielzahl von Abstraktionen erfordern eine weitere Gliederung
  - Modularisierung
  - Kapselung
  - Separation of Concerns

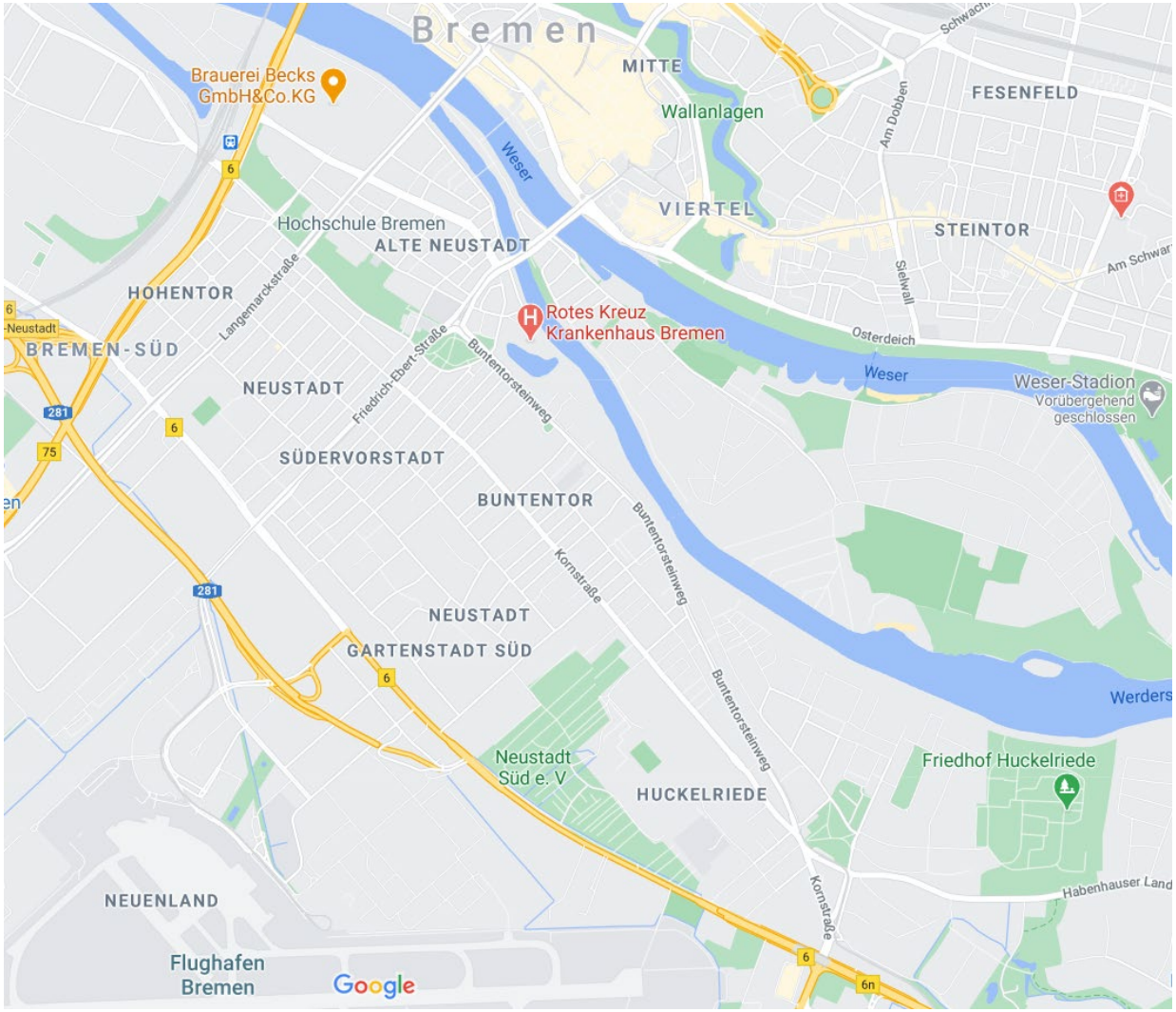
Modell als Abstraktion

- Ein Modell hat nach Stachowiak [1973] grundsätzlich
  - einen Zweck,
  - einen Bezug zu einem Original, und
  - abstrahiert bestimmte Eigenschaften des Originals
  - erfasst i. Allg. nicht alle Attribute des Originals
- Software ist immateriell
  - Modell von sich selbst bzw. durch sie beschriebenen Abläufen
  - Der Code ist ein Modell des ausführbaren Systems
- Zwischen dem Original und seinen Modellen bestehen vielfältige Beziehungen, die sich auch durch den jeweiligen Einsatzzweck bemerkbar machen, z.B.
  - zur Erringung eines Anforderungs- bzw. Systemverständnisses
  - zur konstruktiven Generierung von Code
  - zur qualitätssichernden Ableitung von Tests

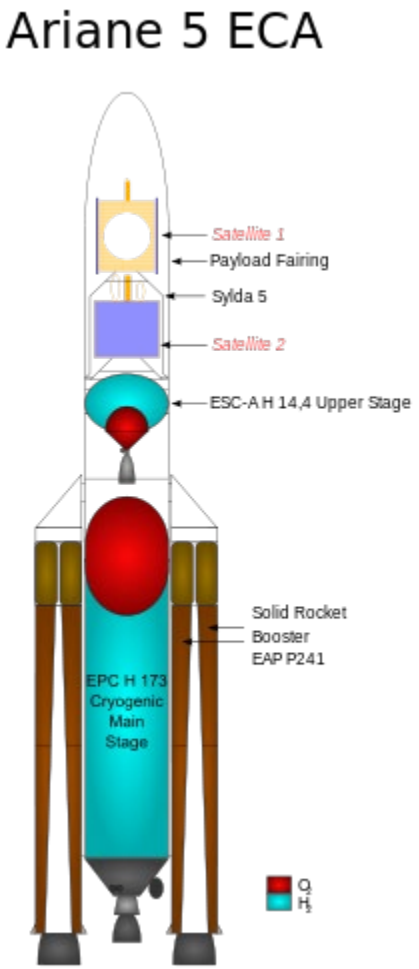
Modelltypen

- Deskriptive Modelle (z.B. ein Stadtplan) beschreiben ein Original zum leichteren Verständnis
- Präskriptive Modelle (z.B. ein Bebauungsplan) tragen zur Erstellung eines Originals bei

Stadtplan als deskriptives Modell



Technisches Konstrukt vs. Modell als Abbild/Vorbild



Prozesse der Modellbildung

- Abgrenzung
  - Nichtberücksichtigung irrelevanter Objekte
- Reduktion
  - Weglassen von Objektdetails
- Dekomposition
  - Zerlegung, Auflösung in einzelne Segmente
- Aggregation
  - Vereinigung von Segmenten zu einem Ganzen
- Abstraktion
  - Begriffs- bzw. Klassenbildung

Entwurfsprinzip Strukturierung

- Arten der Strukturierung
  - Zeitliche Strukturierung: Anforderungsanalyse → Entwurf → Programmierung → Test
  - Qualitative Strukturierung: Effizienz, Robustheit, Korrektheit
  - Perspektivische Strukturierung: Verwendung von Datenstrukturen, Datenfluss, Kontrollfluss
  - Dekomposition (Strukturierung in Bestandteile / Komponenten)
    → Diese Strukturierung ist so wesentlich, dass sie unter dem Prinzip Modularisierung gesondert betrachtet wird!
- Strukturierung ist wichtig auch für
  - Teamarbeit
  - Arbeitsteilung
  - Zuordnung von Verantwortlichkeiten

Entwurfsprinzip Modularisierung - Modul

- Eine logische Einheit von Daten und Operationen
- Repräsentiert Entwurfsentscheidungen
- Besitzt eine gewisse Komplexität
- Soll keine Nebeneffekte zulassen
- Ist ersetzbar und soll getrennt übersetzbar sein
- Ist eine zur Wiederverwendung geeignete Einheit
- Ist separat prüfbar

Entwurfsprinzip Modularisierung - System

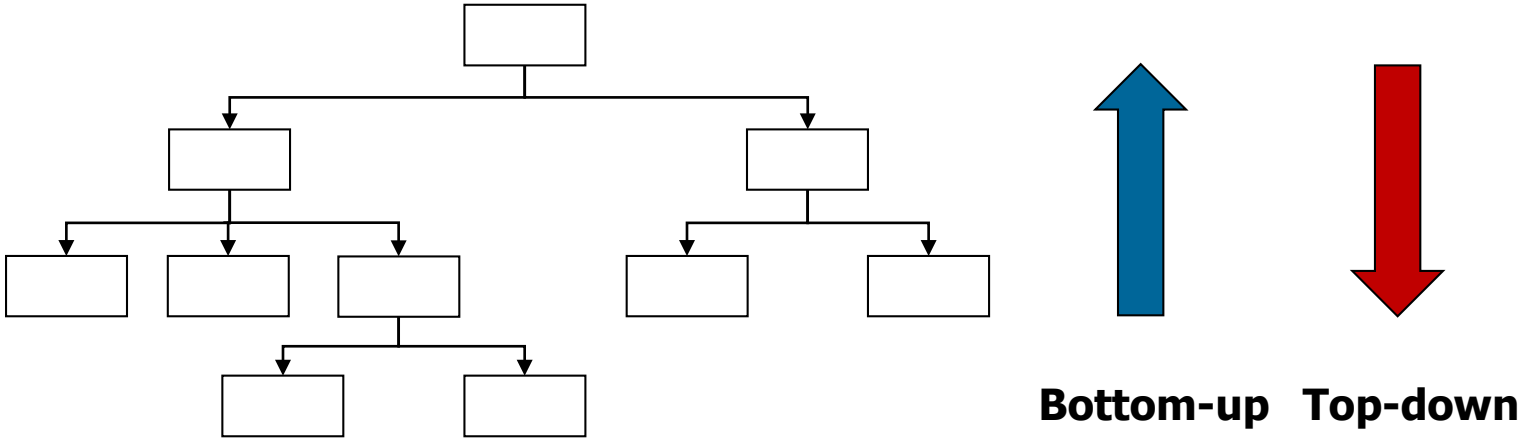
- Zerlegung bzw. Zusammenbau
- Die Eigenschaft eines Systems, das aus einer Menge von in sich geschlossenen & lose gekoppelten Modulen besteht (Kohäsion & Kopplung)
- Modularität und Kapselung - Unterscheidung zwischen Schnittstelle & Implementierung eines Moduls
- **Wichtig**: "richtige" Granularität wählen
- Dient der Beherrschung der Komplexität (in Ergänzung zur Abstraktion)
- Modularisierung erleichtert
  - Das Verstehen der Programme
  - Den Umgang mit Änderungen (Information Hiding, Kapselung)
  - Die Zuordnung der Verantwortlichkeiten (Separation of Concerns)
  - Die Wiederverwendung bereits implementierter und benutzter Module

Entwurfsprinzip Modularisierung - Der Prozess

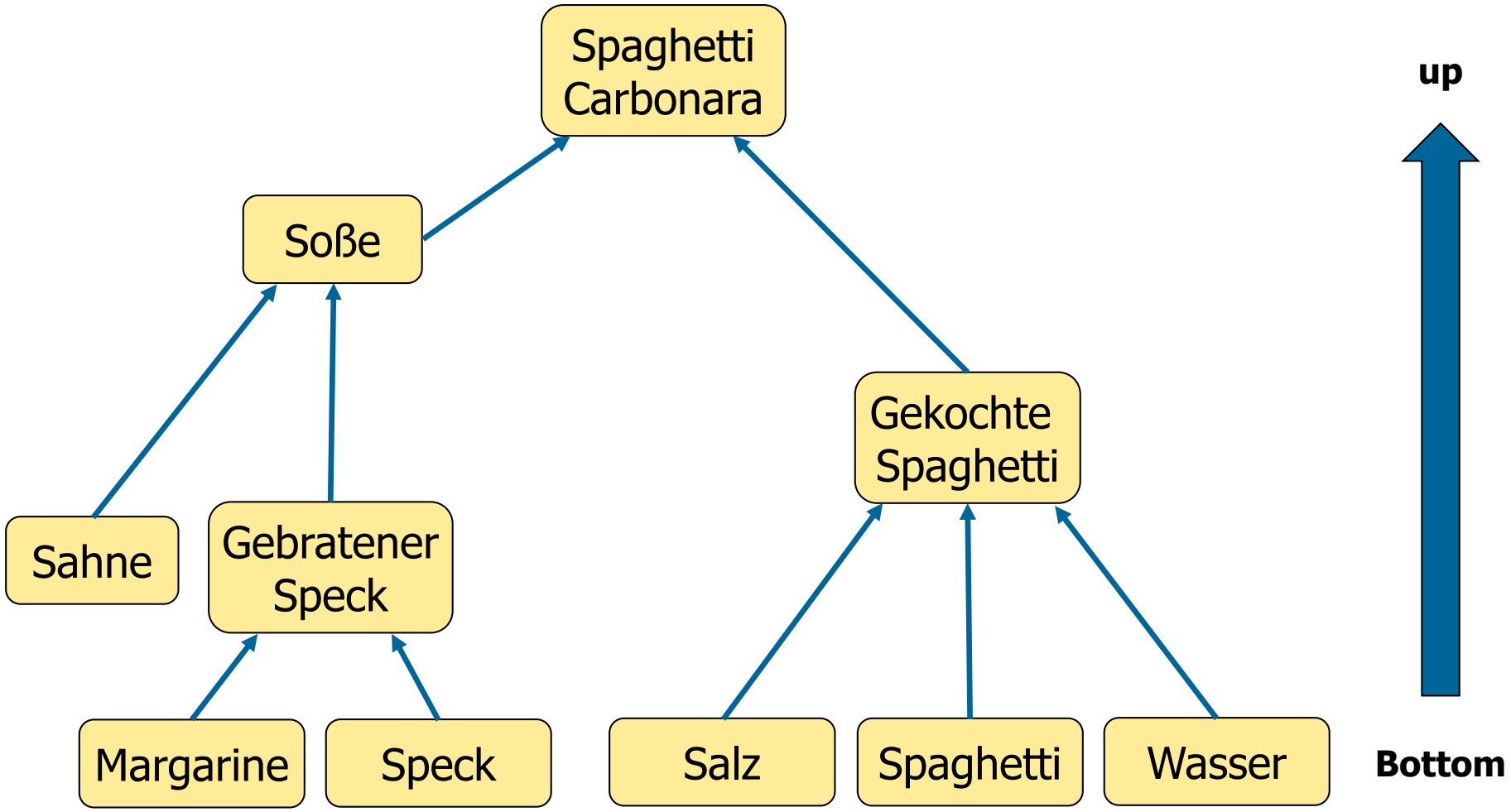
- Bottom-up
  - Zuerst werden unabhängige Module konstruiert
  - Danach werden diese „zusammengebaut“

  → **Zusammensetzbarkeit**
- Top-down
  - Zuerst wird das Gesamtsystem in unabhängige Module aufgeteilt (zerlegt)
  - Danach werden diese einzeln implementiert werden

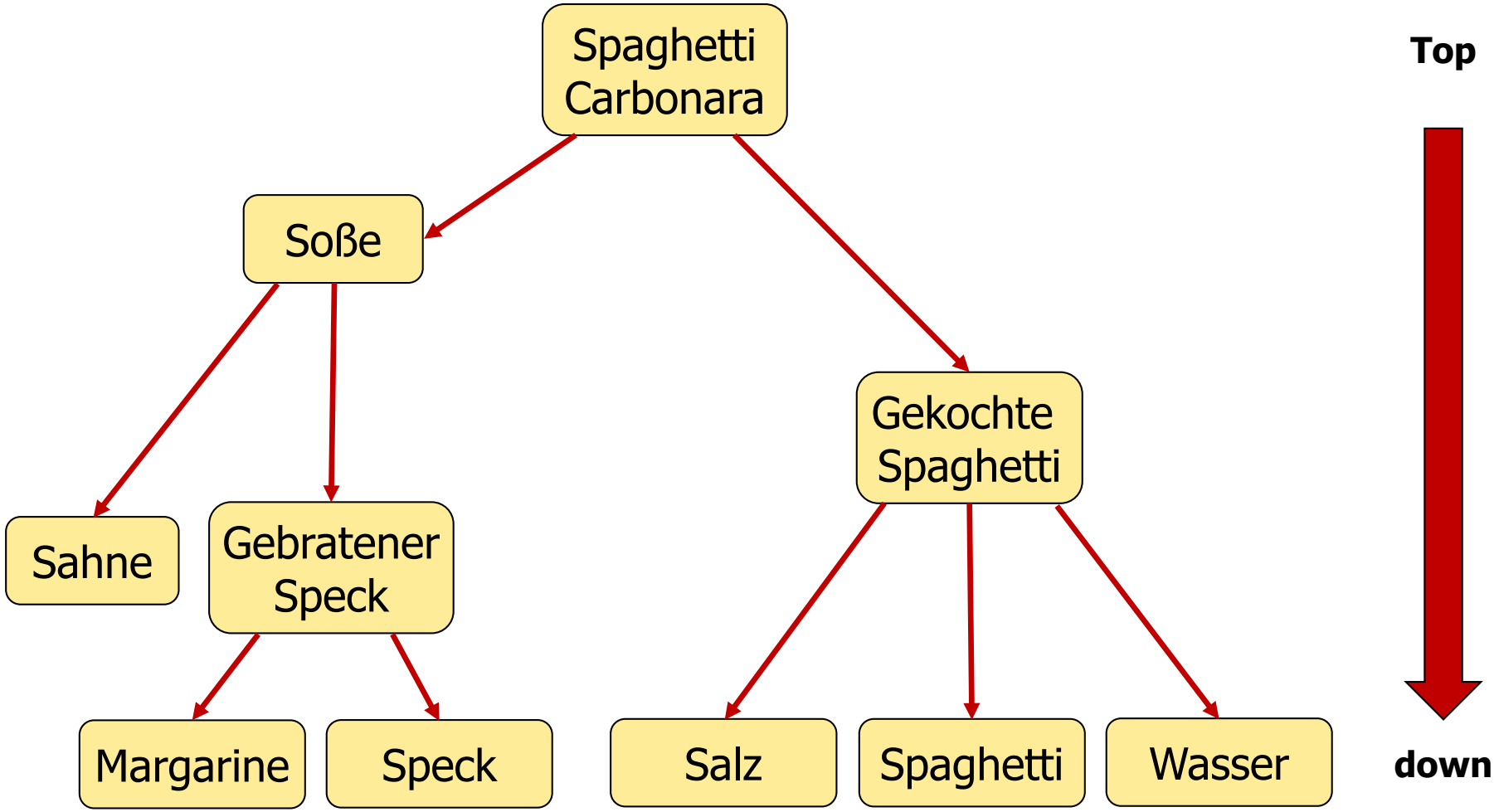
  → **Zerlegbarkeit**



Bottom-up: aus Einzelteilen ein „Ganzes“ bilden



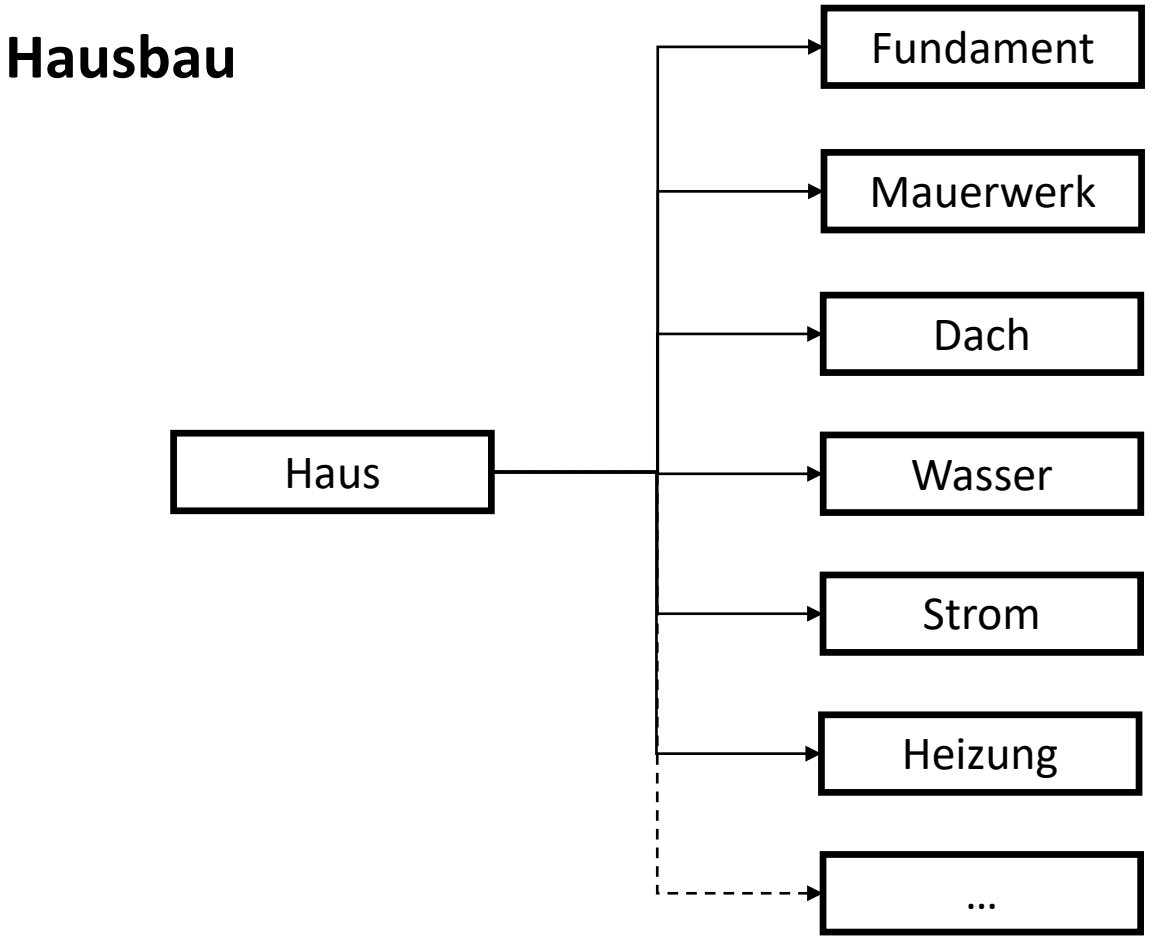
Top-down: vom „Ganzen“ die Einzelteilen finden



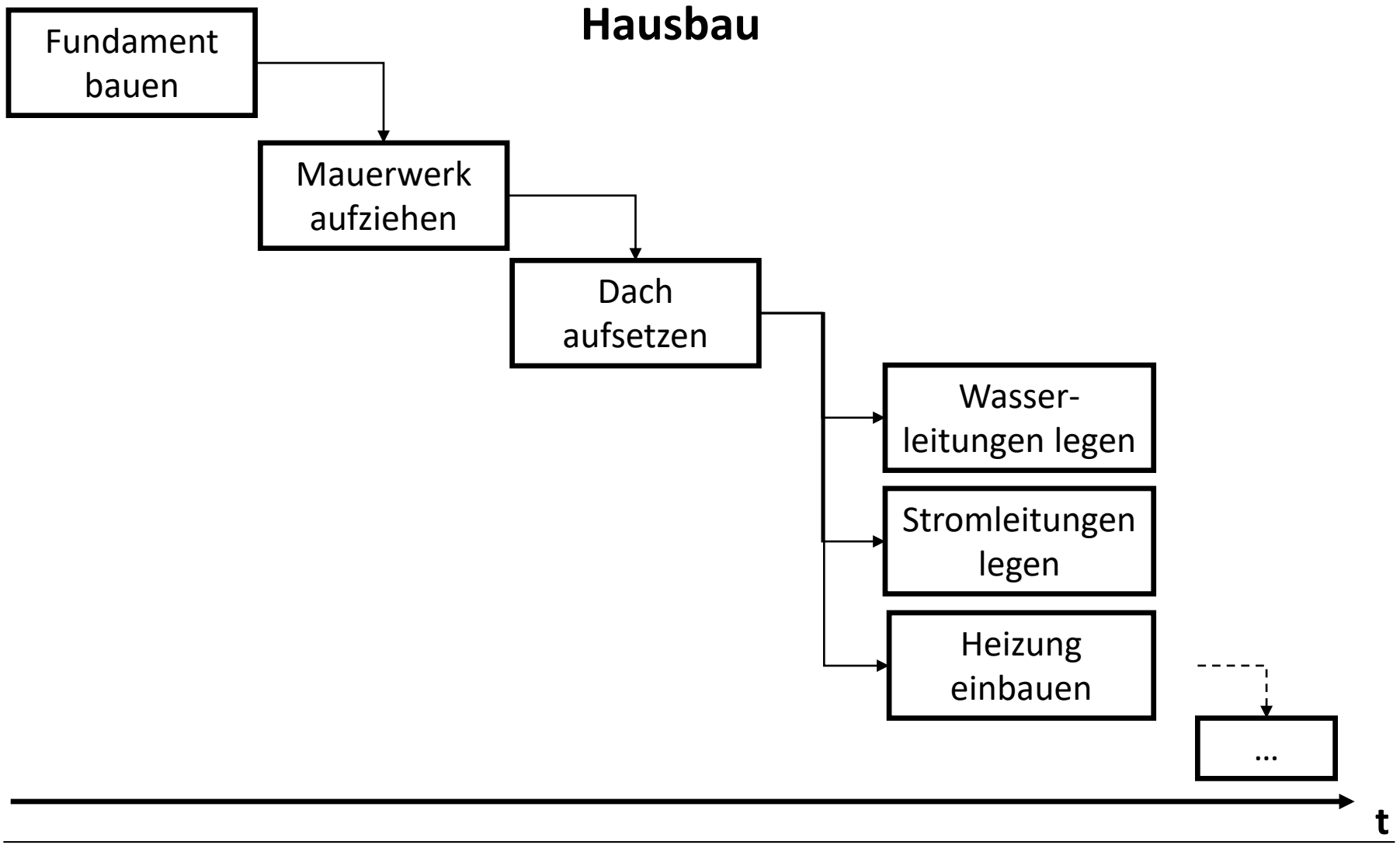
Entwurfsprinzip Hierarchie

- Zusammenfassung von logisch zusammengehörenden Modulen (oder Abstraktionen)
- weiteres Strukturierungsmittel
- Nutzung von Rangfolgen
  - Enthalten-in, besteht-aus - Hierarchie → Struktur
  - „ist-ein“ – Hierarchie → Generalisierung
- Schichtenhierarchie (Abstraktion, Aufgabenteilung)

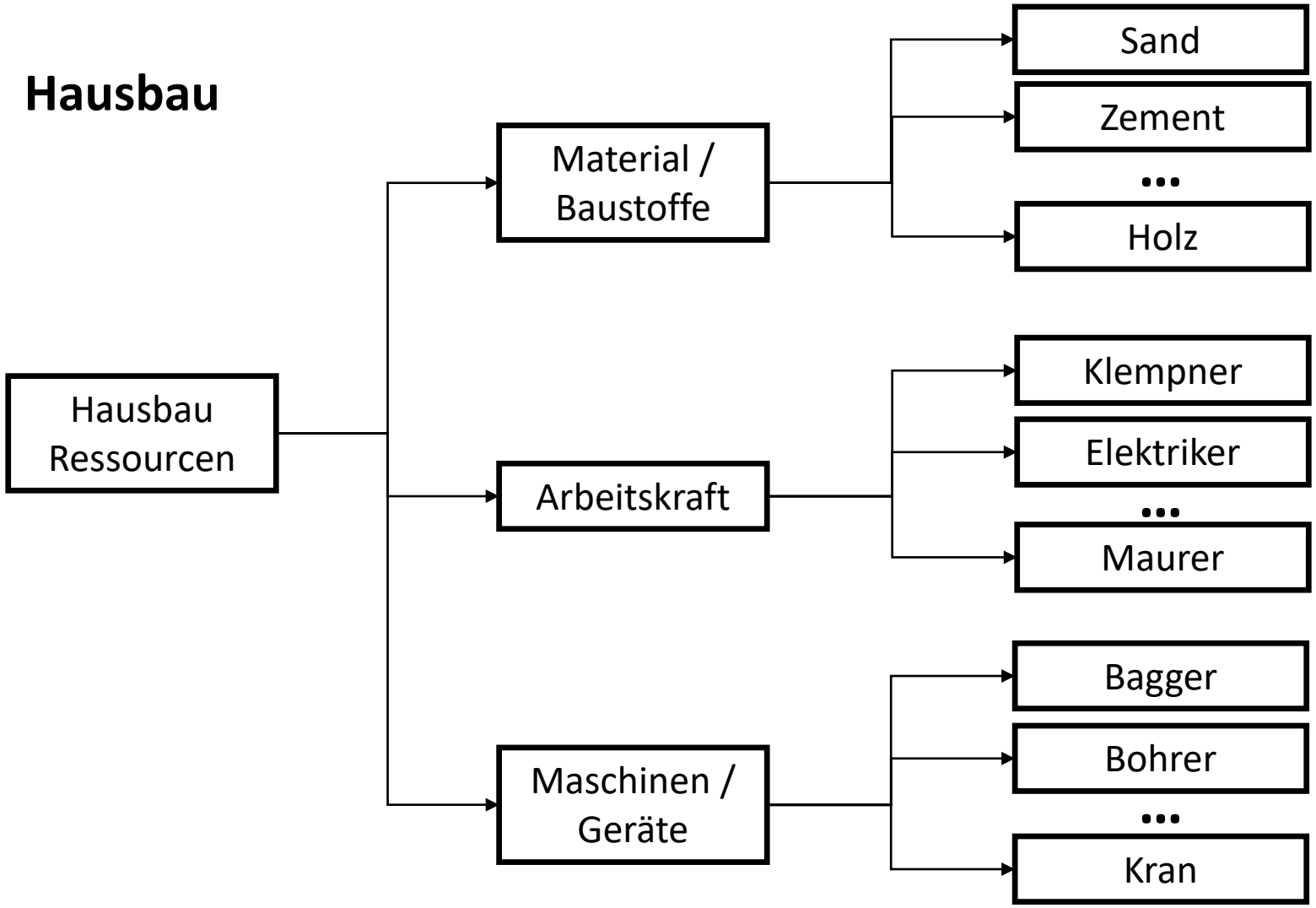
Funktionale Zerlegung



Zeitliche Zerlegung



Ressourcen Zerlegung



Entwurfsprinzip Kapselung

- Begrenzen der Details eines Moduls, die von außen sichtbar sind
- Abgrenzen von Verantwortungsbereichen
- Verringern von Abhängigkeiten zwischen Modulen
- Information hiding (Geheimnisprinzip)
  - Innerer Aufbau eines Bausteins
  - Zentrales Prinzip der Objektorientierung
  - Kapselung (Innensicht) und Abstraktion (Außensicht)
- Zusammenhang mit anderen Prinzipien:
  - Modularisierung wird unterstützt
  - Abstraktionen werden abgegrenzt

Entwurfsprinzip Separation of Concerns [Dijkstra, 1974]

- Trennung von Belangen / Zuständigkeiten
- Aufteilen in unabhängige Belange / Aspekte, z.B.
  - in Kontroll- und Datenfluss,
  - in fachlich und technisch geprägte Lösungsteile,
  - nach Produkteigenschaften,
  - nach externen und internen Qualitätskriterien

Entwurfsprinzip Gründlichkeit und Formalität

- Aspekte u.a.:
  - Formale Anforderungsspezifikation
  - Formale Dokumentation des Software-Prozesses und seiner Ergebnisse
  - Formaler Entwurf
  - Formale Verifikation der Korrektheit von Software
- Formalität ist (zumeist) Voraussetzung für einen Werkzeugeinsatz
- Formalität benötigt i.A. mathematische Fertigkeiten und zumeist auch eine gewisse Disziplin

Übung

- Was ist ein Entwurfskriterium?
- Welche Entwurfskriterien kennen Sie?
- Was ist ein Entwurfsprinzip?
- Zu welchem Prinzip kann die Modellbildung zugeordnet werden?
- Zu welchem Prinzip gehören die Begriffe top-down & bottom-up?
- Woraus bestehen Subsysteme?
- Was ist Software-Architektur?
- Welches Prinzip ist die Voraussetzung für einen Werkzeugeinsatz?

Fragen

- Warum so viele Kriterien und Prinzipien?
- Warum unterschiedliche und so komplizierte Darstellungen?
- Was meinen Sie?

Entwurf

- Unterschiedliche "Sichtweisen" auf ein Problem
- Unterschiedliche Aspekte klarer herausarbeiten
- Klärung des "Ganzen" dadurch erreichen!

Sichtweisen/Vorgehensweisen zum Entwurf

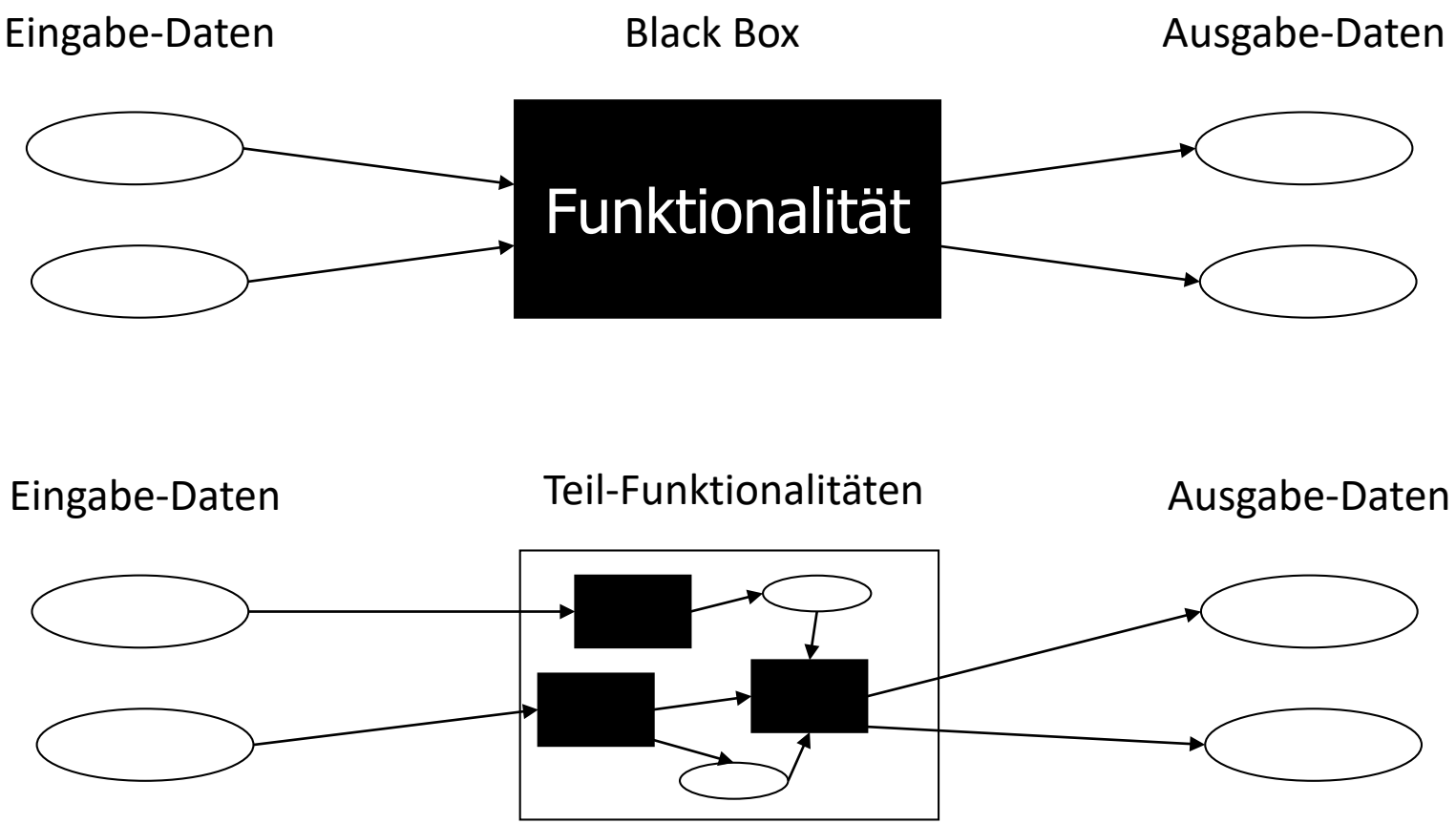
- Funktionsorientierter Entwurf
  - Im Mittelpunkt steht die funktionsorientierte Systemsicht
  - Ausgehend von den funktionalen Anforderungen wird eine aufgabenorientierte Zerlegung des Gesamtsystems vorgenommen
- Datenorientierter Entwurf
  - Im Mittelpunkt stehen die zu verarbeitenden Daten
  - Die Struktur der Daten dient zur Strukturierung des Systems
  - Die Analyse der zu verarbeitenden Daten soll zum Entwurf führen
- Objektorientierter Entwurf
  - Objekte der realen Welt dienen zur Modellierung

Funktionsorientierter Entwurf

- Festlegung der Grobstruktur des Systems
- Feinentwurf der Teilaufgaben
- Schrittweise Verfeinerung (stepwise refinement)
  1. Zerlege die Aufgabe in Teilaufgaben
  2. Betrachte jede Teilaufgabe für sich und zerlege sie wieder in »kleinere« überschaubare Teilaufgaben
  3. Setzte dies solange fort, bis die Teilaufgaben so einfach sind, dass sie sich ohne weiteres in eine Programmiersprache umsetzen lassen

Funktionsorientierter Entwurf - Vorgehen

- Identifikation der wesentlichen Bestandteile und ihr Beitrag zur Gesamtlösung
- Zusammenhang zum übrigen System klären
- Komplexitätsminderung durch schrittweise Verfeinerung, Abstraktion von Details
- Teilaufgaben werden zunehmend konkreter
- Detailinformationen werden gebraucht
- Festlegung der Datenstrukturen
- **Wichtig:** bei jedem Schritt prüfen, ob die bisherige Zerlegung sinnvoll ist, oder geändert werden sollte!

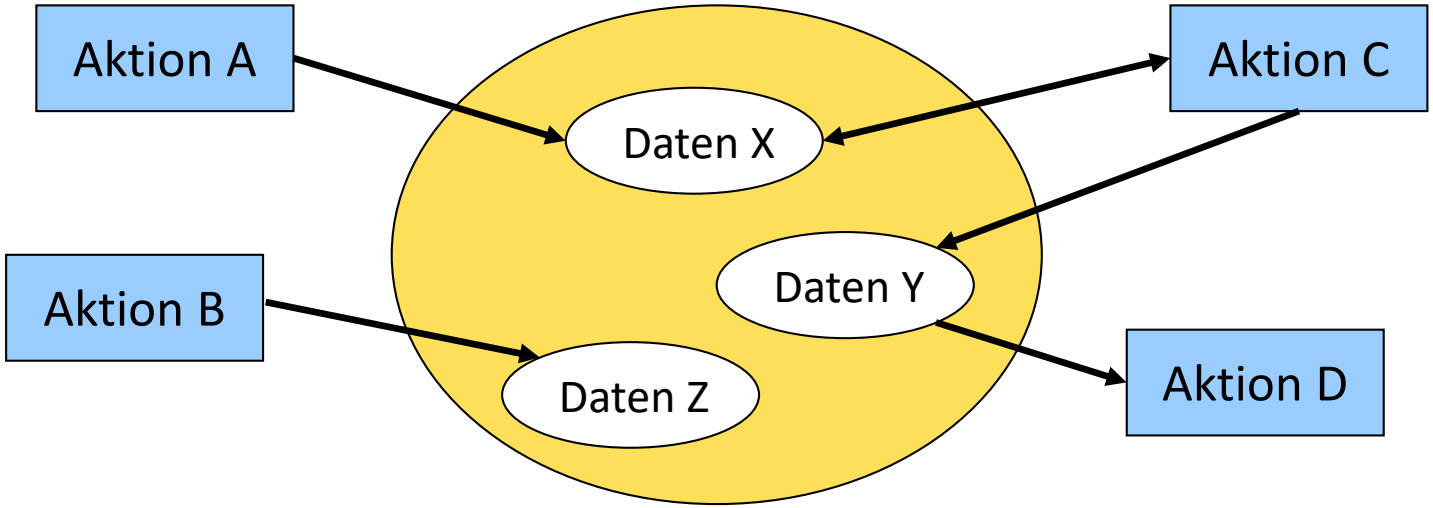


Ergebnis des 1. Verfeinerungsschrittes

Datenorientierter Entwurf

- Aus der Struktur der Daten die Algorithmen zu ihrer Verarbeitung ableiten
- Konzentration auf Datenfluss und Datenstrukturen
- Häufige Aufgabenstellung: Strom von Eingaben, der verarbeitet werden soll (z.B. Compiler)
- Eingabedaten haben eine vorgegebene Struktur, die mit einer Grammatik beschrieben werden kann.
- Grammatik dient auch als Grundlage zur Analyse des Datenstroms
- Geeignet bei
  - Komplexen und (oder) vielen verschiedenen Daten
  - Datenbanken (ER-Diagramme)
  - Strukturierten Datenströmen (attributierte Grammatiken)

Datenorientierter Entwurf



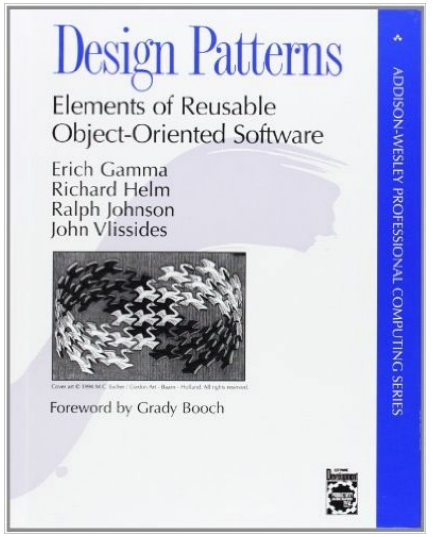
Objektorientierter Entwurf: Definition

- Kerngedanke: der Programmablauf ist ein Zusammenspiel von Objekten und ihren Interaktionen
- Objekte der realen Welt werden modelliert!
  - Funktionen und Daten werden als Einheit betrachtet: Klassen = Attribute (Daten) + Methoden (Funktionen)
  - Hauptelemente: Klassen, Objekte, Interfaces
- Als graphische Modellierungssprache wird die UML (Unified Modeling Language - vereinheitlichte Modellierungssprache) eingesetzt
- Gutes OO-Design besteht darin, jede Klasse so zu gestalten
  - dass deren Aufgabe eindeutig ist
  - jede Aufgabe nur durch eine Klasse realisiert wird
  - die Abhängigkeiten zwischen Klassen möglichst minimal sind

→ Nächste Vorlesung: Objektorientierter Entwurf mit UML

Entwurfsmuster (mehr dazu in SOFTW2)

- Bewährte, generische Lösungen für immer wiederkehrende Entwurfsprobleme, die in bestimmten Situationen auftreten
- **Buch:** Design Patterns, Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, 1996 [Gamma et al. 1996] → *Gang of Four*
- Klassifikation
  - Erzeugungsmuster (Creational Design Patterns)
  - Strukturmuster (Structural Design Patterns)
  - Verhaltensmuster (Behavoiral Design Patterns)



Erzeugungsmuster

- Helfen, ein System unabhängig davon zu machen, wie seine Objekte erzeugt, zusammengesetzt und repräsentiert werden
- Klassenmuster
  - **Factory Method** (Fabrikmethode) - Klassenschnittstelle zur Erzeugung eines Objekts mit delegierter Erzeugung an Unterklassen
- Objektmuster
  - **Singleton** (Einzelstück) - Sicherstellung, dass eine Klasse genau ein Objekt besitzt (globalen Zugriff erlaubt)
  - **Abstract Factory** (Abstrakte Fabrik) - Schnittstelle zur Erzeugung von Familien verwandter oder abhängiger Objekte, ohne konkrete Klassen zu benennen
  - **Builder** (Erbauer) – Trennung der Konstruktion eines komplexen Objekts von seiner Repräsentation
  - **Prototype** (Prototyp) - Erzeugung eines prototypisches Exemplar und neue Objekte durch Kopieren des Prototyps

Strukturmuster

- Befassen sich mit der **Zusammensetzung** von Klassen und Objekten zu größeren **Strukturen** mit unterschiedlicher Funktionalität. Die Flexibilität entsteht durch die Möglichkeit die **Komposition zur Laufzeit** zu verändern:
  - **Composite** (Kompositum) wird verwendet um Teil-Ganzes-Hierarchien darzustellen
  - Im **Proxy** Muster dient ein Proxy als Platzhalter für ein anderes Objekt, z.B. auf einem anderen Rechner
  - **Flyweight** (Fliegengewicht) definiert eine Struktur zum Teilen von Objekten
  - Mit **Facade** repräsentiert ein einzelnes Objekt ein ganzes Teilsystem
  - **Adapter** (Wrapper) übersetzt eine Schnittstelle in eine andere um die Kommunikation von Klassen mit zueinander inkompatiblen Schnittstellen zu ermöglichen
  - Das **Bridge** (Brücke) Muster trennt die Abstraktion eines Objektes von dessen Implementierung
  - **Decorator** beschreibt wie Verantwortlichkeiten an ein Objekt dynamisch hinzugefügt werden können

Verhaltensmuster

- Befassen sich mit der Interaktion zwischen Objekten und Klassen und beschreiben komplexe Kontrollflüsse, die zur Laufzeit schwer nachvollziehbar sind
- Klassenmuster
  - **Interpreter** (Interpreter)
  - **Template Method** (Schablonenmethode)
- Objektmuster
  - **Chain of Responsibility** (Zuständigkeitskette)
  - **Command** (Befehl, Kommando, Action, Transaction)
  - **Iterator** (Iterator, Cursor)
  - **Mediator** (Vermittler)
  - **Memento** (Memento, Token)
  - **Observer** (Beobachter, Dependents, Publish-Subscribe)
  - **State** (Zustand, Objects for States)
  - **Strategy** (Strategie, Policy)
  - **Visitor** (Besucher)

Subsysteme

- Probleme
  - zu viele kleine Teile (Module, Klassen, ...) mit zu vielen Beziehungen erschweren Verständnis erheblich
  - zu viele Beziehungen zwischen den Teilen erzeugen zu viele Abhängigkeiten
- Lösungsideen
  - Weitere Abstraktion einführen
  - Gruppieren von Teilen -> Subsystem (Teilsystem)

Software-Architektur (mehr dazu in SOFTW2)

- Definition: **Software-Architektur** ist die grundlegende Organisation eines Systems, dargestellt durch dessen Komponenten, deren Beziehungen zueinander und zur Umgebung sowie die Prinzipien, die den Entwurf und die Evolution des Systems bestimmen

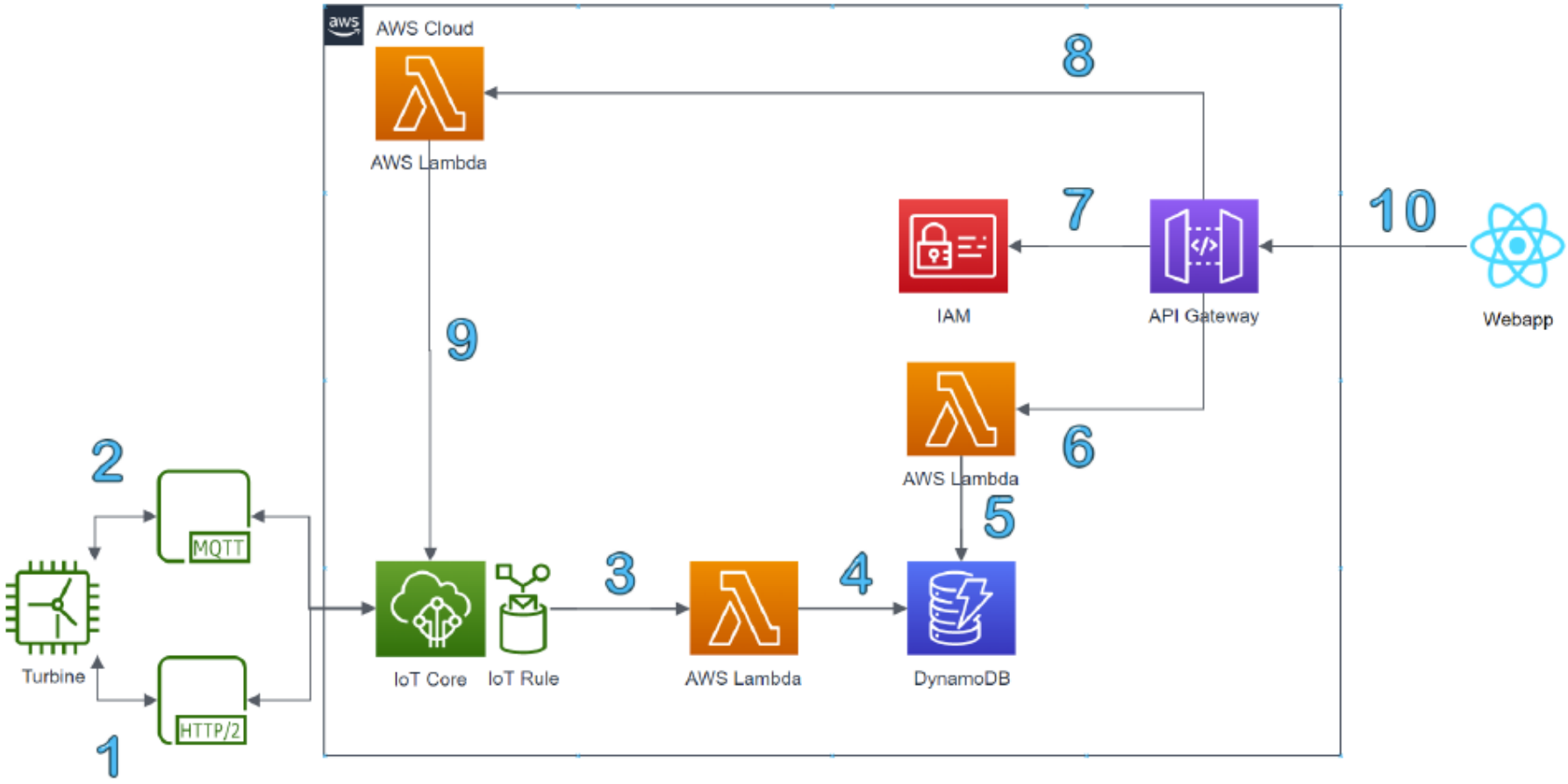
  [angelehnt an den ANSI/IEEE 1471-2000 Standard]

- Definition: **Architekturbeschreibung** ist eine Menge von Modellen (z.B. textuelle Spezifikationen oder graphische Diagramme wie die UML-Diagramme), die die Software-Architektur dokumentieren.

Architekturmuster

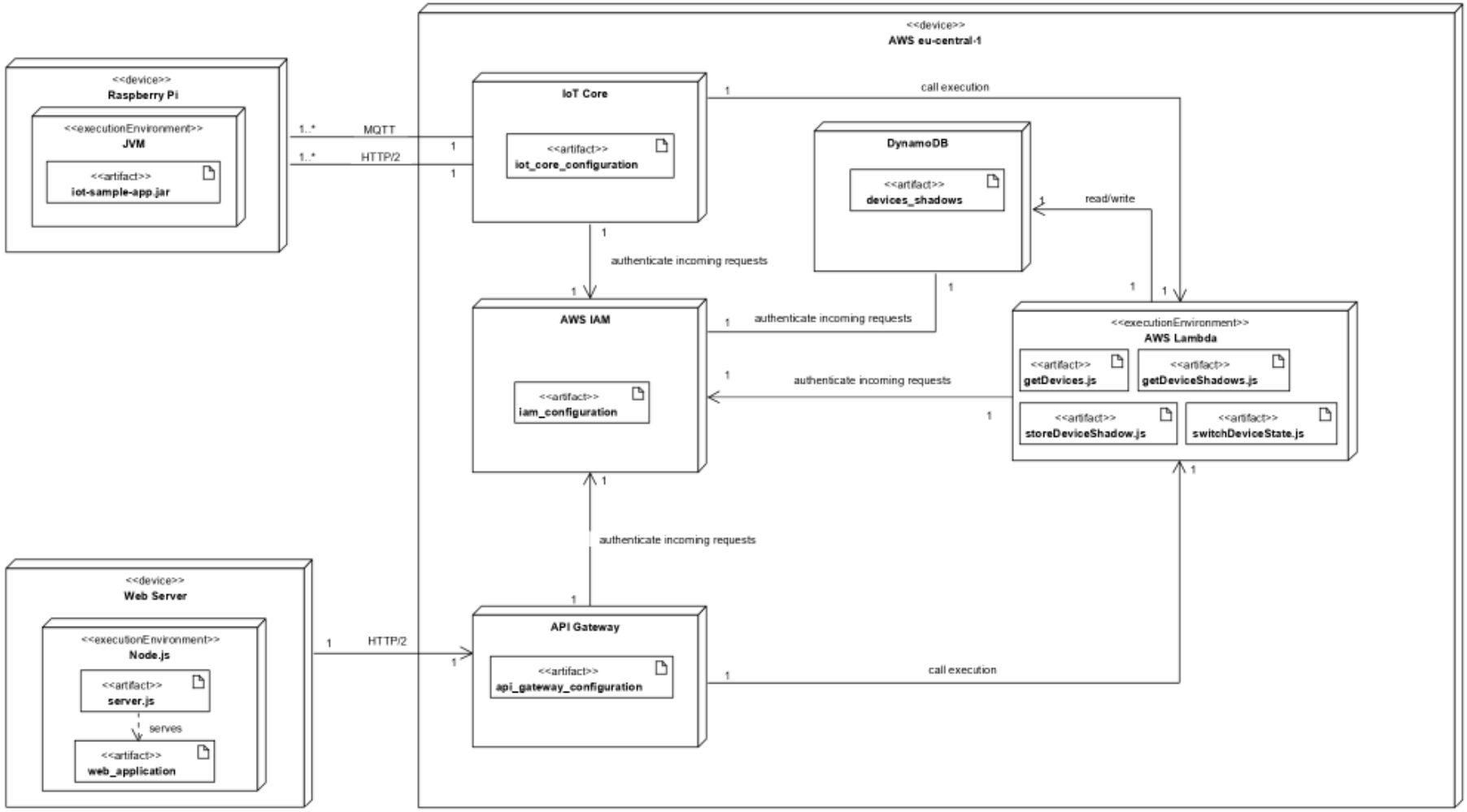
- Beschreiben die grundlegende Organisation und Interaktion zwischen den Komponenten einer Anwendung
- Bekannte Architekturmuster
  - **Client-Server:** Mehrere Clients greifen asynchron auf Dienste eines Servers zu
  - **Peer-to-Peer:** Die Clients sind gleichberechtigt und tauschen Dienste aus
  - **Model-View-Controller (MVC):** Aufteilung in Model (Datenhaltung), View (Darstellung) und Controller (Steuerung)
  - **Schichten-Modelle:** Das System wird in mehreren aufeinander “gestapelten” Schichten strukturiert; jede Schicht kommuniziert nur mit ihren Nachbarschichten
  - **Service Oriented Architecture (SOA):** Kapselt die Komponenten in nach außen sichtbare Dienste
  - **Pipes & Filters:** Filter verarbeiten Datenstöme und reichen diese durch Pipes an nachfolgende Filter weiter
  - → mehr dazu in Softwaretechnik 2 im SoSe

Beispiel: Gesamtarchitektur



[BT M. Soldin, Juni 2020]

Beispiel: Gesamtarchitektur



[BT M. Soldin, Juni 2020]

Heuristiken/Erfahrungen

- Aus dem Projektumfeld
  - Hinterfragen Sie Anforderungen
  - Betrachten Sie Use-Cases
  - Nutzen Sie andere Systeme als Inspiration
  - Suchen Sie Feedback
- Für den Entwurf
  - So einfach wie möglich
  - Bleiben Sie auf der „richtigen“ Detailebene
  - Modellieren Sie angemessen komplex
  - Denken Sie in Verantwortlichkeiten
  - Nicht übergeneralisieren
  - Nicht übermodellieren
  - Konzentrieren Sie sich auf die Schnittstellen
  - Entwerfen Sie so lokal wie möglich

[aus T. Posch et al.: Basiswissen Softwarearchitektur, dpunkt, 2004]

Heuristiken / Erfahrungen

- Zur Beherrschung von Komplexität
  - Teile und herrsche
  - Verstecken Sie Details
  - Kapseln Sie Risiken
  - Trennen Sie Funktionalität und Kontrolle
  - Trennen Sie Fachlogik und Infrastruktur
- Zur Arbeitsmethodik
  - Beginnen Sie mit den schwierigsten Teilen
  - Verwenden Sie Prototypen
  - Werfen Sie Prototypen weg
  - Dokumentieren Sie Entscheidungen

[aus T. Posch et al.: Basiswissen Softwarearchitektur, dpunkt, 2004]

Zusammenfassung - Was Sie beantworten können sollten

- Es gibt folgende Entwurfskriterien …
- Es gibt folgende Entwurfsprinzipien …
- top-down und bottom-up-Entwurf unterscheiden sich in ...
- Modularität ist ...
- Abstraktion ist …
- Entwurfsmuster sind … Es gibt folgende drei Kategorien:
- Architekturmuster sind … Folgende sind die meistbekannten:
- Software-Architektur ist …
- Beim funktions-/daten-/objektorientierten Entwurf steht im Mittelpunkt der Überlegungen ...
- **+ alle Fragen in den Übungsfolien!**

Literaturhinweise

- **Primär: Materialien im AULIS!**
- I. Sommerville - Software Engineering, 2012
- C. Rupp, S. Queins, die SOPHISTen - UML 2 glasklar: Praxiswissen für die UML-Modellierung, 2012
- E. Gamma, R. Helm, R. E. Johnson - Design Patterns. Elements of Reusable Object-Oriented Software, 1996
- P. Clements - Documenting Software Architectures: Views and Beyond, 2003
- L. Bass, P. Clements, R. Kazman - Software Architecture in Practice, 2012
- ISO/IEC/IEEE 15288:2008 - Systems and software engineering -- System life cycle processes
- ISO/IEC/IEEE 16326:2009 - Systems and software engineering -- Life cycle processes -- Project management
- ISO/IEC 25010:2011 - Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models
- A. Spillner, T. Linz - Basiswissen Softwaretest, 2019
- M. Broy, M. Kuhrmann - Projektorganisation und Management im Software Engineering, 2013
- S. Röpstorff, R. Wiechmann - Scrum in der Praxis, 2016