

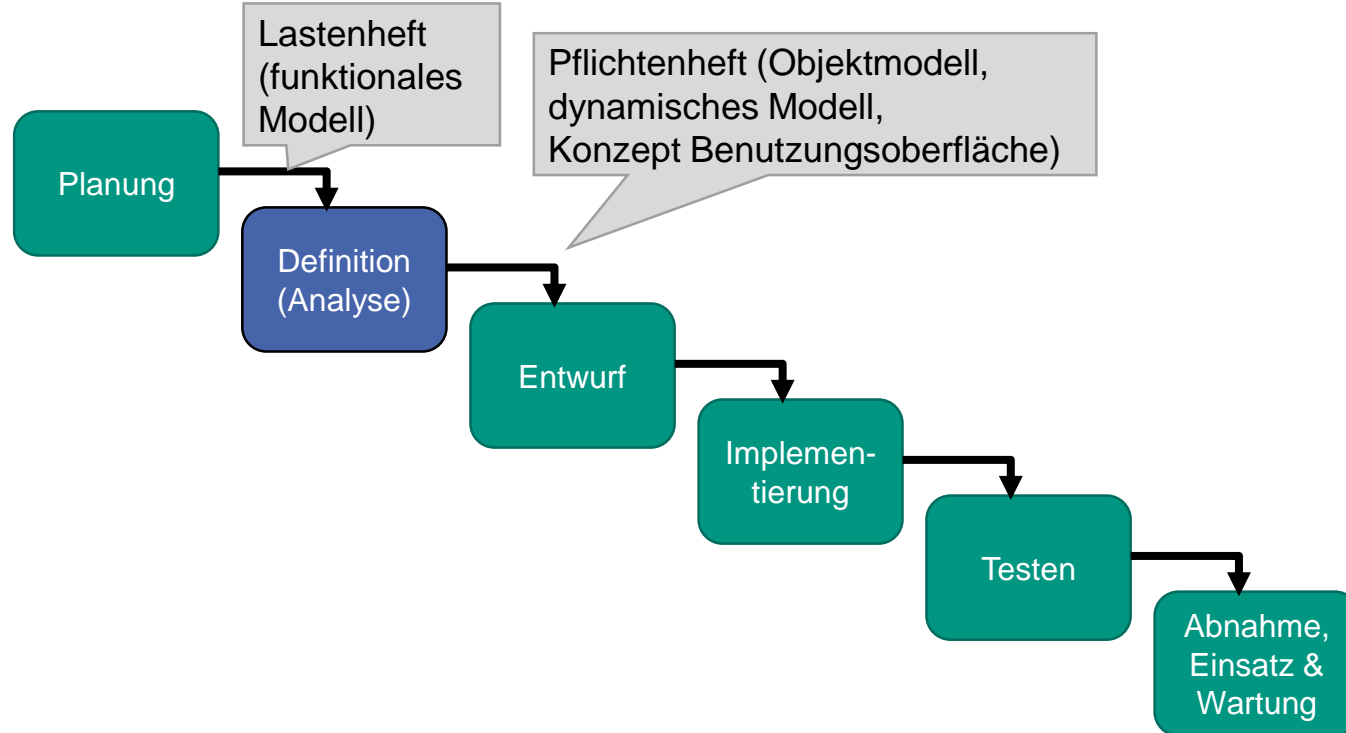
# Kapitel 2 – Die Definitionsphase

**SWT I – Sommersemester 2021**

**Walter F. Tichy, Christopher Gerking, Tobias Hey**



# Wo sind wir gerade?



# Lernziele

- Die Definitionsphase erklären und ihre Rolle im Entwicklungsprozess definieren
- Objektmodelle und dynamische Modelle erstellen können (UML)
- Für vorgegebene Aufgabenstellungen ein Pflichtenheft entsprechend dem beschriebenen Pflichtenheft-Schema erstellen können.

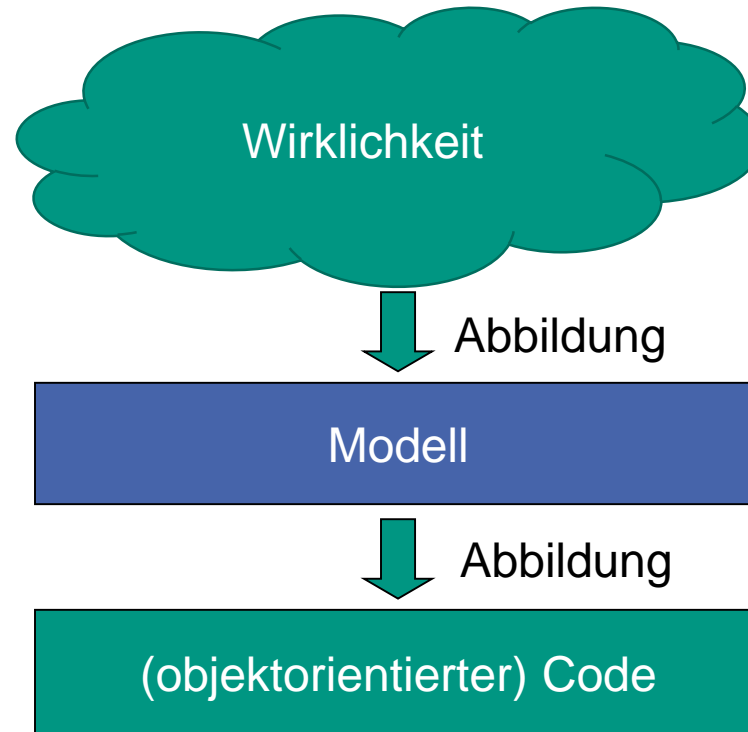
# Definitions- oder Analysephase

- In der Definitionsphase **entsteht das Pflichtenheft**
- Das Pflichtenheft **definiert** („modelliert“) das zu erstellende **System** (oder die Änderungen an einem existierenden System) so **vollständig und exakt**, dass Entwickler das System implementieren können, ohne nachfragen oder raten zu müssen, was zu implementieren ist.
- Das Pflichtenheft beschreibt nicht, wie, sondern nur was zu implementieren ist. (z.B. werden Algorithmen und Datenstrukturen nicht beschrieben).
- Das Pflichtenheft ist eine **Verfeinerung** des Lastenheftes.

# Modellierung

- Das Pflichtenheft liefert ein Modell des zu implementierenden Systems.
- Modellarten:
  - Funktionales Modell (aus dem Lastenheft)
    - Szenarien und Anwendungsfall-Diagramme (scenarios, use case diagrams)
  - Objektmodell
    - Klassen- und Objektdiagramme (class and object diagrams)
  - Dynamisches Modell
    - Sequenzdiagramme (sequence diagrams)
    - Zustandsdiagramme (state chart diagrams)
    - Aktivitätsdiagramme (activity diagrams)

# Einordnung



# Modell und Realität

## ■ Realität R

- Reale Dinge, Personen, Konzepte, etc.,
- Abläufe, die eine gewisse Zeit brauchen,
- Beziehungen zwischen Dingen, Personen, Konzepten

## ■ Modell M: Abstraktion von existierenden oder imaginären

- Dingen, Personen, Konzepte, etc.,
- Abläufen,
- Beziehungen dazwischen

# Wozu Modelle?

## ■ Wir benutzen Modelle...

- Um von Details der Realität zu **abstrahieren**, dadurch können wir über die komplexe Realität schlussfolgern, indem wir einfache, definierte Deduktionsschritte auf dem Modell vornehmen.
- Um **Erkenntnisse** über Vergangenheit oder Gegenwart zu erhalten.
- Um **Vorhersagen** über die Zukunft zu treffen.

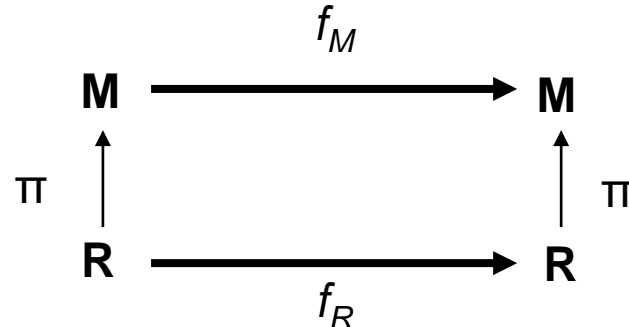
## ■ Speziell Softwaremodelle

- Wir verwenden Modelle (der Software) um ein Verständnis für die mögliche zukünftige Realität (die Software) zu entwickeln.
- In anderen Worten: Wir verwenden Modelle um zu sehen was wir bekommen würden, wenn wir es bauen würden.



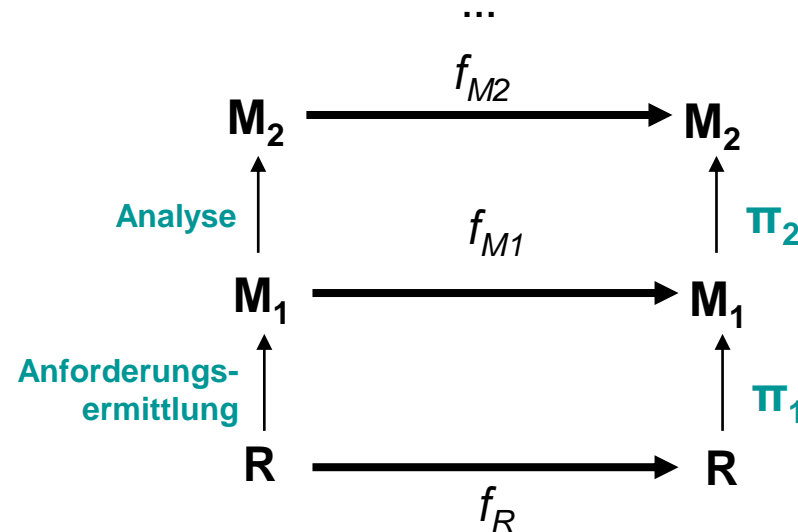
# Was ist ein „gutes“ Modell?

- Beziehungen, die in der Realität  $R$  gültig sind, sind auch im Modell  $M$  gültig.
  - $\pi$  : Abbildung der Realität  $R$  auf ein Modell  $M$  (Abstraktion)
  - $f_M$ : Beziehungen zwischen Abstraktionen in  $M$
  - $f_R$ : äquivalente Beziehungen zwischen echten Dingen in  $R$
- In einem guten Modell ist das folgende Diagramm kommutativ:



# Metamodellierung: Modelle von Modellen von Modellen...

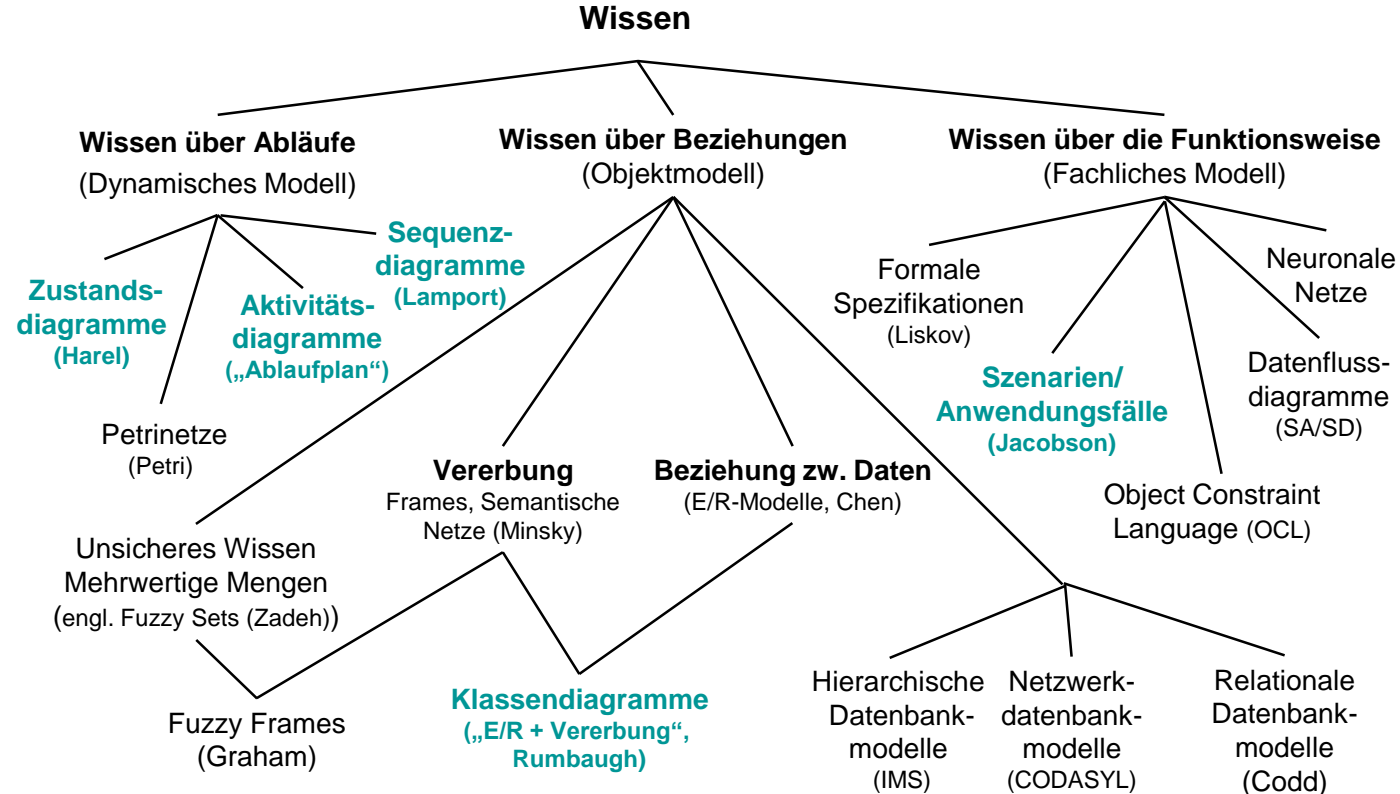
- Modellierung ist relativ
- Wir können uns ein Modell als Realität vorstellen und dafür ein neues Modell (mit weiteren Abstraktionen) definieren
  - Softwareentwicklung heißt Modelltransformation



# „Realitäten“ des Softwareingenieurs

- Softwareingenieure können verschiedene „Realitäten“ modellieren und realisieren:
  - Ein **existierendes System** (physikalisch, technisch, sozial oder Softwaresystem) modellieren und eine Realisierung bauen
    - Das „Softwaresystem“ stellt einen wichtigen Spezialfall dar: Wir sprechen von Altlasten (engl. „Legacy-System“)
  - Eine **Idee** ohne entsprechendes Gegenstück in der Realität modellieren und realisieren
    - Ein visionäres Szenario oder eine Kundenanforderung
    - In solchen Fällen wird häufig nur ein Teil des Originalmodells gebaut, weil der Rest z.B. zu kompliziert, zu teuer oder unnütz ist

# Wie modellieren wir komplexe Systeme?



# Modellierungsarten auf Englisch

