Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Quo vadis, Modellierung?

Prof. Dr. Holger Giese

Fachgebiet Systemanalyse und Modellierung

Übersicht

I. Was ist Modellierung?

II. Modellierung in der Informatik

III. Aktuelle Herausforderungen

IV. Die Zukunft der Modellierung

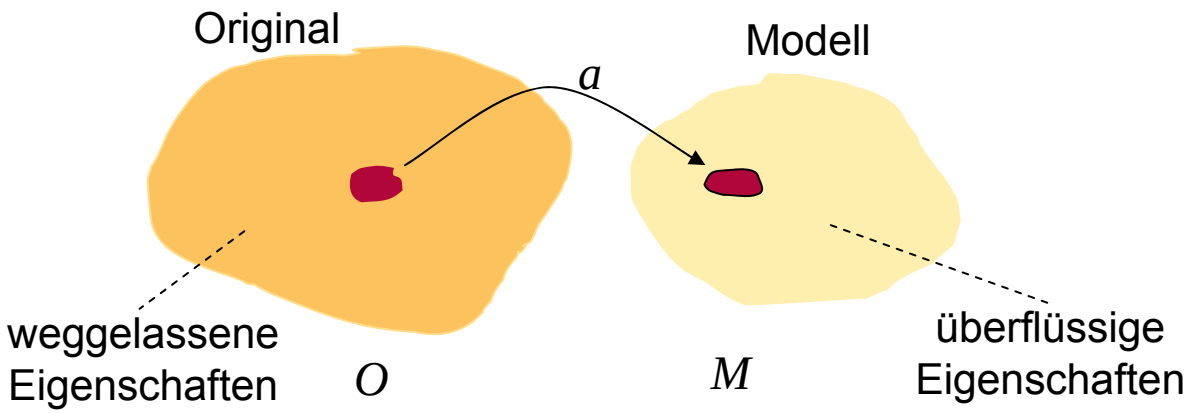
V. Ausblick & Arbeiten am Fachgebiet

I. Was ist Modellierung?

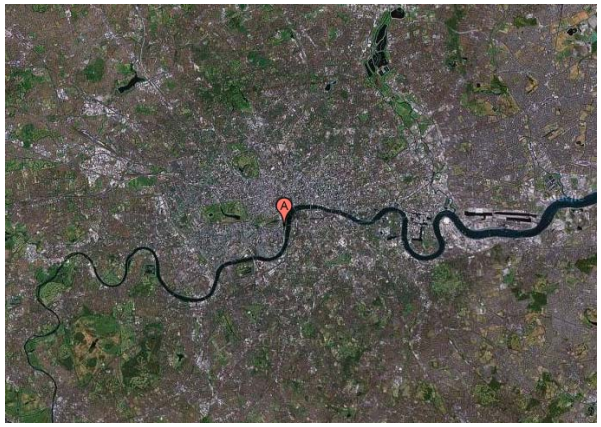
Herbert Stachowiak; *Allgemeine Modelltheorie*, Springer-Verlag, Wien 1973.

Modelle sind im Allgemeinen abstrakte Abbilder oder Vorbilder zu konkreten oder abstrakten Originalen.

- **Abbildung** eines Originals: es gibt immer einen Bezugspunkt
  - Eine Funktion a die dem Original O ein Modell M zuordnet (**Abstraktion**).
  - Eine nicht eindeutige Rückabbildung i die einem Modell M ein Original O zuordnet (**Interpretation**).
- **Reduktion:** nicht alle **Eigenschaften** werden wiedergegeben
- **Pragmatik:** Modell kann das Original für einen **Zweck** ersetzten



Beispiel: Karte der Londoner U-Bahn



London

a

Karte der U-Bahn von London

Abbildung:

- Bezug zur „echten“ U-Bahn

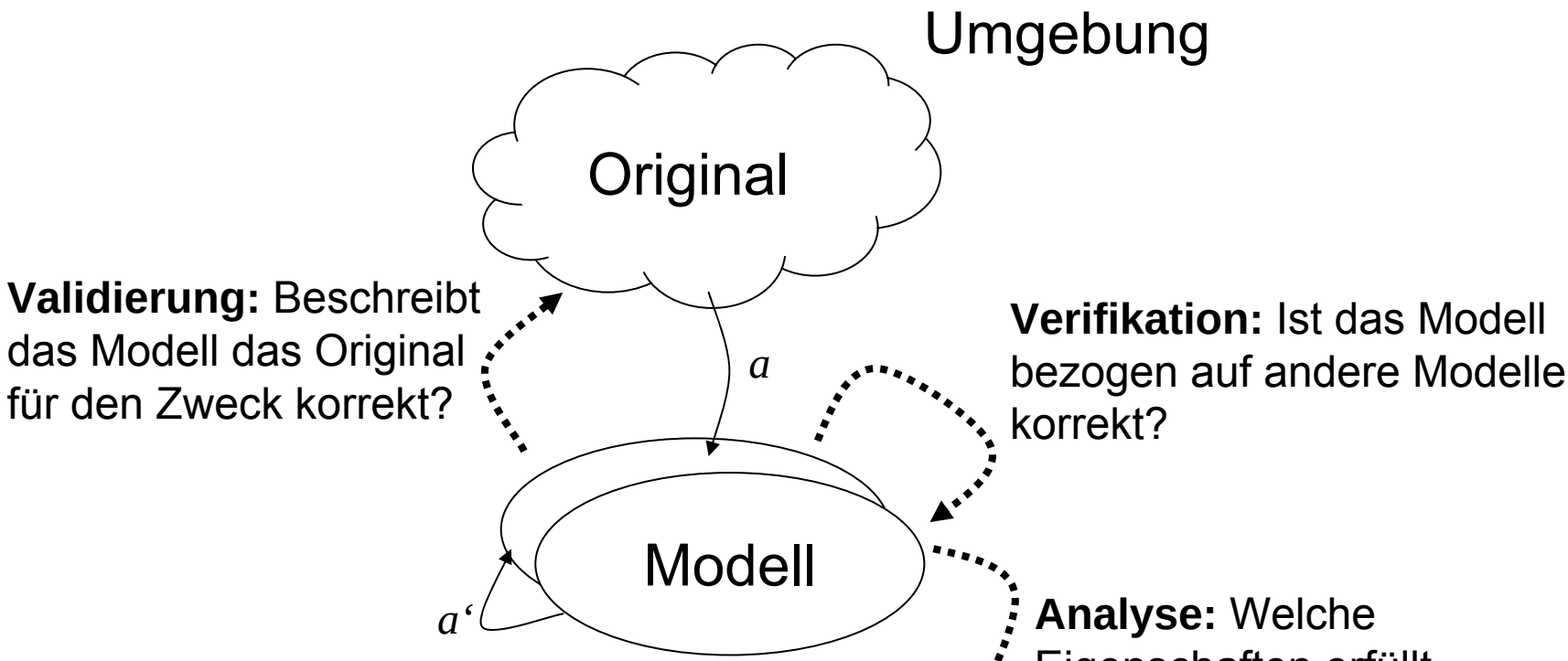
Reduktion:

- Weggelassene Eigenschaften:
  - Korrekte geometrische Distanzen/Lage
  - Höheninformationen, Straßen, …
- Überflüssige Eigenschaften:
  - Geometrische Distanzen/Lage sind nicht korrekt

Pragmatik:

- Navigation in der U-Bahn (aber nicht darüber hinaus!)

Validierung, Verifikation & Analyse



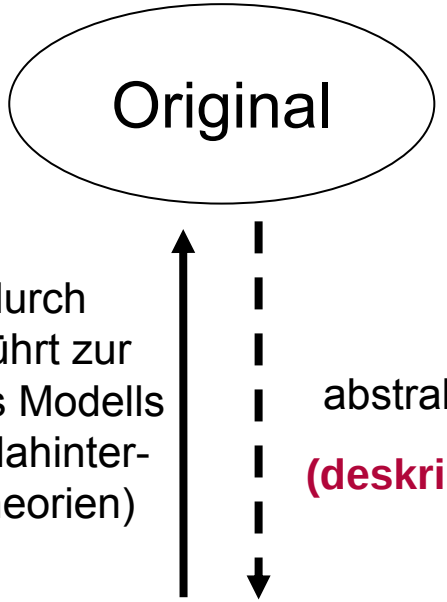
Validierung: Beschreibt das Modell das Original für den Zweck korrekt?

Verifikation: Ist das Modell bezogen auf andere Modelle korrekt?

Analyse: Welche Eigenschaften erfüllt das Modell?

Durch den Zweck des Modells beschränkt!

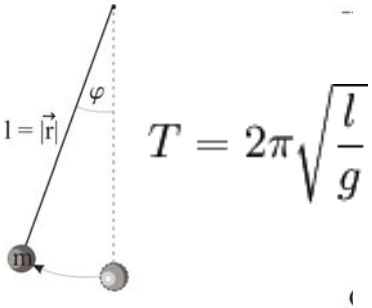
Modelle in den Naturwissenschaften



Vergleich durch Experiment führt zur Validierung des Modells (und ggf. der dahinter-stehenden Theorien)

abstrahiert

(deskriptiv)



$$T = 2\pi\sqrt{\frac{l}{g}}$$

Konstruktion:

- Modell wird in der Regel auf Basis anerkannter wissenschaftlicher Theorien hergeleitet

Nutzen:

- Ein valides Modell ermöglicht Vorhersagen für ein System
- Validität des Modells bestätigt Theorie

Analyse ermöglicht Vorhersagen

Modelle bei den Ingenieuren

realisiert?

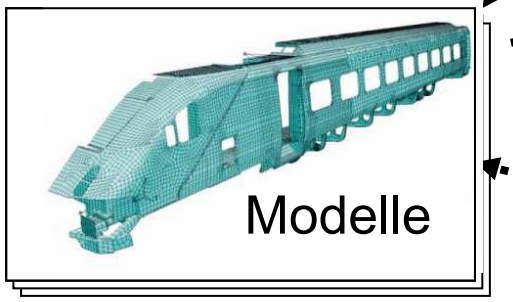
System-anforderungen

Konstruktion

abstrahiert

(präskriptiv)

abstrahiert (deskriptiv)



Analysen bestätigen die Erfüllung der Anforderungen

Konstruktion:

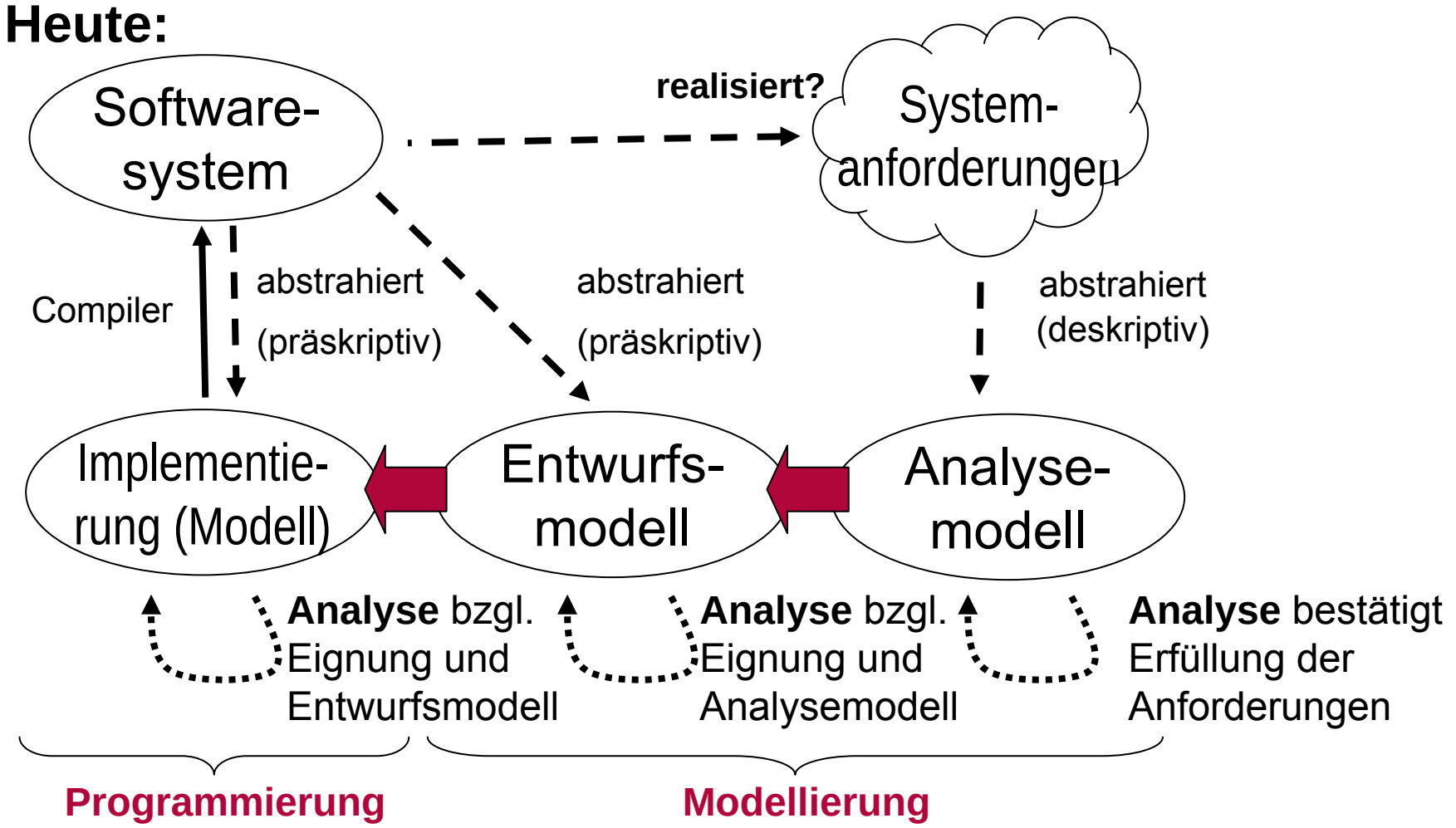
- Modell wird auf Basis der Systemanforderungen und anerkannter wissenschaftlicher Theorien hergeleitet

Nutzen:

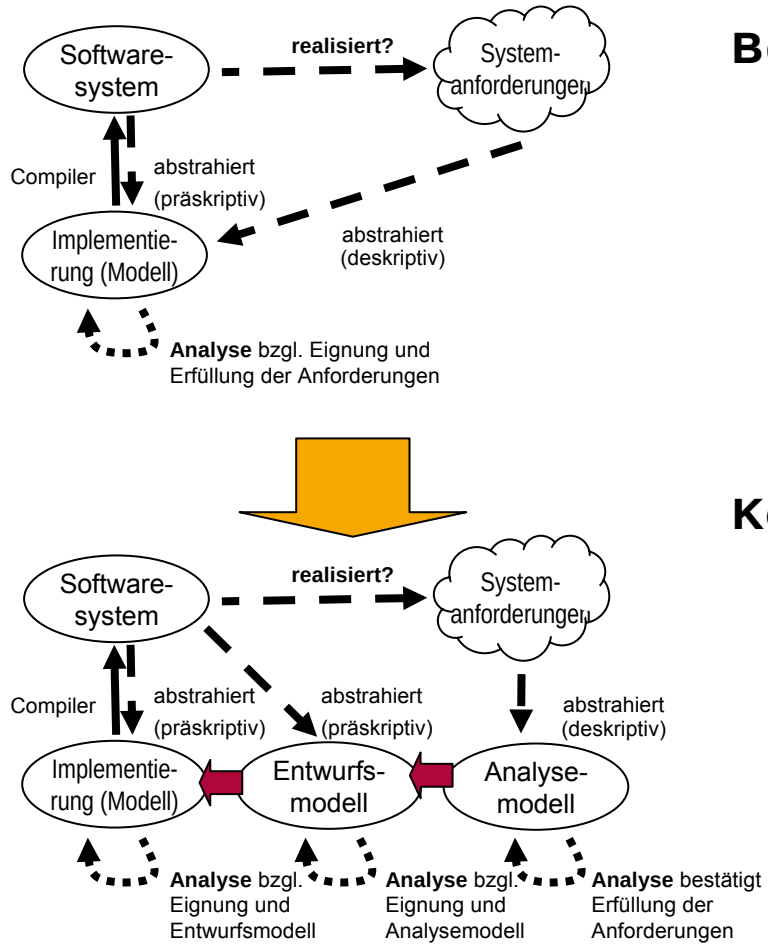
- Erfüllung der Anforderungen kann kostengünstiger am Modell gezeigt werden, da
  - Analyse am Modell einfacher ist und
  - Original oft erst später analysiert werden kann.

II. Modellierung in der Informatik

Beobachtungen: Original und Modelle sind **immateriell**



Wieso nicht nur Programmierung?



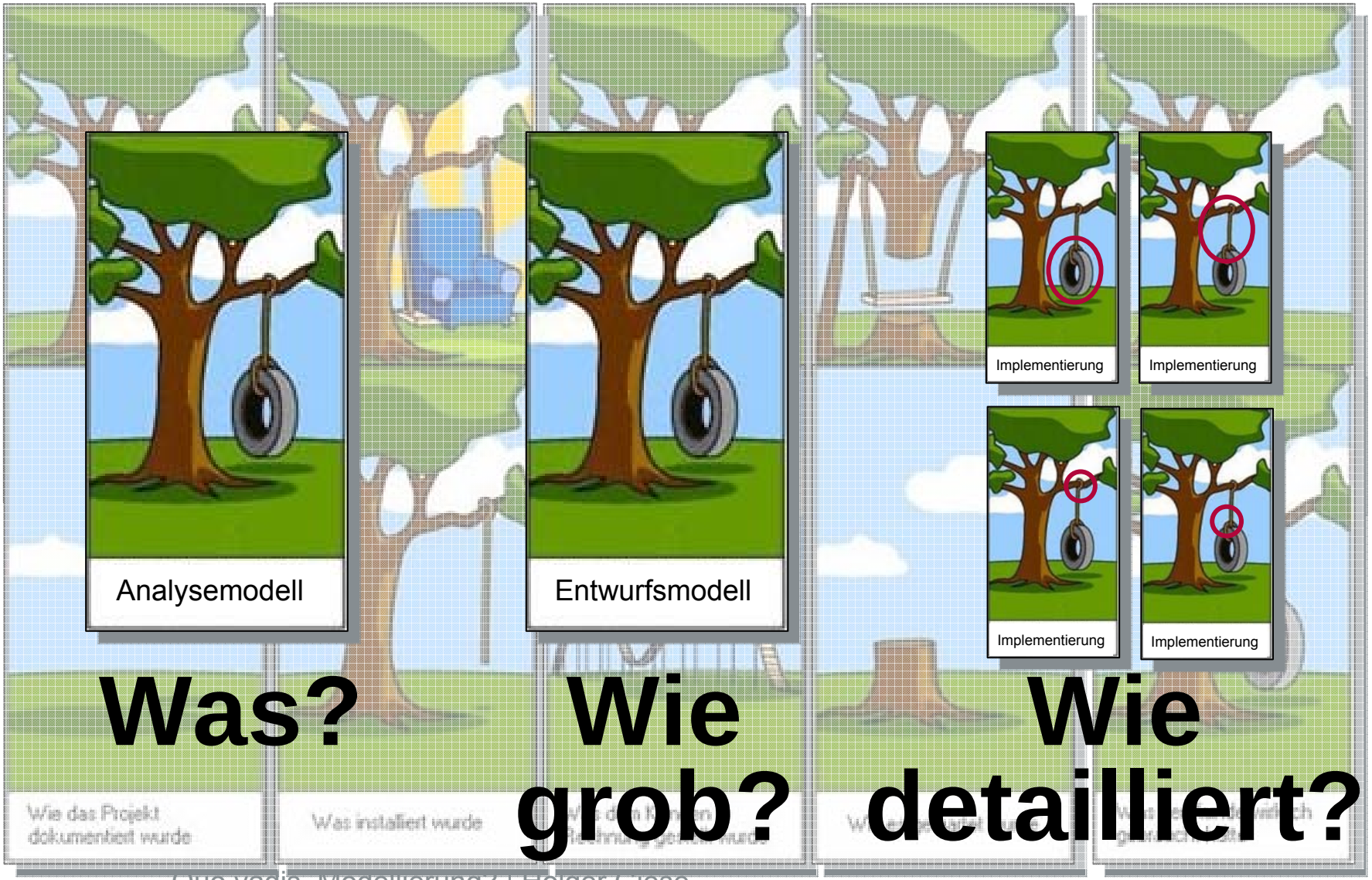
Beobachtungen:

- Nach der allgemeinen Definition findet **immer** Modellierung statt!
- Ein einziger Schritt ist aber häufig zu schwierig (hohe Komplexität), da
  - **Konsensbildung** immer nötig und
  - **Arbeitsteilung** häufig nötig

Konsequenzen:

- **Programmierung reicht nicht!**
- **Zusätzliches Analysemodell:** Zuerst verstehen was der Kunde wirklich will (**Konsensbildung**)!
- **Zusätzliches Entwurfsmodell:** Detaillierte Lösung als Grundlage für die **Arbeitsteilung** zuerst auf höherer Abstraktionsebene planen!

Konsensbildung & Arbeitsteilung



III. Aktuelle Herausforderungen

(1) Ständige Evolution statt Revolution

(2) Agilität statt Vorhersagbarkeit

(3) Exponentiell wachsende Komplexität

Bemerkung:

- Je nach **Anwendungsdomäne** sind die Herausforderungen aber verschieden relevant!

(1) Ständige Evolution statt Revolution

Beobachtung: Softwaresysteme werden oft „unersetzlich”

Erklärung: Software beinhaltet oft „Wissen” der Organisation

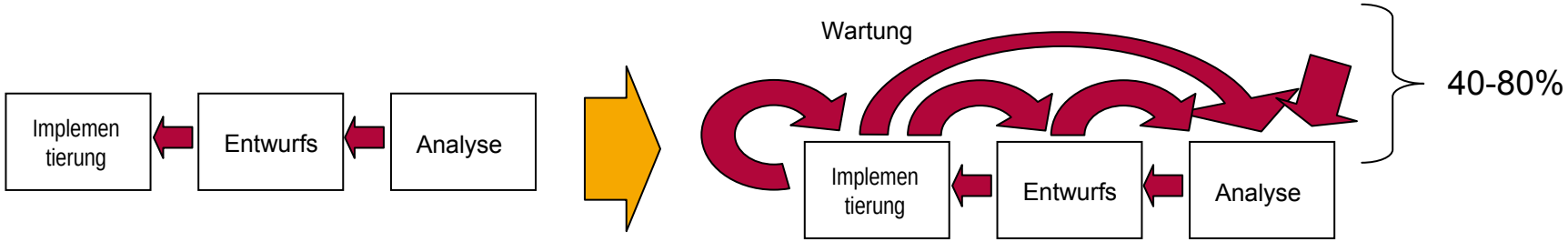
Beobachtung: “Programme altern”, obwohl sie digital sind.

Erklärung: “Lehman’s Laws of Software Evolution”

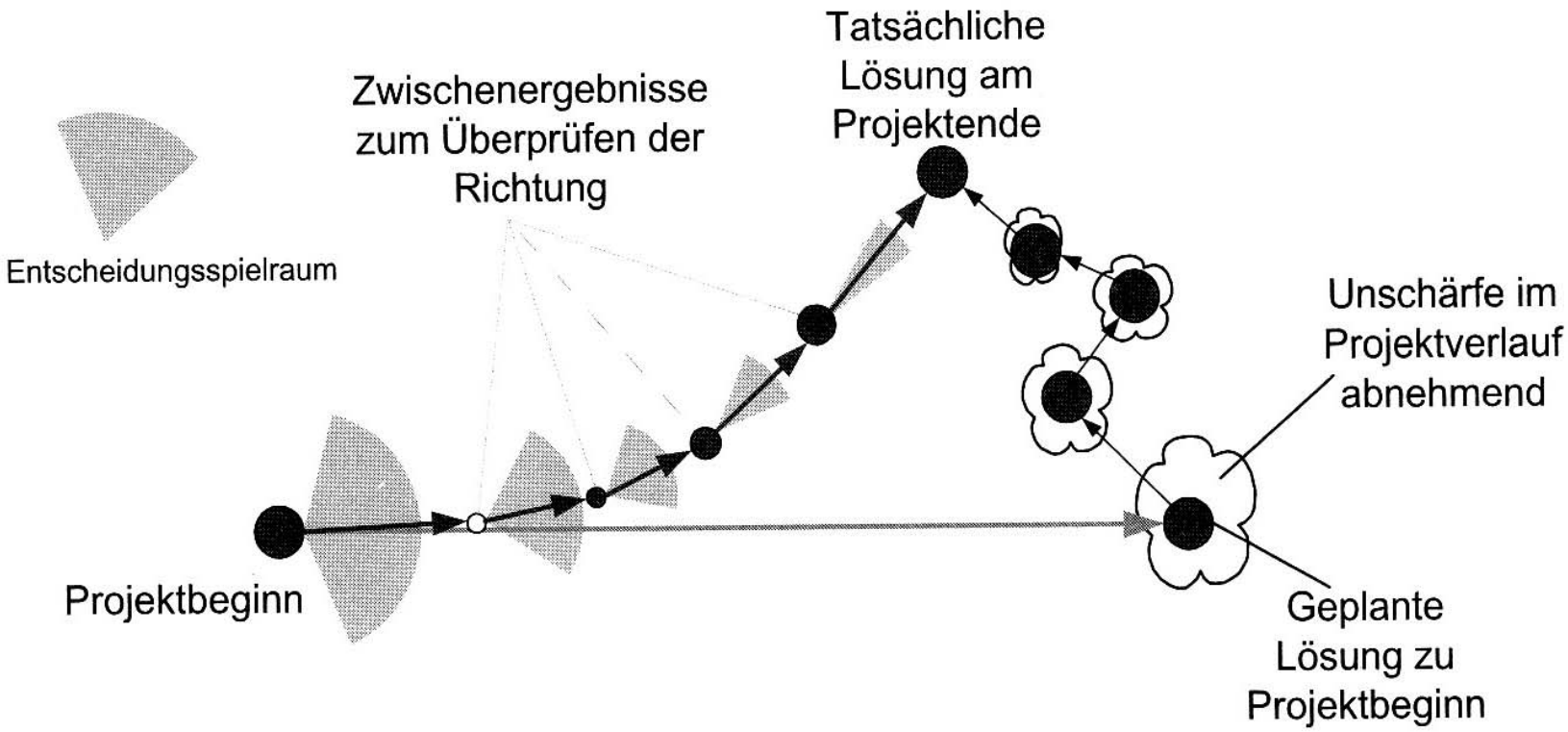
1: Ständige Anpassung an Umgebung, sonst nutzlos (Evolution der Umgebung)

2: Ständig steigende Komplexität, wenn nicht gegengesteuert wird (Patches)

6: Ständige Erweiterung der Funktionalität wird für Kundenzufriedenheit benötigt



(2) Agilität statt Vorhersagbarkeit



Fragestellungen:
- Wie gut versteht der Kunde seine wirklichen Bedürfnisse?
- Wie schnell ändert sich der Markt für die Software?

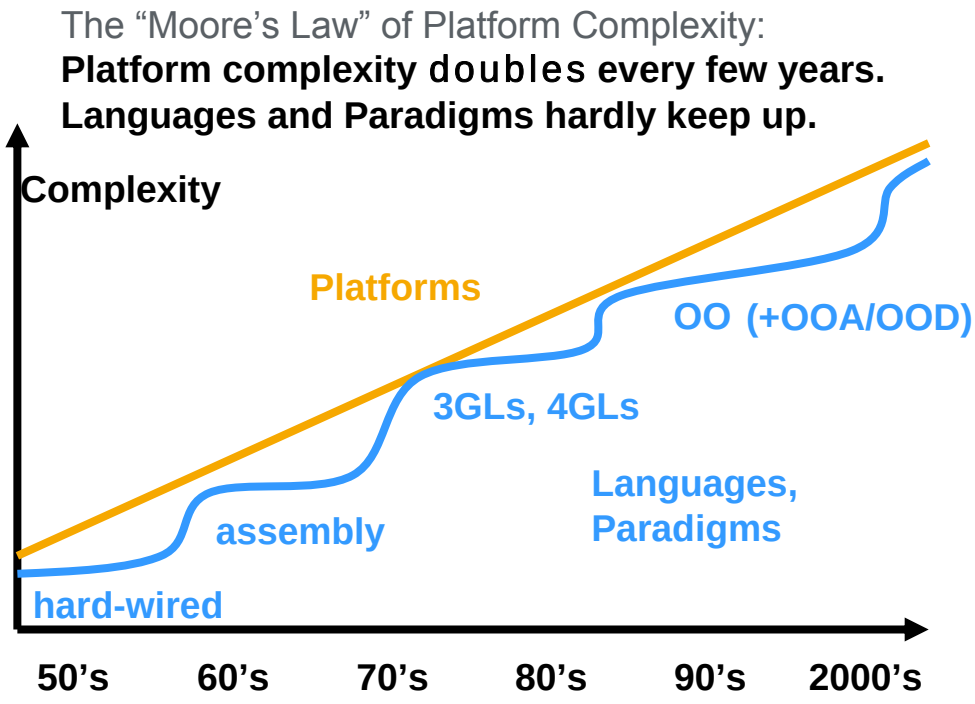
⇨ Wieviel Aufwand lohnt sich bei den frühen Entscheidungen?

(3) Exponentiell wachsende Komplexität

Hintergrund:

- **Moore's Law:**
  Leistungsfähigkeit der Hardware verdoppelt sich (bisher) alle 18 Monate.
- **Nathan's 2. Law:**
  Software wächst bis sie durch Moore's Law begrenzt wird.
- **Nathan's 4. Law:**
  Bedarf nach Software ist nur durch menschliche Zielsetzung/Erwartungen begrenzt.

⇨ Möglichkeiten werden immer voll ausgeschöpft werden!



Dr. Axel Uhl, Chief Development Architect, SAP, Office of the CTO

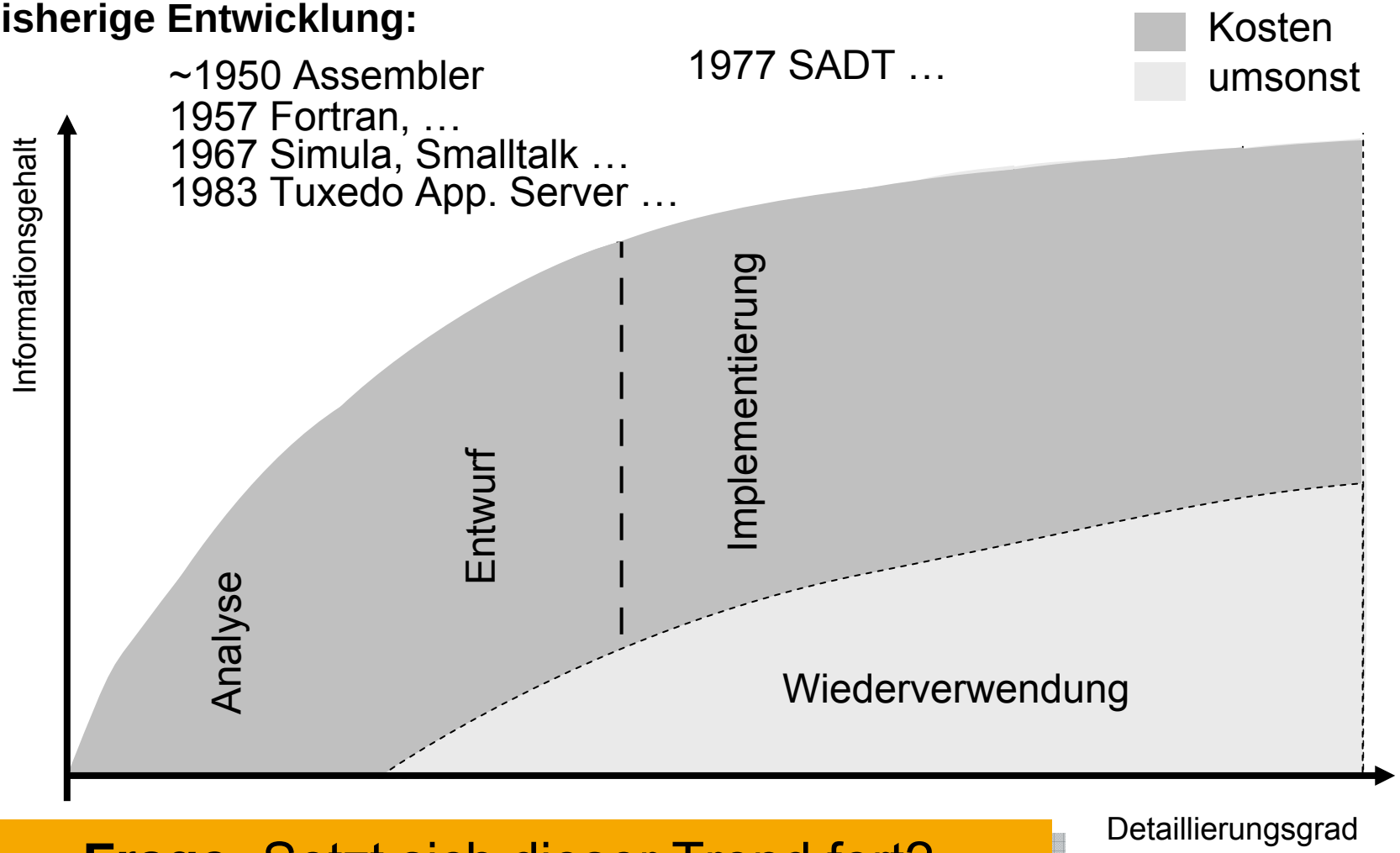
Herausforderungen und Modellierung?

- Ständige Evolution statt Revolution
  - ⇨ **Problem:** Es gibt nur Programme und **keine Modelle**?
  - ⇨ **Problem:** **Konsistenz** der Modelle und des Programms?
- Agilität statt Vorhersagbarkeit?
  - ⇨ **Problem:** **Frühes Feedback** für Anforderungen wichtig!
  - ⇨ **Problem:** Zusätzliche **Änderung** der Modelle notwendig!
- Exponentiell wachsende Komplexität
  - ⇨ **Problem:** **Steigerung der Produktivität**?

Resultierende Frage: Fahren wir in Zukunft besser mit mehr oder weniger Modellieren als heute?

IV. Die Zukunft der Modellierung?

Bisherige Entwicklung:



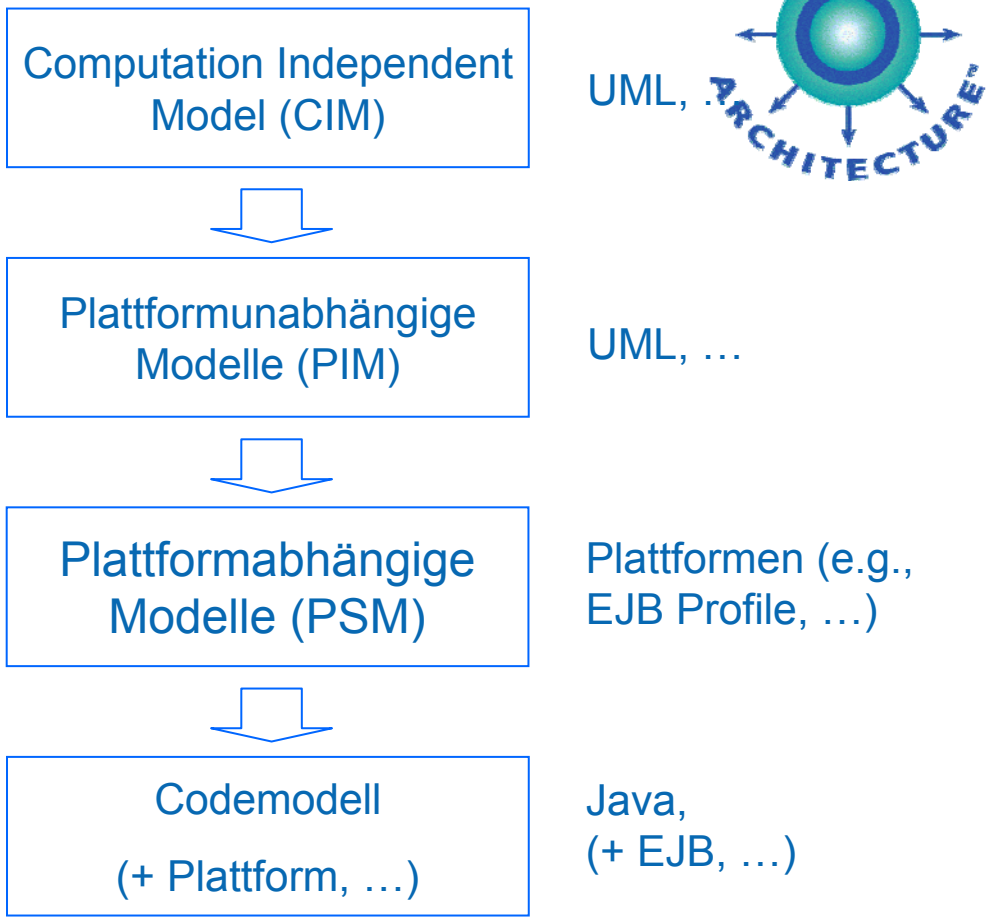
Frage: Setzt sich dieser Trend fort?

(1) Model-Driven Architecture (MDA)

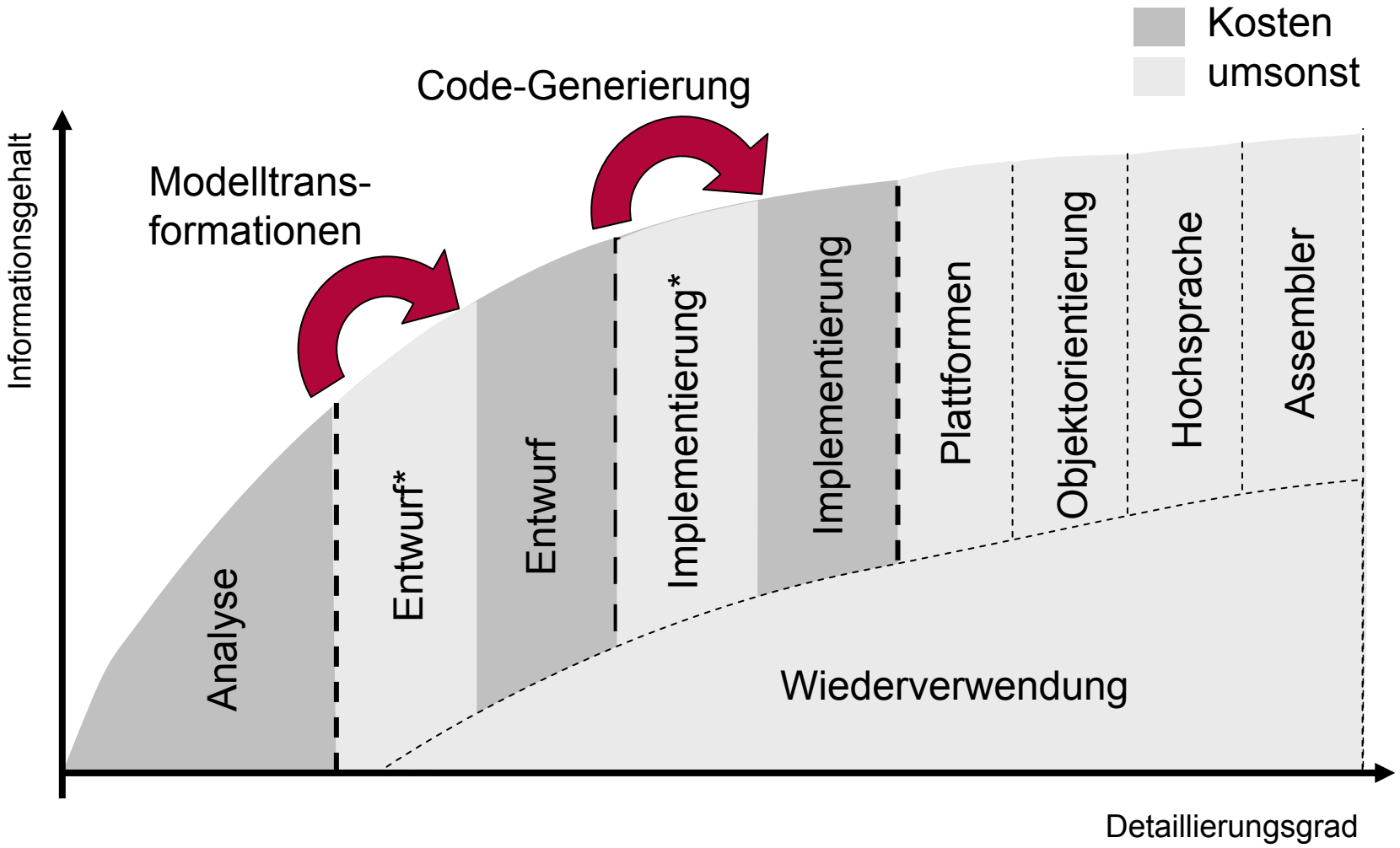
Ein Ansatz zur Spezifikation von IT Systemen, der

- die Spezifikation der **Systemfunktionalität** betreibt
- ohne jedoch sich bereits auf eine gewisse **Technologieplattform** festzulegen.

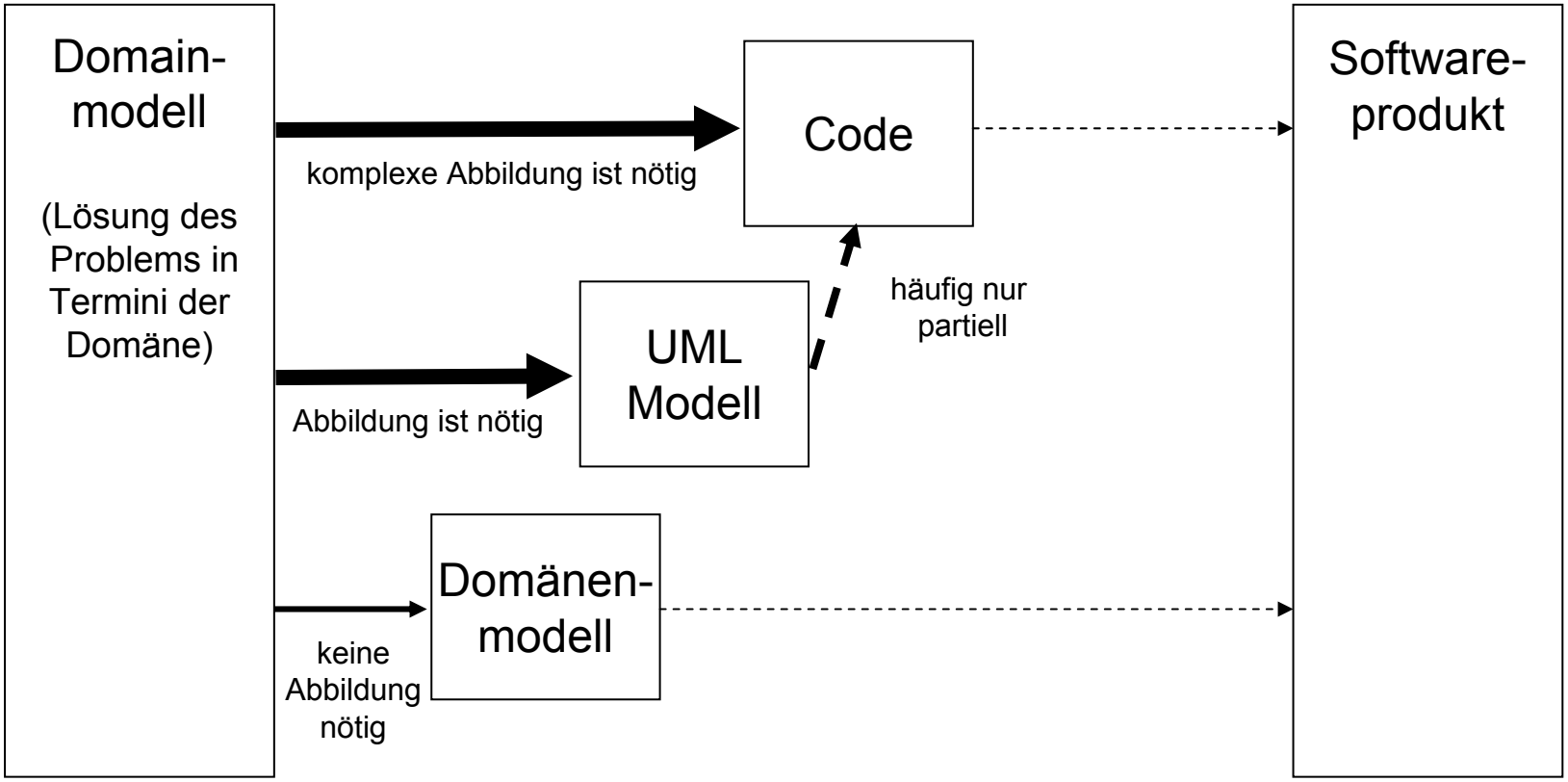
"Design **once**, build it on **any platform**"



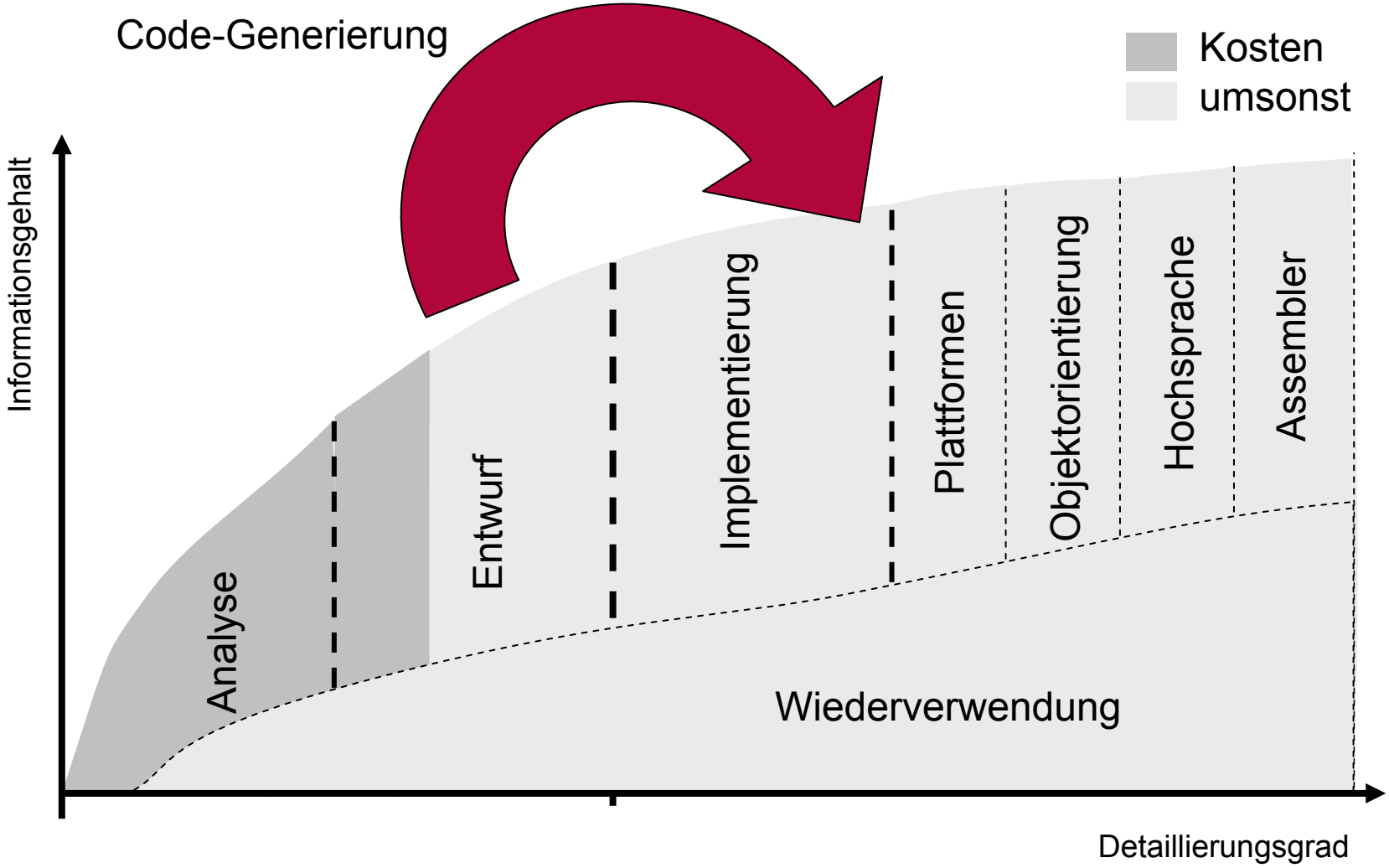
Zukünftig Entwicklung: Ansatz der MDA



(2) Domänenspezifische Sprachen (DSL)



Zukünftig Entwicklung: Ansatz für DSLs



Luft- und Raumfahrt: Beispiel SCADE (1/2)

Werkzeug:

Safety Critical Applications Development Environment (SCADE)

Source: Esterel Technologies

Anwendung:

A340/600 FCSC (Flight Control Secondary Computer):

Ergebnis:

- 70% automatisch generierter Code
- 50% Einsparung bei den **Entwicklungskosten**
- Verkürzung der **Änderungszeit** um den Faktor 3

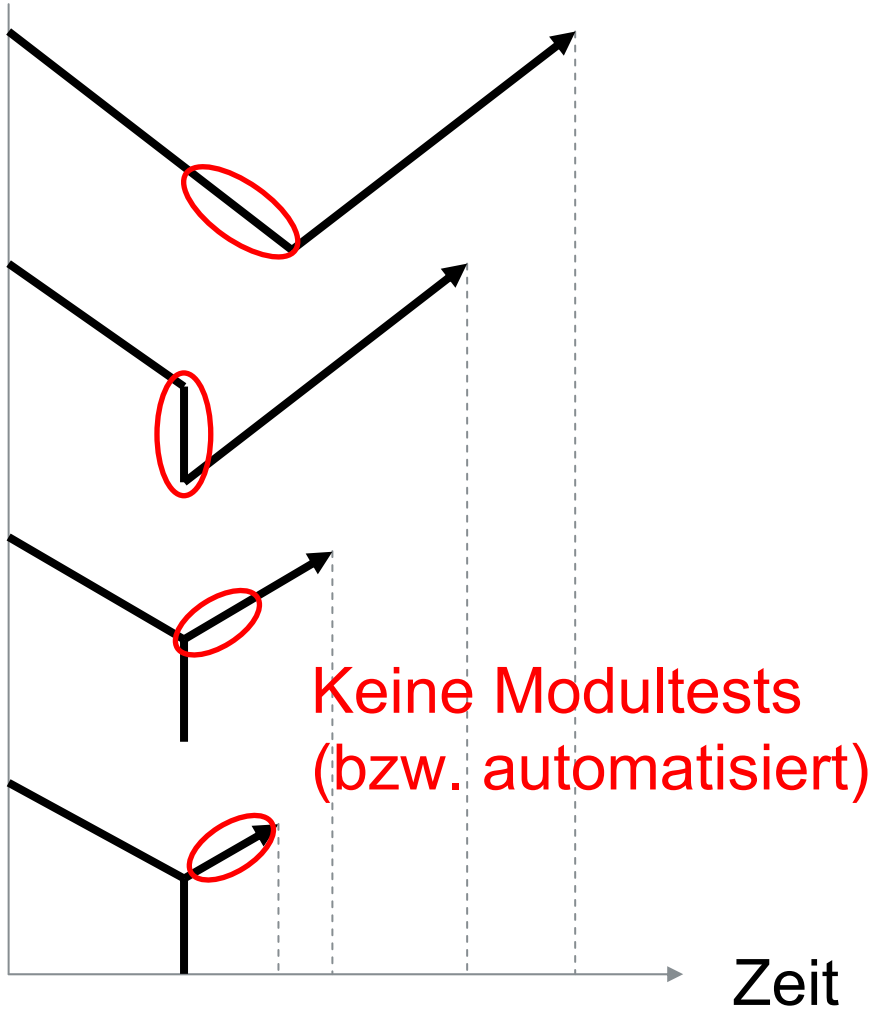
Luft- und Raumfahrt: Beispiel SCADE (2/2)

Manuelle Programmierung

Automatische Codegeneratoren

Qualifizierte Codegeneratoren

Verifikation auf Basis der Modelle



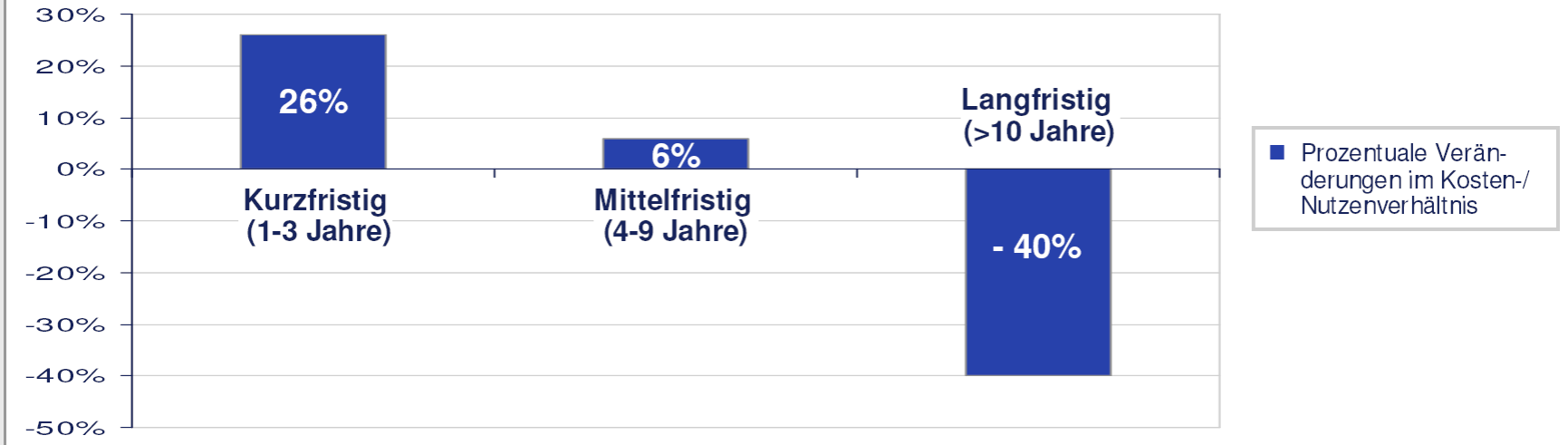
[Camus&Dion2003] http://www.safeair.org/
http://www.safeair.org/description/D4-2_final_Report_v1.7.pdf

Luft- und Raumfahrt: Einsparungspotential

Einsparung durch die modellbasierte Softwareentwicklung

35 %	Kosteneinsparung durch die modellbasierte Softwareentwicklung
25 %	Zeiteinsparung durch die modellbasierte Softwareentwicklung

Veränderung im Kosten-/Nutzenverhältnis durch die modellbasierte Softwareentwicklung



Quelle: Arthur D. Little GmbH

Hans-Peter Erl & Sascha Kirstan . Studie "Kosten-/Nutzenanalyse der Modellbasierten Softwareentwicklung im Automobil" Arthur D Litte/TU Munich, München, Januar 2007.

Diskussion der Ansätze

Frage	MDA	DSLs	MDA+DSLs
Anwendbarkeit	komplexe Projekte	einfache Projekte	komplex Projekte
Aufwand (Infrastruktur)	niedrig	**sehr hoch**	**mittel**
Aufwand (Analysetechniken)	**mittel**	**sehr hoch**	**hoch**
Aufwand (Projekt)	**mittel**	sehr niedrig	niedrig
Keine Modelle	**Rückwärts?**	**Rückwärts?**	**Rückwärts?**
Konsistenz / Zusätzliche Änderung	Vorwärts ✓ **Rückwärts?**	Vorwärts ✓ (Rückwärts)	Vorwärts ✓ **Rückwärts?**
Frühes Feedback?	**?**	(Ausführung)	(Ausführung)
Steigerung der Produktivität	+	+ + +	+ +

V. Ausblick & Arbeiten am Fachgebiet

Wesentliche offene Punkte:

- Frühes Feedback?
- Keine Modelle?
- Konsistenz/zusätzliche Änderung
- Infrastrukturkosten?
- Kosten für Analysentechniken?

„Prognosen sind schwierig, besonders wenn sie die Zukunft betreffen.“ (Winston Churchill u.a.)

Prognosen:

- Für einige Domänen werden sich entsprechende DSLs durchsetzen (**Dominanz**):
  - Luft- und Raumfahrt: ✓
  - Eingebettete Systeme (und speziell Automotive Systeme): wird kommen!
  - Business IT: Geschäftsprozessnotationen werden vermehrt als DSLs fungieren
- Für komplexe, allgemeine IT-Systeme wird es beides geben (**Koexistenz**):
  - Auf Architekturebene Modelle (wahrscheinlich UML)
  - Innerhalb der Architekturkomponenten Programmiersprachen oder DSLs

Zukunftsvision:

- Software, die sich mit Hilfe von Modellen teilweise selbst funktionstüchtig erhält (autonomic computing, Selbst-X).

Arbeiten am Fachgebiet

„Die beste Art, die Zukunft vorauszusagen, ist, die Zukunft zu erfinden“ (Alan Kay)

Wesentliche offene Punkte:

- Frühes Feedback?
- Keine Modelle?
- Konsistenz/zusätzliche Änderung
- Infrastrukturkosten?
- Kosten für Analysentechniken?

Prototyping mit Modellen

HPI – Stanford Design Thinking Research Program

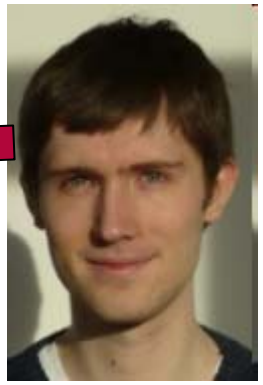
Modellsynchronisation (vorwärts, rückwärts)

Modellkonsistenz

Modellanalysen

Modelle, SOA & Selbst-X (Agilität & Evolution zur Laufzeit)

N.N.



Stephan Hildebrandt

Thomas Vogel

Basil Becker

Stefan Neumann

Andreas Seibel

Fr. Miers

Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Fragen?

Prof. Dr. Holger Giese

Fachgebiet Systemanalyse und Modellierung