AUTOMOTIVE | INFOKOM | VERKEHR, UMWELT & ENERGIETECHNIK | LUFTFAHRT | RAUMFAHRT | VERTEIDIGUNG & SICHERHEIT

Modularisierung von Simulationssystemen

Erfahrungen aus der Praxis

Dr. Martin Rother

IABG • Ottobrunn • 21.01.2013

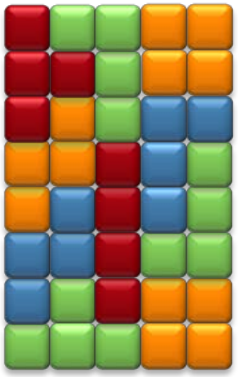
Agenda

- Mehrwert durch Modularisierung?
- Beobachtungen, Softwaretechnik
- Modellbildung
- Dokumentation
- Systemarchitektur

Querverweis: ITIS-Studie „Komponentenbasierte Modellentwicklung“

Mehrwert durch Modularisierung

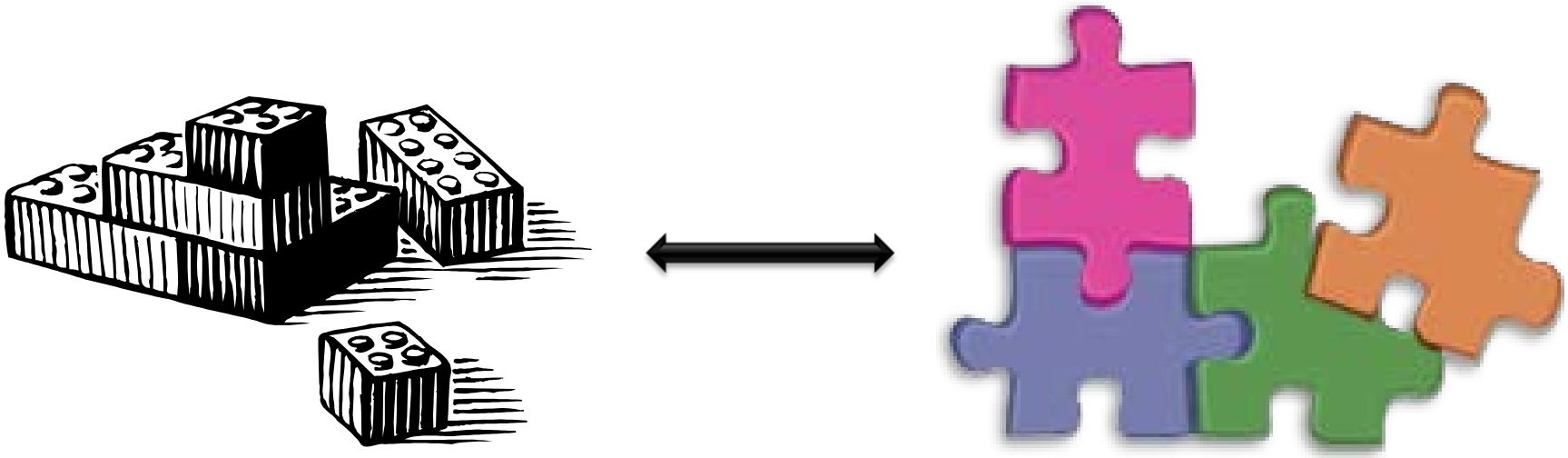
- „Legoprinzip“
- Wiederverwendung
- Flexibilisierung
- Zeitersparnis
- Qualität



- alle Simulationssysteme sind „modular“ bzw. aus Komponenten aufgebaut
- ergibt sich aus Systemanalyse, findet sich im Design wieder

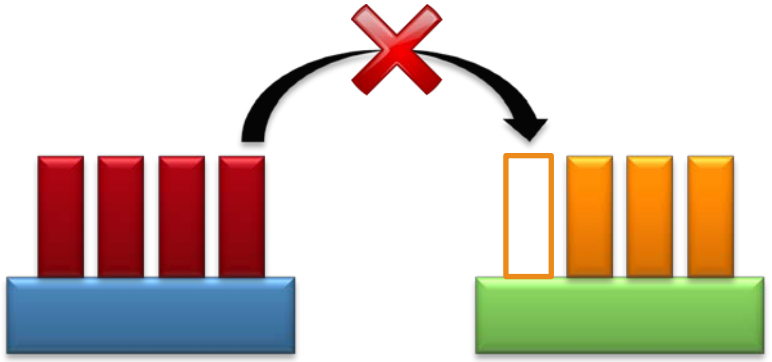
Typische Architektur

Wird der behauptete Mehrwert in der Praxis auch tatsächlich realisiert?

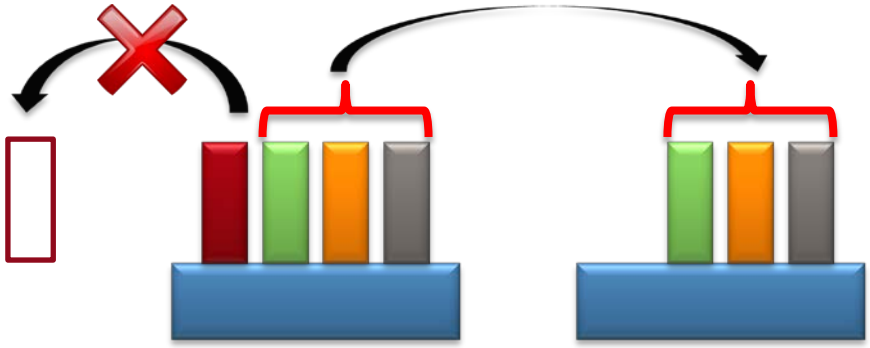


Beobachtungen

- Systeme sind nur „in sich“ modular
  - nicht über Systeme hinweg
  - nicht über Architekturen hinweg

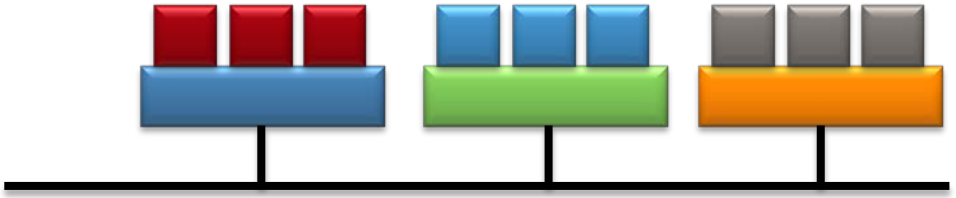


- Systeme sind nur „in sich“ flexibel
  - Komponenten nicht allein wiederverwendbar
  - Abhängigkeiten über Komponenten hinweg
  - nur bestimmte Komponenten passen zusammen



- Wiederverwendung oft Makroskopisch oder Mikroskopisch
  - vernetzte Simulation

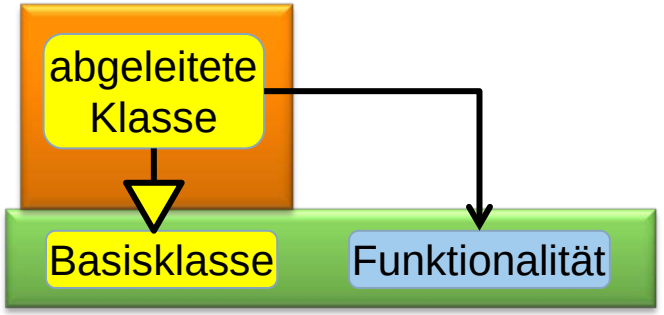
DIS, HLA, DDS, …



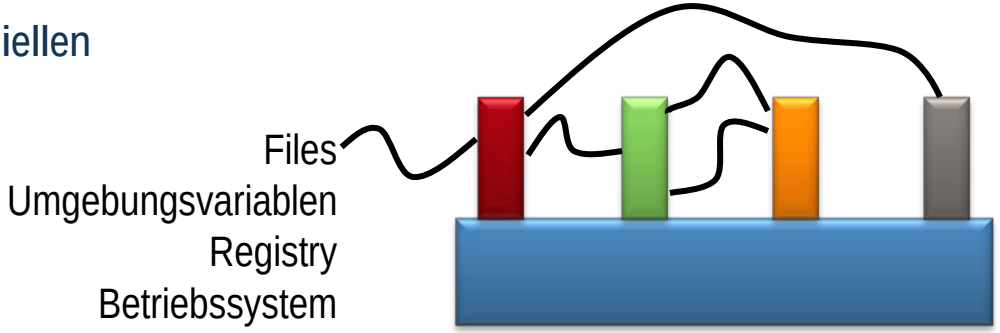
  - „kleine“ Funktionalitäten, z.B. Numerik-Bibliotheken

Ursachen

- Softwaredesign
  - Vererbung
  - unmittelbare und mittelbare Abhängigkeiten



- Entwickler
  - Querverbindungen an der offiziellen Schnittstelle vorbei



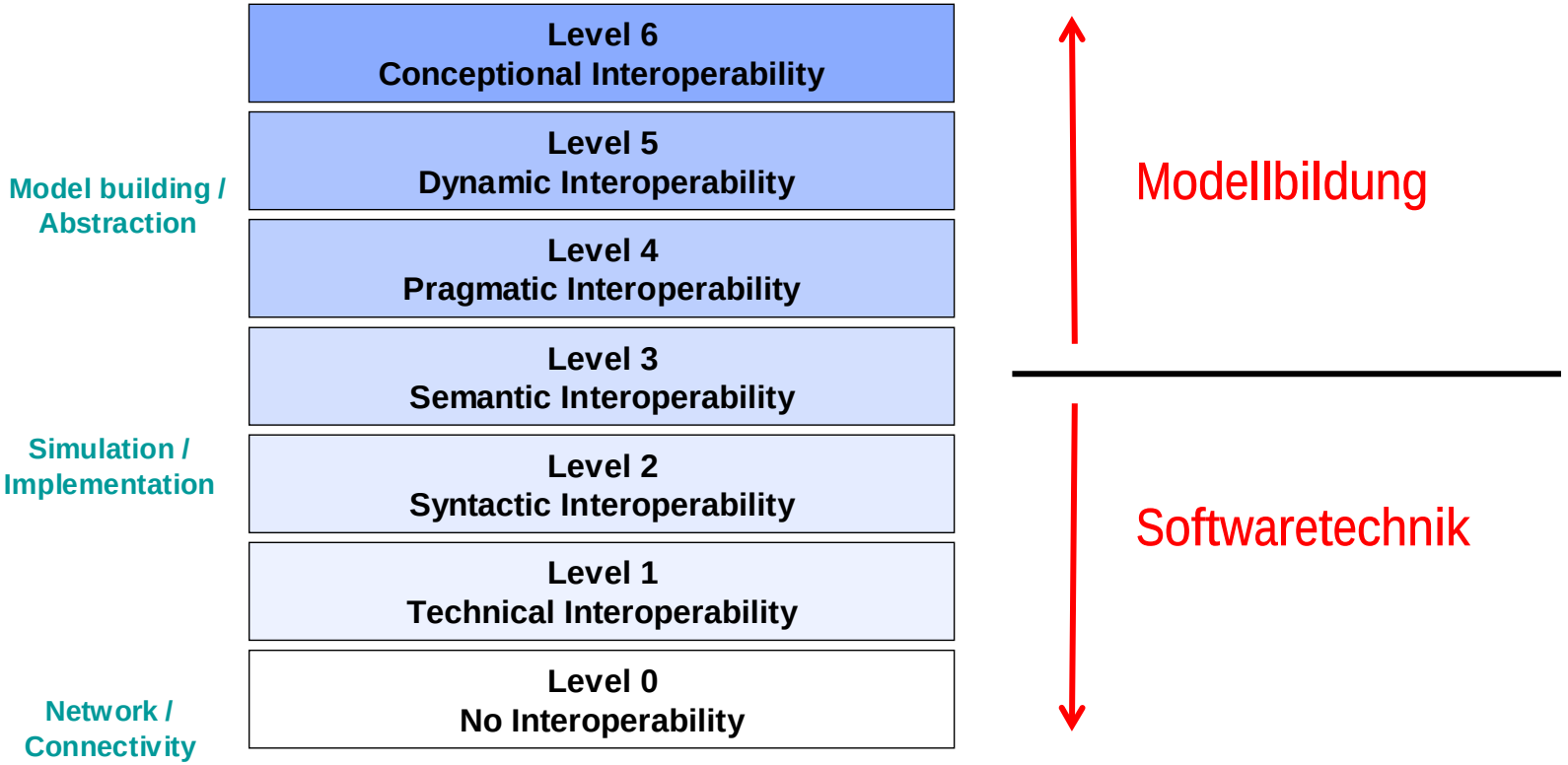
  - Featuritis
  - unnötige Verallgemeinerungen
  - individuelle Vorlieben
  - Hypes
  - „Pattern-Overflow“

} Verletzung des KISS-Prinzips

„Leicht“ zu beheben, da „nur“ Softwaretechnik und Management

Interoperabilität

von Simulationssystemen → gilt auch für Komponenten, Module und Modelle



A. Tolk et. al.

Querverweis: NATO MSG-86

Modularität von Simulationssystemen braucht mehr als Softwaretechnik!

Wie müssen „gute“ Module aussehen?

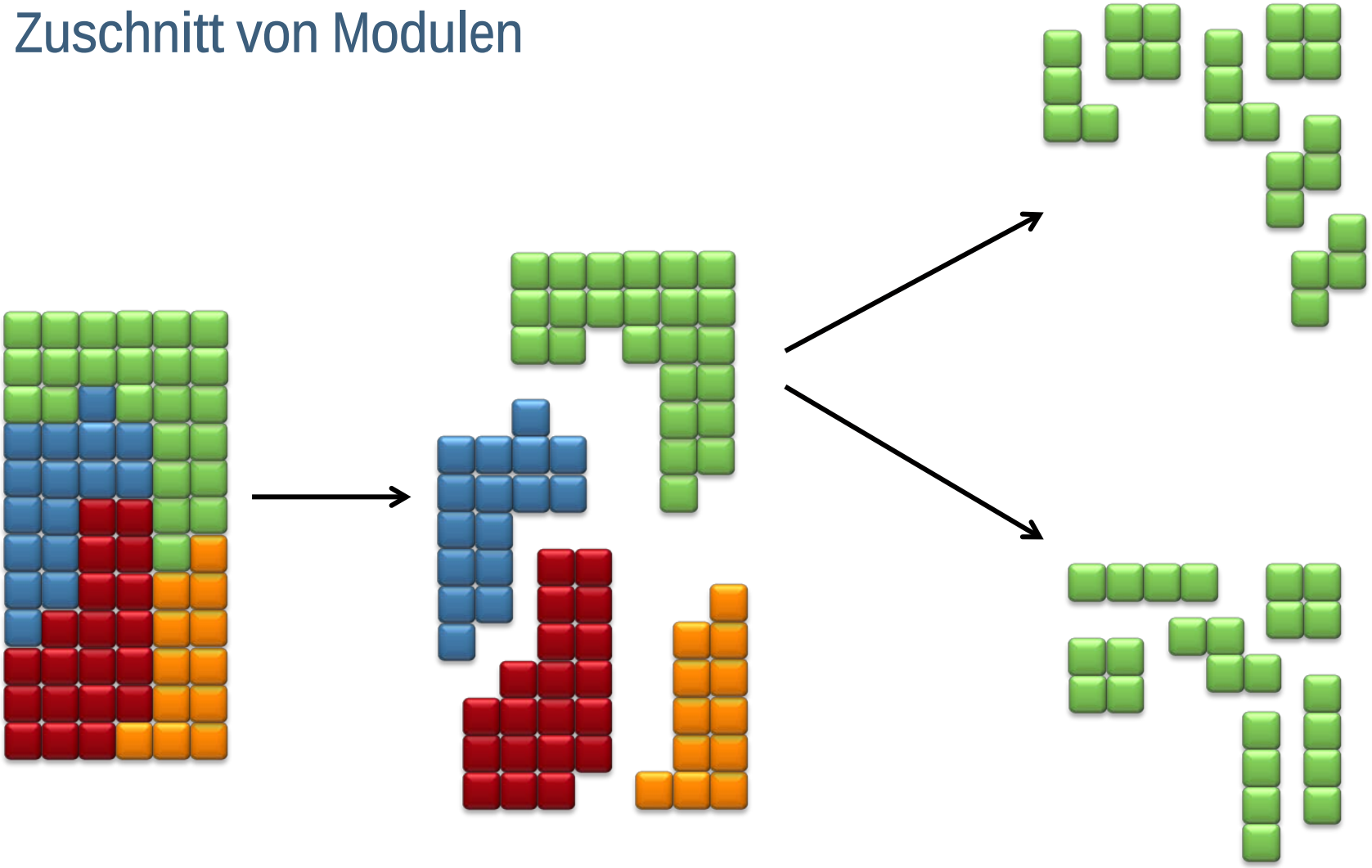
Modelle

Komponenten

Hinweis: nicht gemeint sind Template-Libraries, Numerik-Bibliotheken, usw.

Zuschnitt von Modulen

Zuschnitt von Modulen



Welche Wahl der Module und Schnittstellen ist „besser“?

Zuschnitt von Modulen

- Helikopter
- Soldat
  - Zielsucher + Launcher
  - Gehirn
- Flugkörper
  - Antrieb
  - Sprengladung
  - Suchkopf

nach Objekten

- sehen
- bewegen
  - auf Wegpunkten
  - nach Aerodynamik
- entscheiden

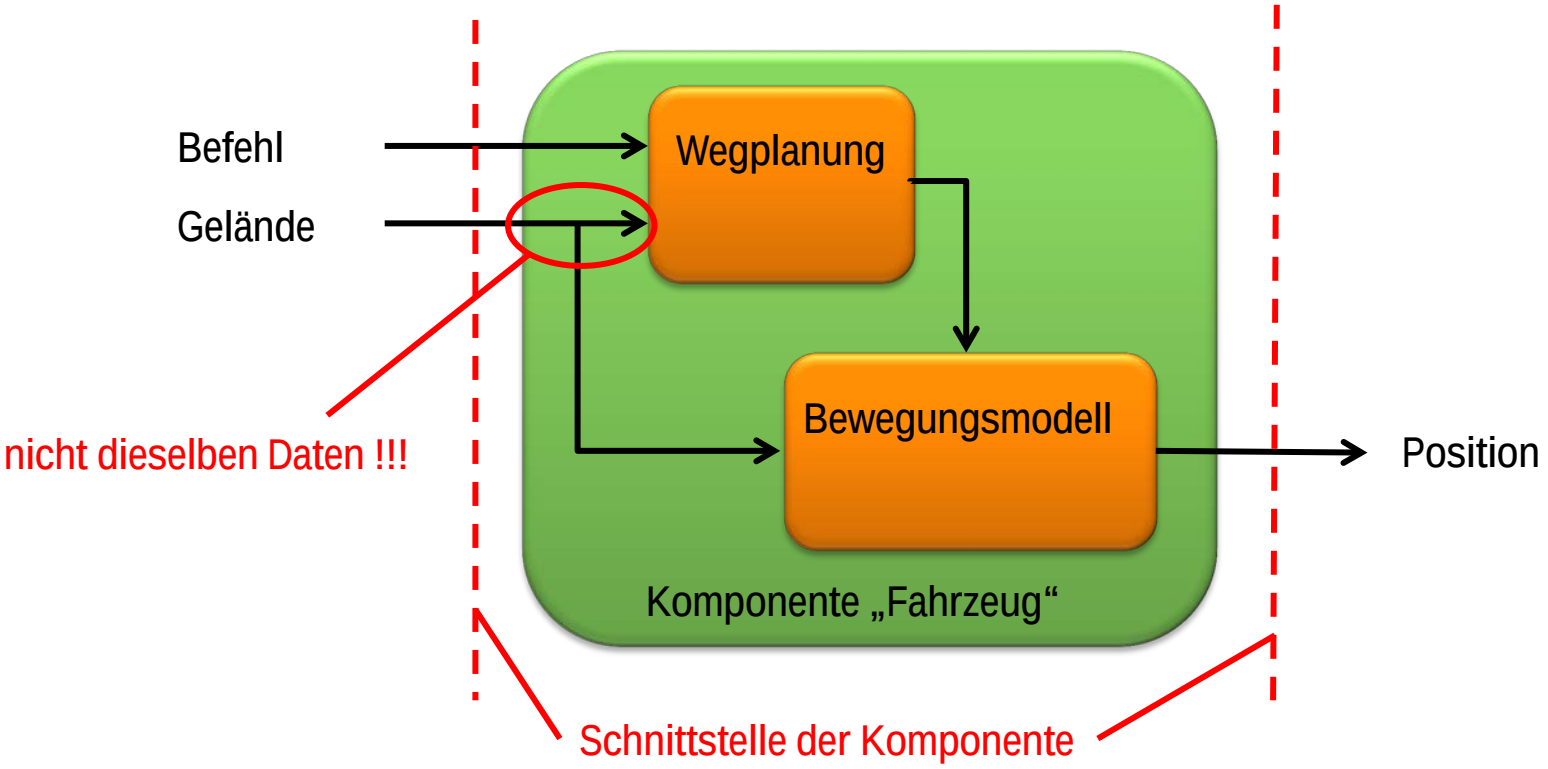
nach Funktionen

Zuschnitt von Modulen

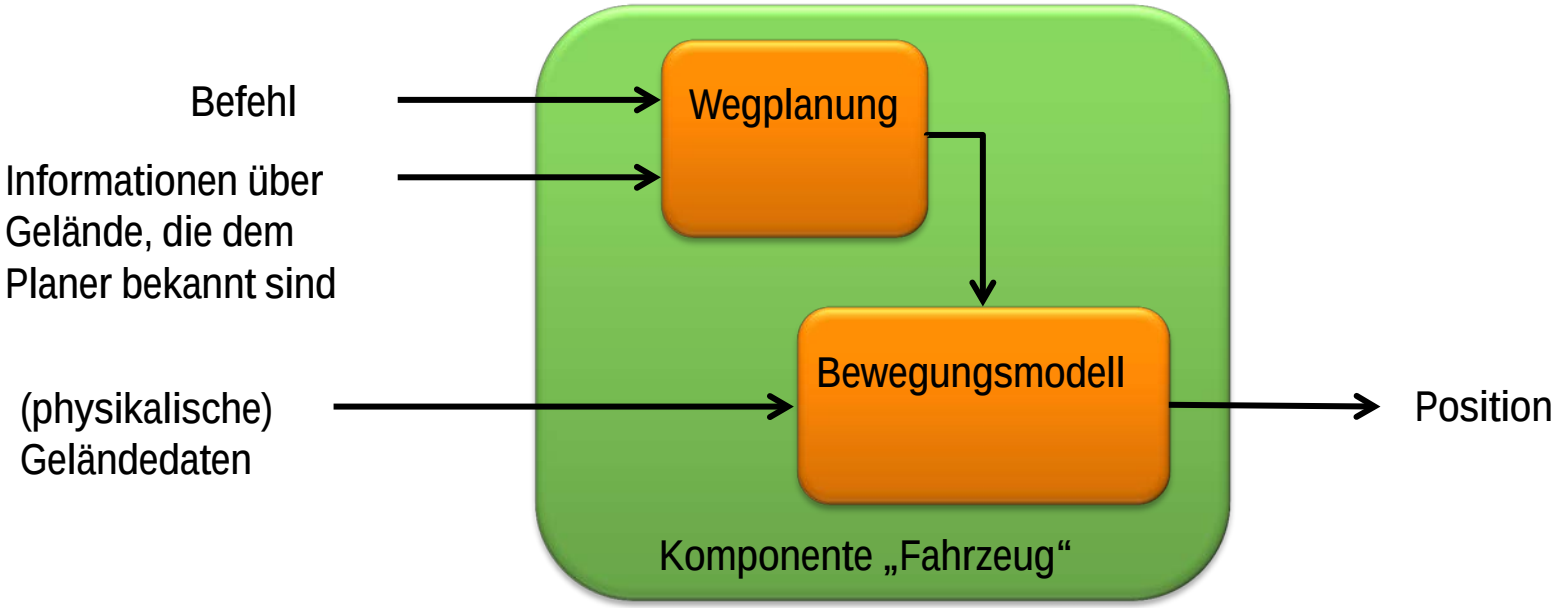
- Kombination der Module muss dieselbe Simulation ergeben
- Mischformen sind möglich und wahrscheinlich
- Vermutung: funktionale Modularisierung ist günstiger
  - Funktionen lassen sich leichter wiederverwenden als Objekte
    - Zustandsfreiheit
    - Nebenwirkungsfrei (die gesamte Wirkung erfolgt über die Schnittstelle, keine Veränderung an Objektvariablen)
    - vgl. funktionale Programmiersprachen
  - vgl. NATO Architecture Framework
    - NOV-5 Operational Activitiy Decomposition
    - NSV-5 System Function Decomposition
    - d.h. durchgängiger Bezug zur operationellen Realität und echten Systemfunktionen
  - Nachteile
    - Bruch mit klassischer Objektorientierung
    - schwerer vermittelbar

Querverweis: Services in VIntEL

Realitätsnahe Modellierung

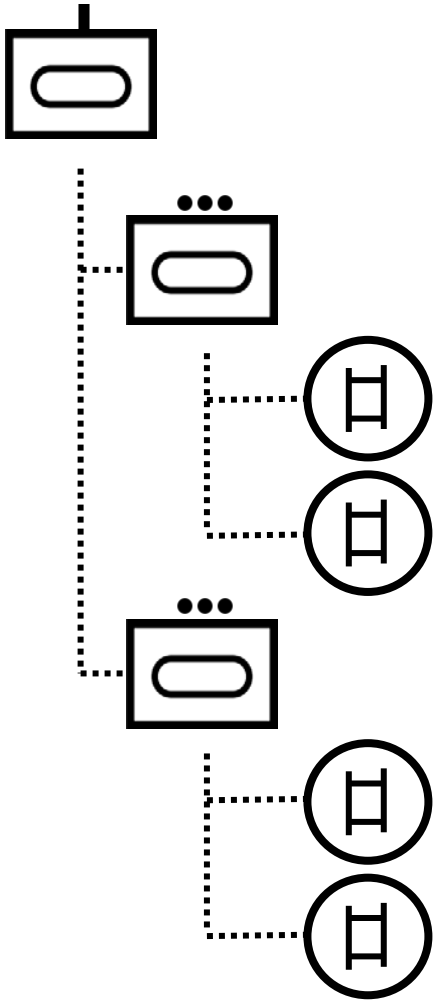


Realitätsnahe Modellierung

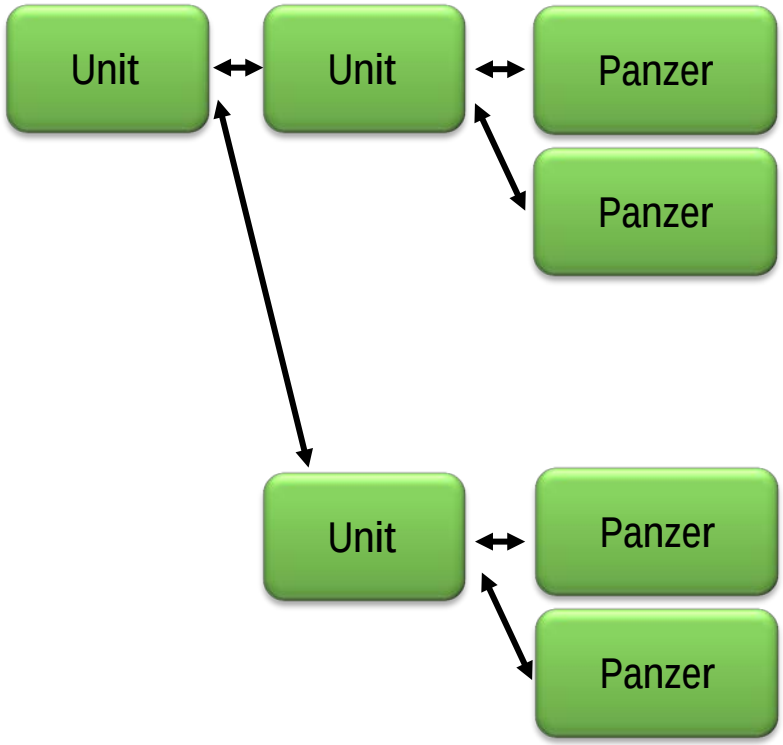


- kein (rein) semantisches Problem
- implizite Modellannahmen
- Schnittstelle nicht breit genug
- Wiederverwendbarkeit der Komponente eingeschränkt

Realitätsnahe Modellierung



- nur 2 Typen von Komponenten erforderlich



Realitätsnahe Modellierung

ABER

Unit

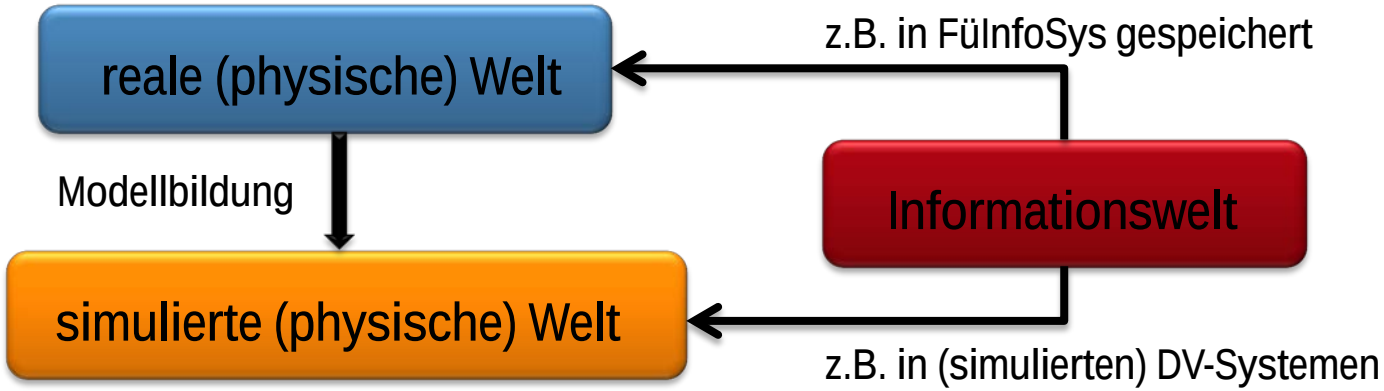
- Zusammensetzung jederzeit änderbar
- besteht nur aus „Wissen“ des Berichtsweges
- viele Möglichkeiten der Beschreibung, operationelle Standards (z.B. JC3IEDM)

Panzer

- Zusammensetzung fest durch Konstruktion bestimmt
- besteht aus Hardware-Komponenten
- keine Standards zur Beschreibung notwendig, sondern reales Vorbild, Konstruktion, Ingenieurswissen

- Units sind rein logische Objekte
- in der simulierten (physischen) Umgebung darf es (eigentlich) keine Units geben

Realitätsnahe Modellierung



- simulierte physikalische Daten und Objekte klar von (simulierten operationellen) Informationen und Informationsverarbeitung trennen
- Information kann erzeugt, geändert, dupliziert und gelöscht werden, physikalische Objekte nicht
- Schnittstellen so legen wie in der Realität
  - wenn die Schnittstellen der Module dort liegen, wo sie in der Realität liegen, dann haben sie dieselbe „Komponierbarkeit“
  - Hardware-in-the-Loop/Software-in-the-Loop leicht realisierbar

Dokumentation

häufig dokumentiert

- Metadaten
- Schnittstellenbeschreibung
- UML-Diagramme
- Freitext

was man noch braucht

- Semantik, Pragmatik, Dynamik, Konzeption
  - Was hat sich der Modellierer dabei gedacht
  - Warum ist das Modell so gebaut, wie es gebaut ist
  - Wie ist der Kontext, in dem das Modell arbeiten soll
  - Wie wird der Input verwendet, wie ist der Output zu interpretieren
- den Weg zum Modell (Modul, Komponente) nachvollziehbar machen

Dokumentation

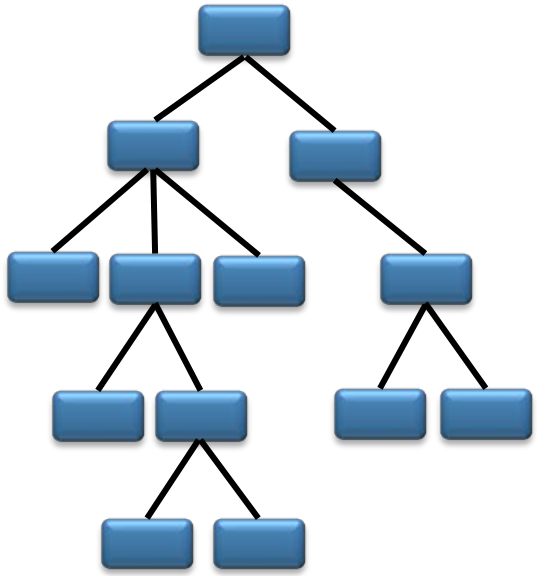
Ontologien sind hilfreich

- klare Beschreibung
  - Kontext
  - Semantik
  - auch über die beschriebene Komponente hinaus
- modular
  - komponierbar, ebenso wie das Modell selbst
- maschinenlesbar
  - W3C-Standards: OWL, RDF, RDFS,… auf Basis RDF-XML
  - verifizierbar
- beschreibend statt definierend
- reichhaltig
  - Basisontologien verfügbar (SUMO)
  - domänenspezifische (Basis-)Ontologien leicht ableitbar (z.B. aus JC3IEDM)
  - einfach um „private Anteile“ erweiterbar
  - kombiniert Klassen, Instanzen und Relationen

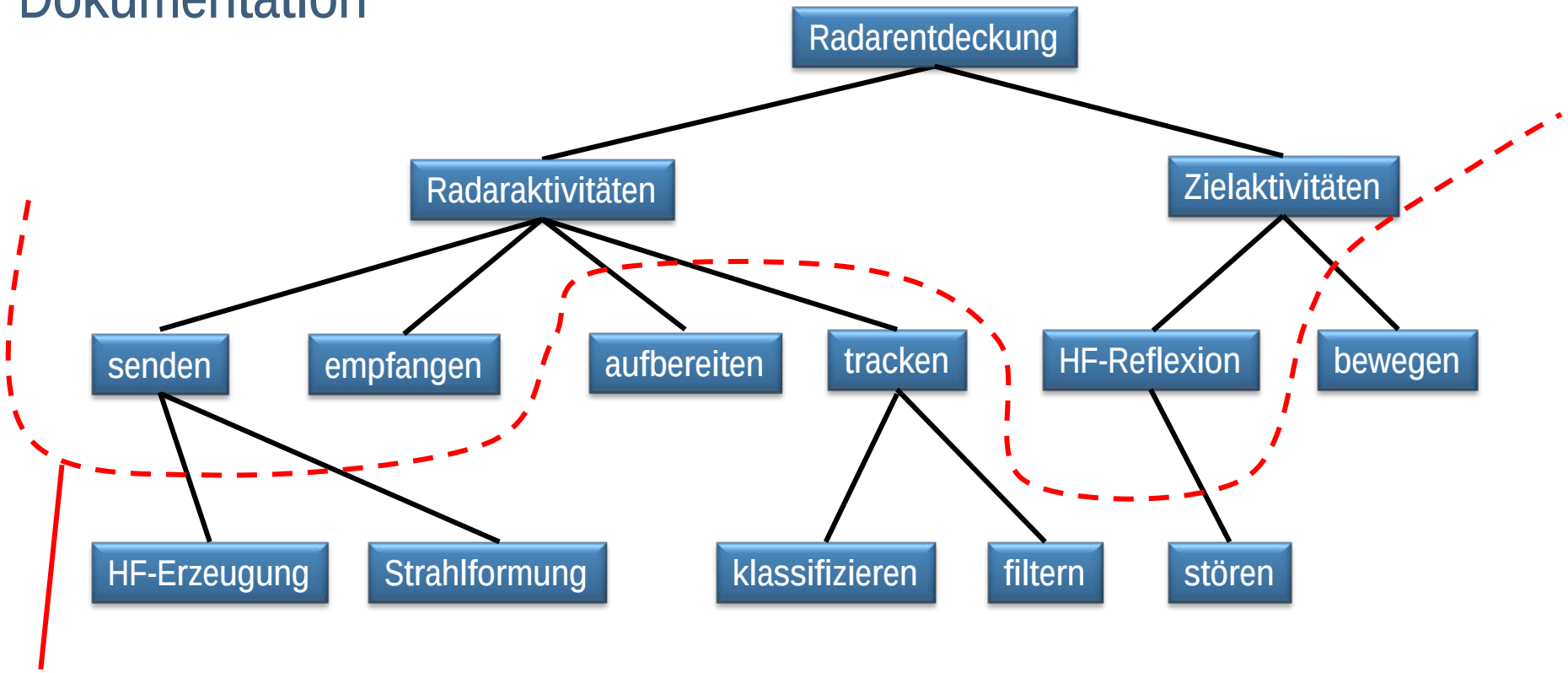
Dokumentation

Ontologien sind hilfreich, aber reichen nicht aus

- Dynamik
- innere Struktur von Modellen
  - Annahmen
  - Näherungen, Vereinfachungen
  - Interpretation der Inputs („innere“ Pragmatik)



Dokumentation

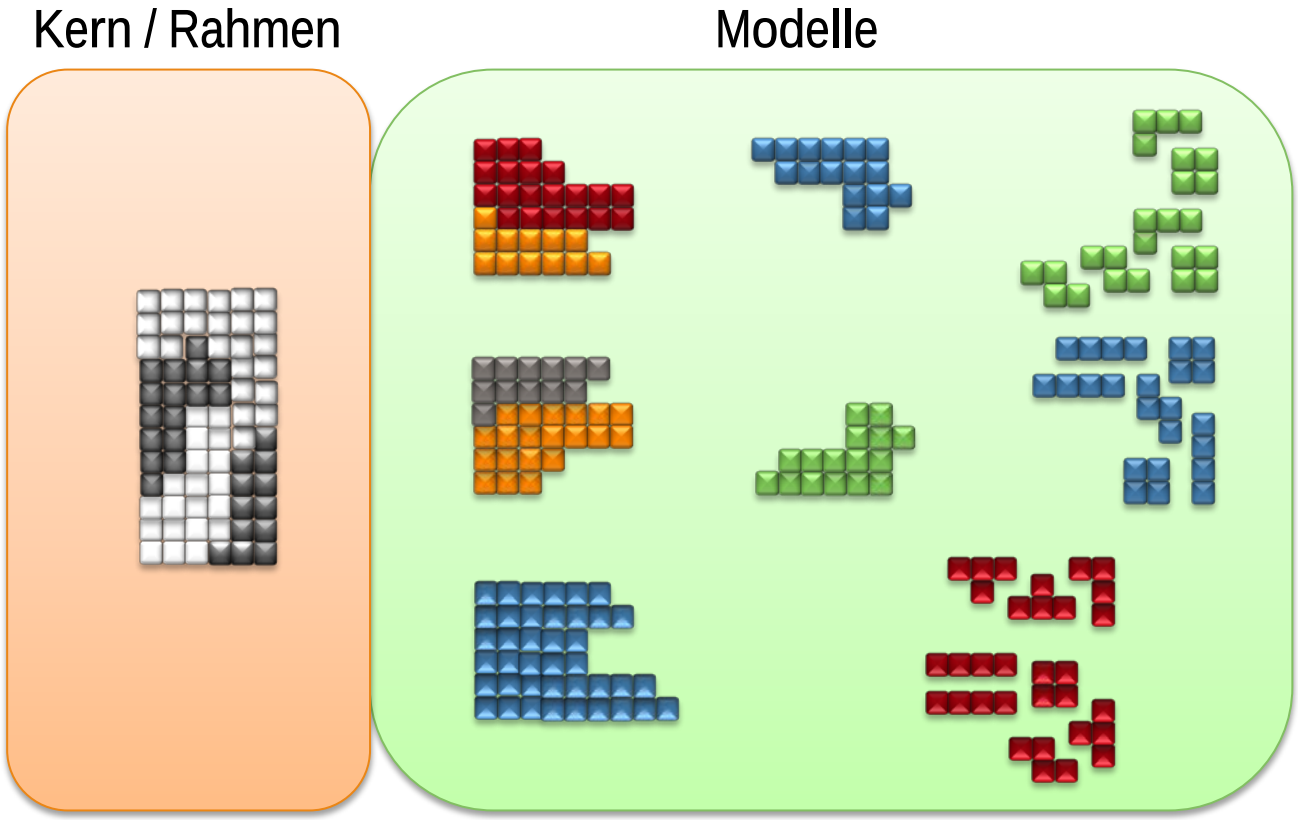


Modellgrenze entsteht durch

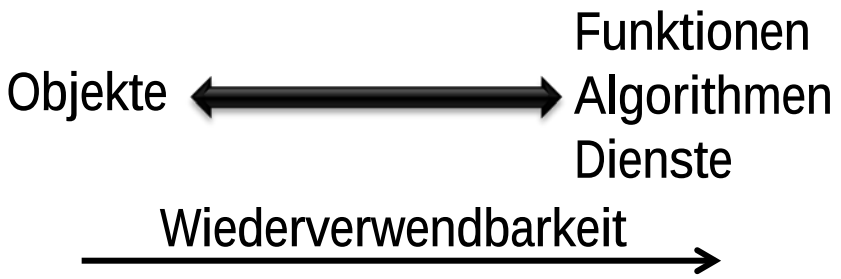
- Vereinfachen (z.B. Kennlinie statt Berechnung)
- Näherungen
- Weglassen

hilfsweise dokumentieren, was die nächst „genauere“ oder „ungenauere“ Modellierung wäre

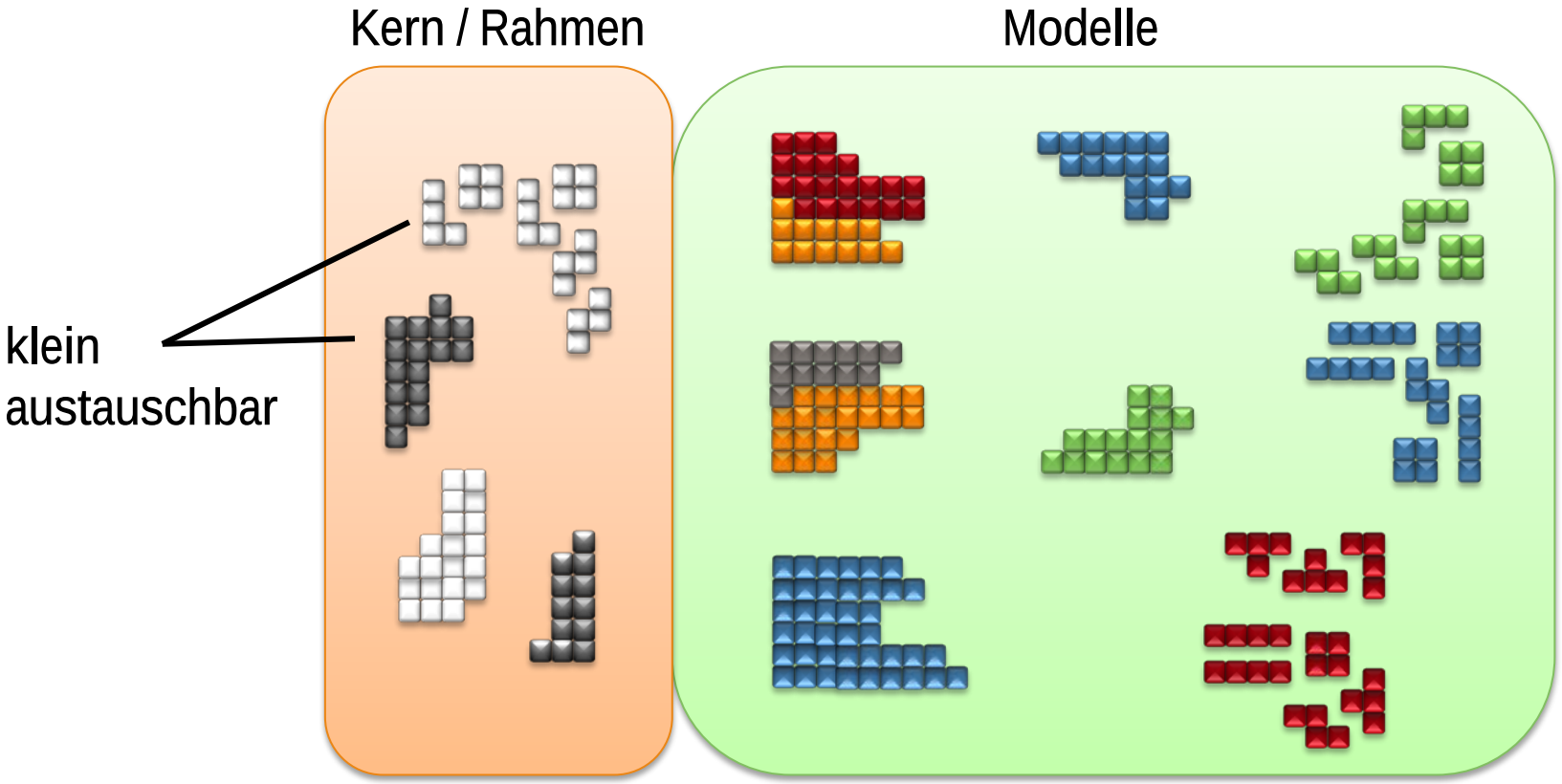
Architektur



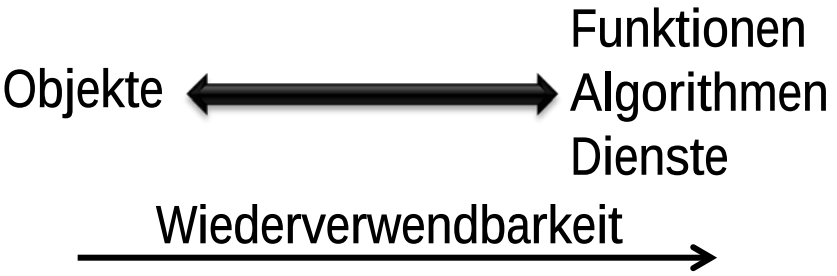
- wiederverwendbare und nicht-wiederverwendbare Teile trennen



Architektur



- wiederverwendbare und nicht-wiederverwendbare Teile trennen
- auch den „Kern“ modularisieren



Fazit

- Modularisieren ist einfach
- einen Mehrwert zu realisieren ist schwieriger
- es gibt (vermutlich) keinen Königsweg um den bestmöglichen Mehrwert von Komponenten zu realisieren
- es gibt viele Möglichkeiten, den Mehrwert von Komponenten zu verringern