

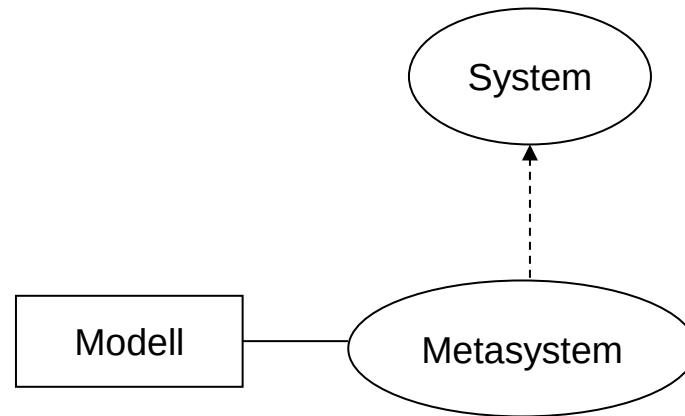
Modell

Christian Silberbauer

Modellbegriff

- Ein Modell ist die Abbildung einer Systemebene.
- Es beschreibt den Zustand einer Systemebene.
- Ein Modell gibt Vorgaben für dessen Verhalten.
- Die Kommunikation eines Systems wird durch sein Metasystem auf Basis seines Modells realisiert.

Modell vs. System



- In der Programmierung bringt z.B. ein Interpreter (Metasystem) eine Anwendung (System) zur Ausführung durch Interpretation des Programms (Modell).

Modell als Sprache

- Modelle sind Sprachen.
- Z.B. ist ein Computerprogramm die Sprache der Anwendung.

Metamodell

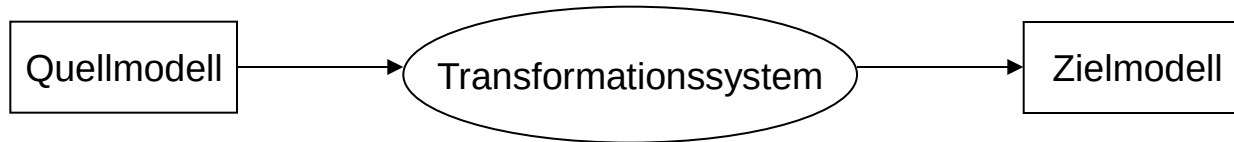
- Ein Metamodell ist ein Modell zur Beschreibung von Modellen.
- Ein Metamodell ist demnach *nicht* ein Modell eines Metasystems.
- Z.B. Programmiersprachen sind Metamodelle.
- Metamodelle dienen zur Modellvalidierung und zur Modelltransformation.

Metamodelle in der Programmierung

Metaebene	Name	Beispiel
M3	Metametamodell	MOF
M2	Metamodell	UML-Superstructure, OCL
M1	Modell	Klassendiagramm, Programmcode

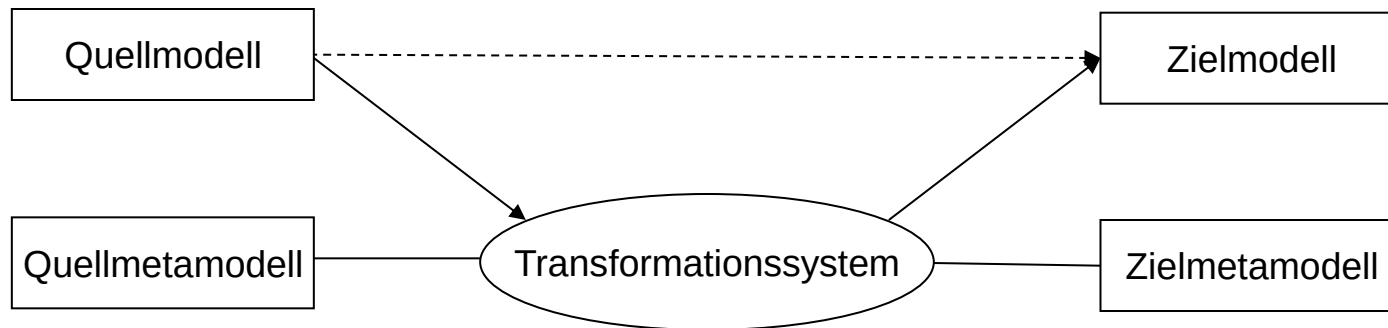
- Aus M1 ist schließlich das System bzw. die Anwendung hervorgehend (M0).

Modelltransformation



- Ein Transformationssystem überführt ein Quellmodell in ein Zielmodell.
- Beispielsweise überführt ein Compiler Quellcode in Maschinencode.

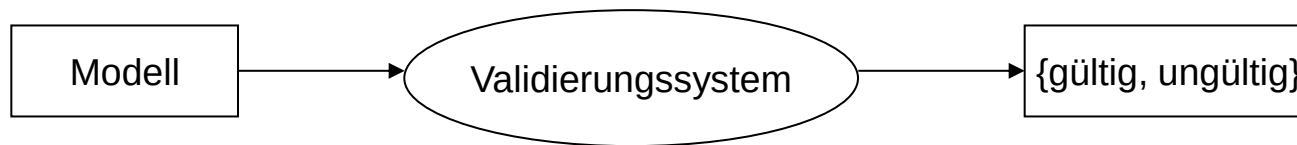
Modelltransformation



- Basieren Quellmodell und Zielmodell auf Metamodellen, erlaubt dies ein generischeres Transformationssystem.
- Z.B. kann hierfür DOM verwendet werden (<https://www.w3.org/TR/WD-DOM/>).

Modellvalidierung

- Eine Modelltransformation setzt eine Validierung des Quellmodells voraus.
- Modellvalidierung ist die Prüfung der Gültigkeit des Modells.



- Ein gültiges Quellmodell ermöglicht ein einfacheres Transformationssystem, da Fehler unberücksichtigt bleiben können.

Modellvalidierung

- Ein Validierungssystem kann als Transformationssystem betrachtet werden, dessen Zielmodell die Aussage über die Gültigkeit des Quellmodells ist.

Validierung

- Frühe Validierung: Validierung auf Basis des Quellmodells
- Späte Validierung: Validierung durch das System
- Späte Validierung ist notwendig.
- Ein stabiles System kann nicht stabil *sein*, sondern es *tendiert* zur Stabilität.

Beispiel aus der Programmierung

- Bei der Programmierung eines Softwaresystems können Fehler, die die interne Kommunikation dessen betreffen durch Compilerfehler erkannt werden.
- Starke, statische Typsicherheit verhilft dabei mehr Fehler zur Übersetzungszeit zu erkennen.

Beispiel aus der Programmierung

- Aus der Java-Spezifikation, Chapter 4:
(siehe <http://docs.oracle.com/javase/specs/jls/se8/html/jls-4.html>)

“The Java programming language is a *statically typed* language, which means that every variable and every expression has a type that is known at compile time.

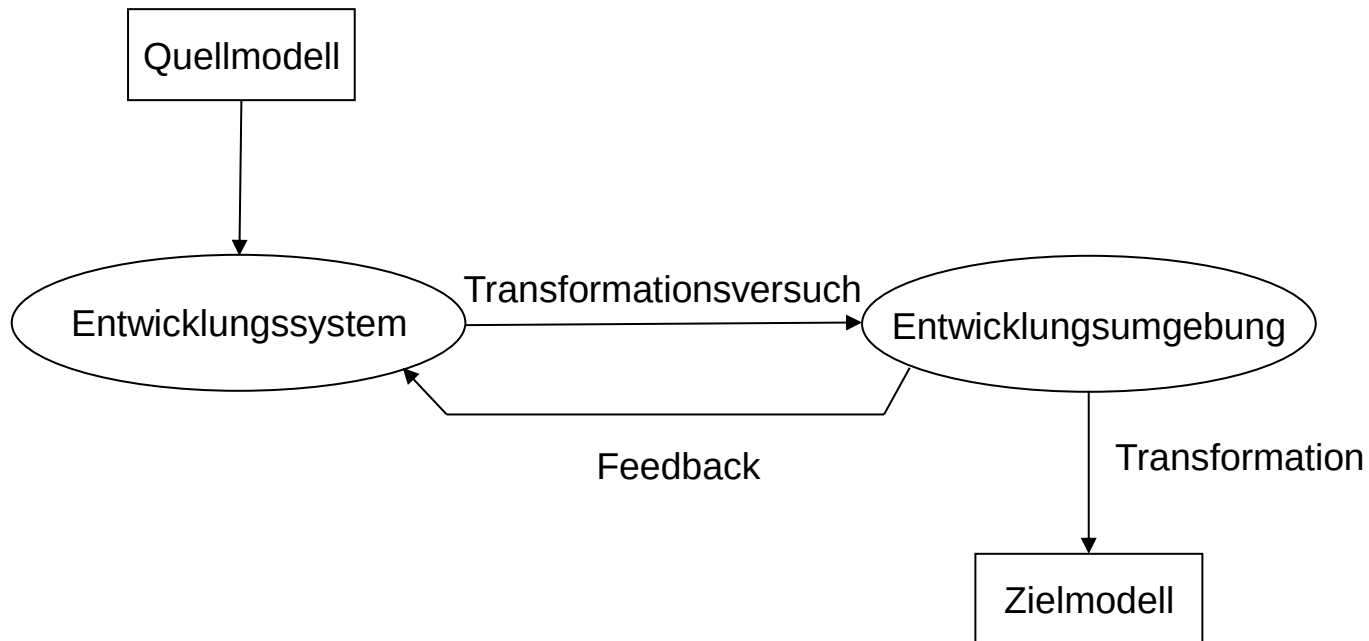
The Java programming language is also a *strongly typed* language, because types limit the values that a variable (§4.12) can hold or that an expression can produce, limit the operations supported on those values, and determine the meaning of the operations. Strong static typing helps detect errors at compile time.”

Beispiel aus der Programmierung

- Fehler bei externer Kommunikation eines Anwendungssystems können hingegen nicht zur Übersetzungszeit erkannt werden.
- Typische Beispiele hierfür:
 - Dateizugriff
 - Datenbankzugriff
 - Netzwerkkommunikation
 - Benutzereingaben
- Hierfür bieten Java und andere Programmiersprachen das Konzept der **Exceptions**.

Modellierung

- Modellierung bezeichnet den Prozess der Modellbildung.
- Er fokussiert passives Lernen.



Modellierung

- Sowohl Transformation als auch Validierung wird von den beiden Systemen Entwicklungssystem und Entwicklungsumgebung übernommen.
- In der Programmierung ist das Entwicklungssystem z.B. der Programmierer und die Entwicklungsumgebung z.B. die IDE.

Modellierung

- Weitere Beispiele der Modellierung:
 - Ein Chirurg operiert.
 - Ein Maurer baut ein Haus.
 - Ein Psychotherapeut therapiert.

