



**Hasso
Plattner
Institut**

IT Systems Engineering | Universität Potsdam

Quo vadis, Modellierung?

Prof. Dr. Holger Giese

Fachgebiet Systemanalyse und Modellierung

Übersicht

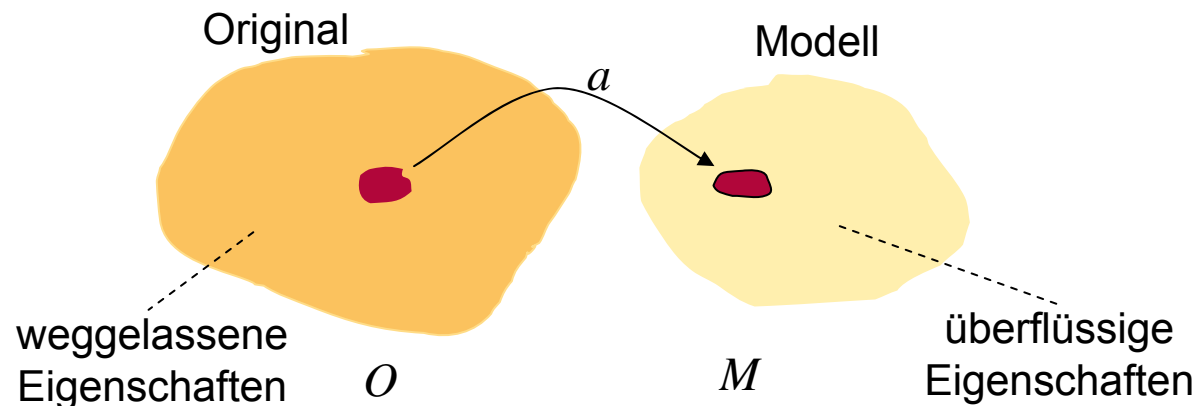
2

- 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?

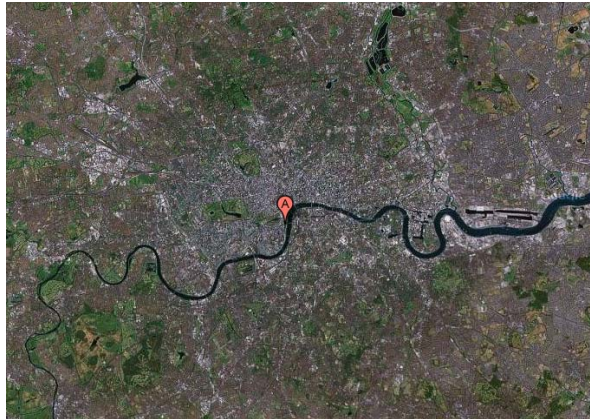
Modelle sind im Allgemeinen abstrakte Abbilder oder Vorbilder zu konkreten oder abstrakten Originalen.

- **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** ersetzen



Beispiel: Karte der Londoner U-Bahn

4



London



Karte der U-Bahn von London

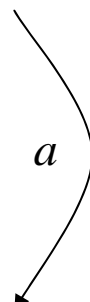


Abbildung:

- Bezug zur „echten“ U-Bahn

Reduktion:

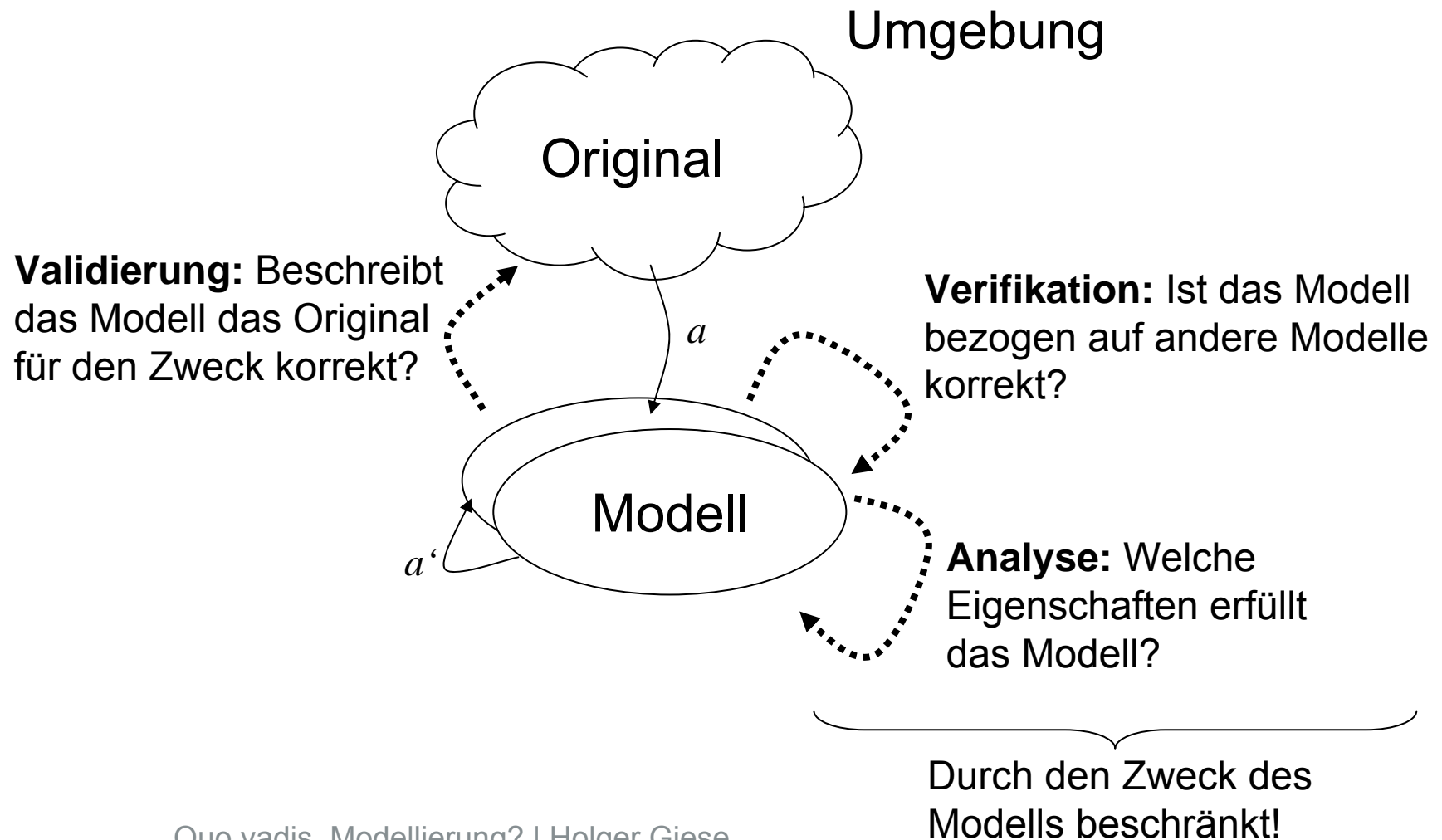
- Weggelassene Eigenschaften:
 - Korrekte geometrische Distanzen/Lage
 - Höheninformationen, Straßen, ...
- Überflüssige Eigenschaften:
 - Geometrische Distanzen/Lage sind nicht korrekt

Pragmatik:

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

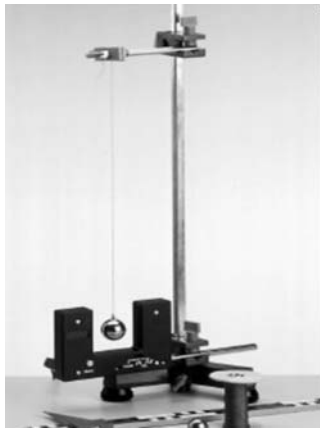
Validierung, Verifikation & Analyse

5

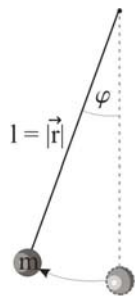


Modelle in den Naturwissenschaften

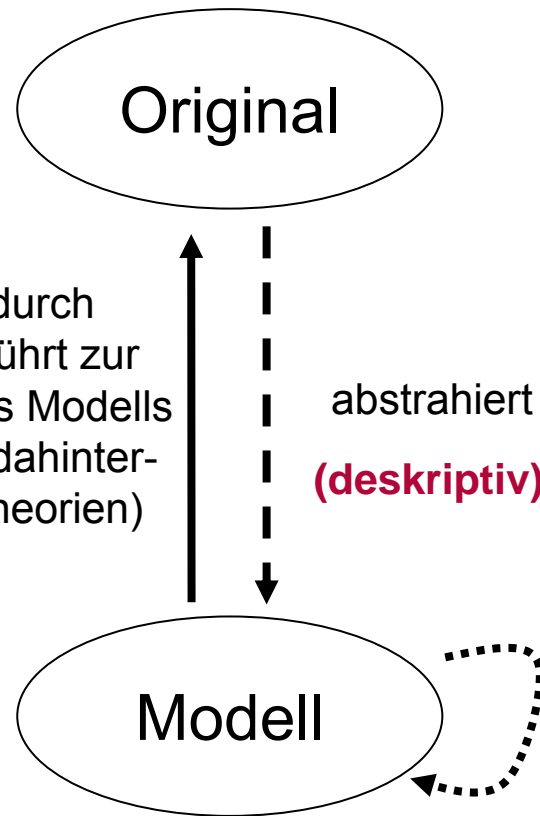
6



Vergleich durch Experiment führt zur Validierung des Modells (und ggf. der dahinterstehenden Theorien)



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



Konstruktion:

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

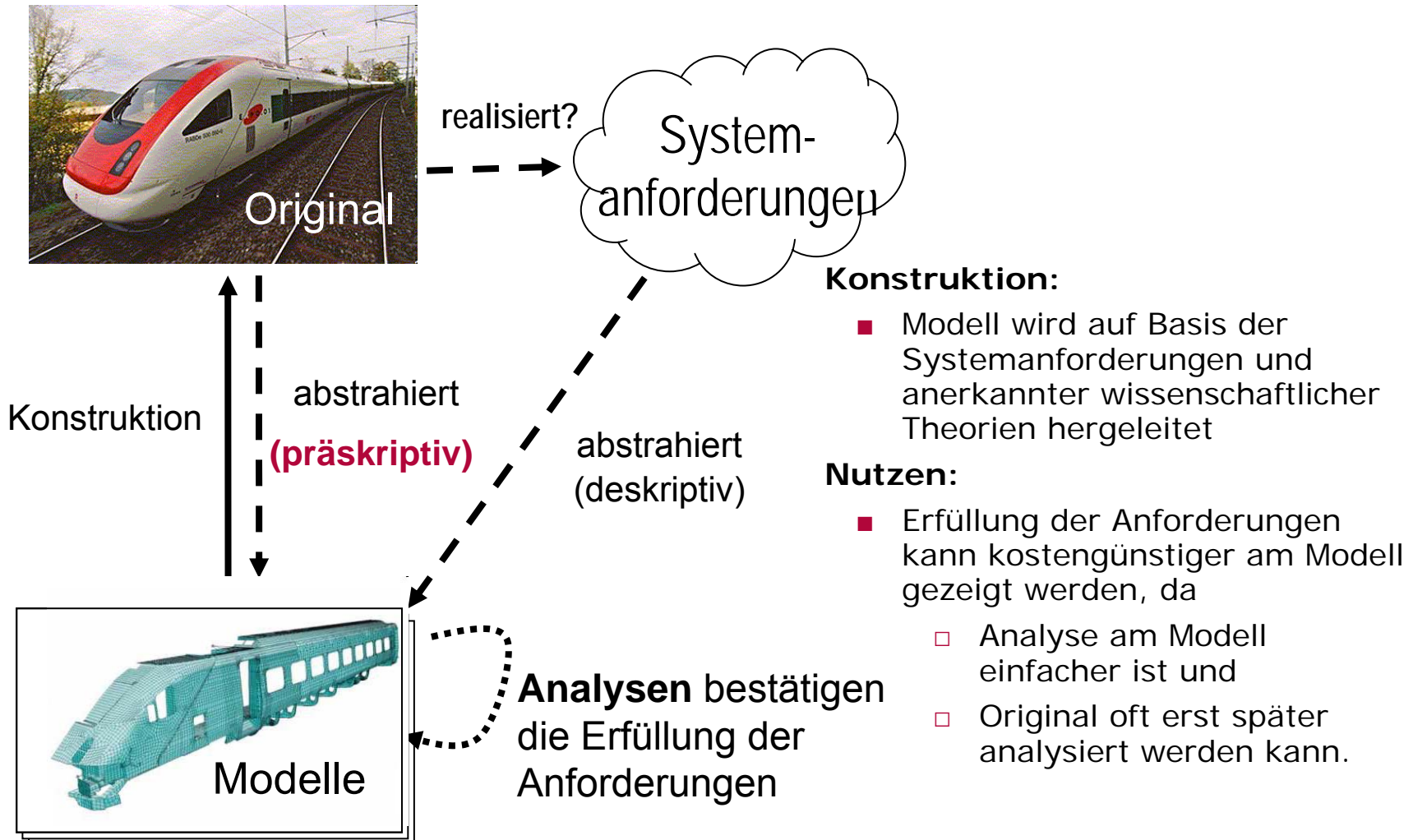
Nutzen:

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

Analyse
ermöglicht
Vorhersagen

Modelle bei den Ingenieuren

7

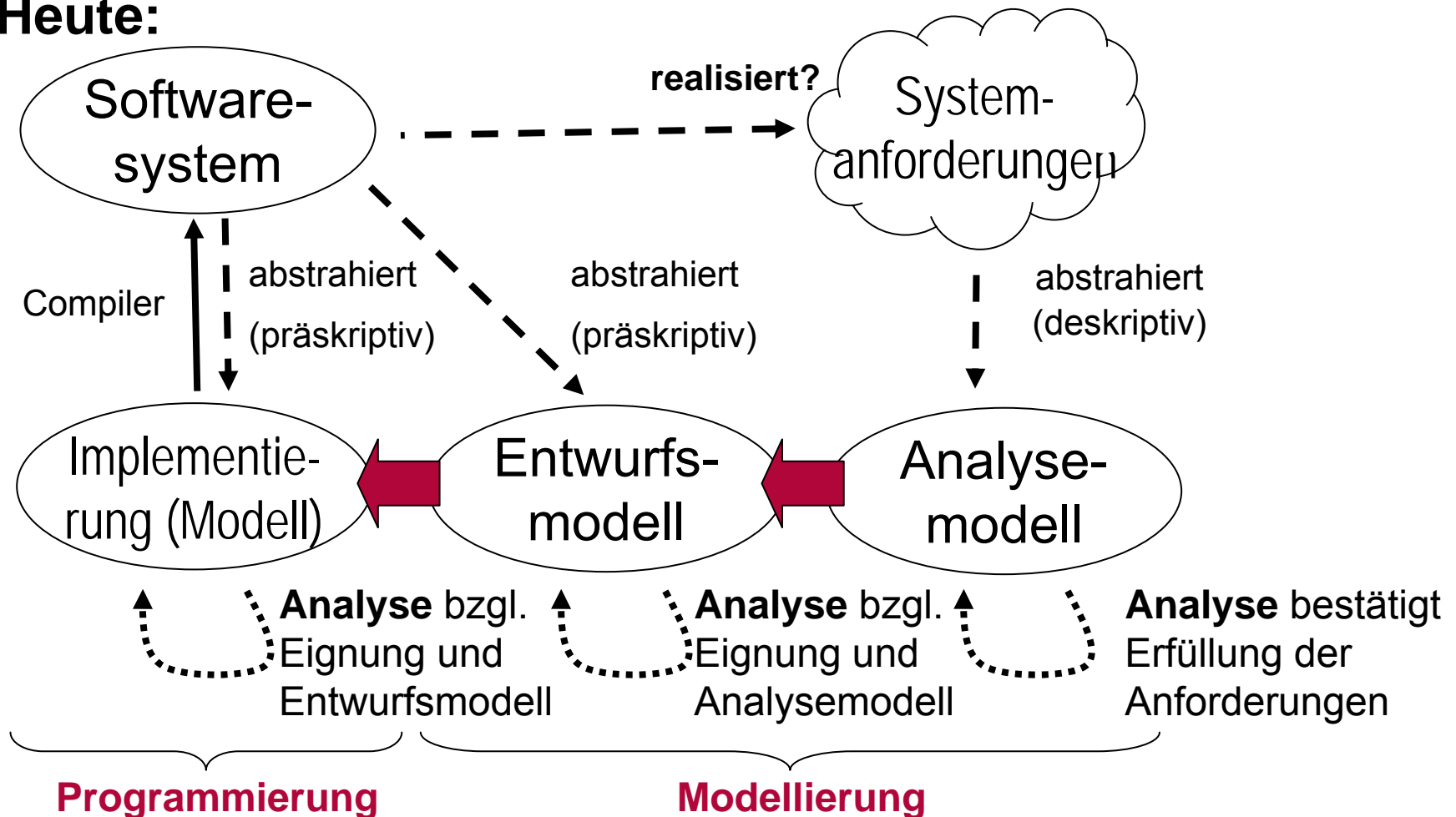


II. Modellierung in der Informatik

8

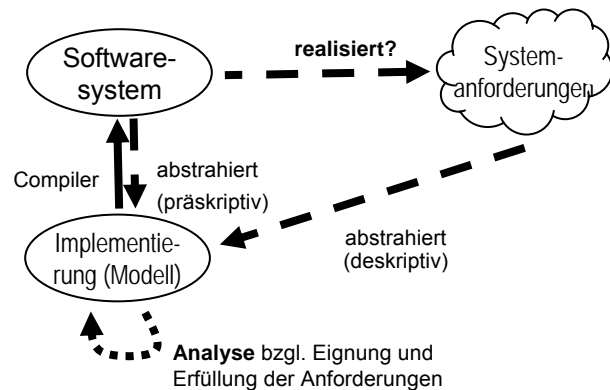
Beobachtungen: Original und Modelle sind **immateriell**

Heute:



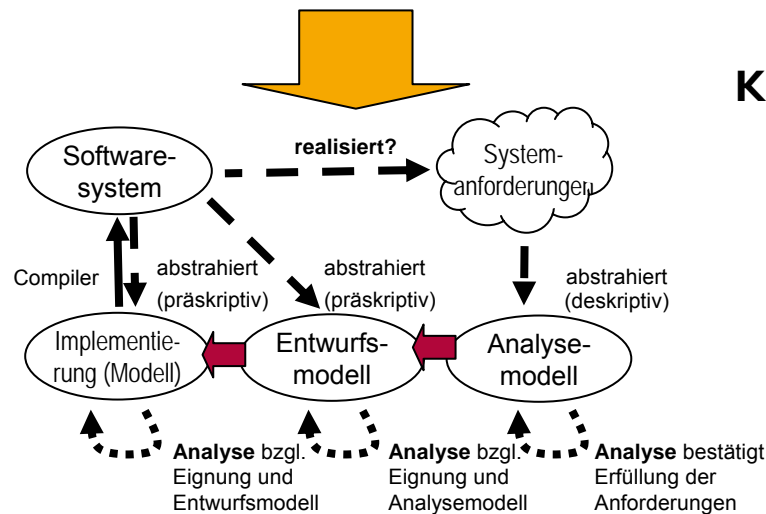
Wieso nicht nur Programmierung?

9



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

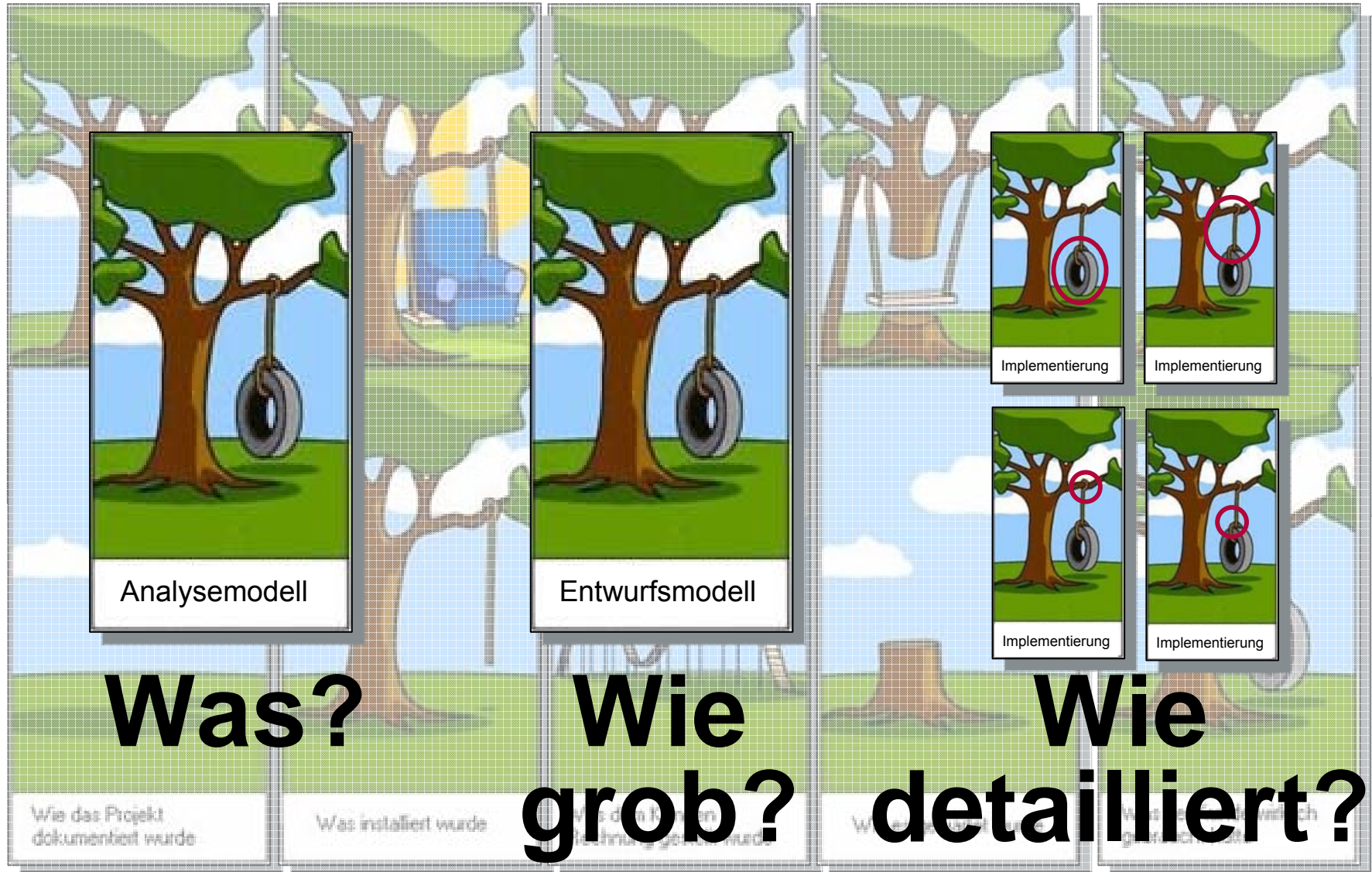


Konsequenzen:

- **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

10



III. Aktuelle Herausforderungen

11

- (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

12

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)

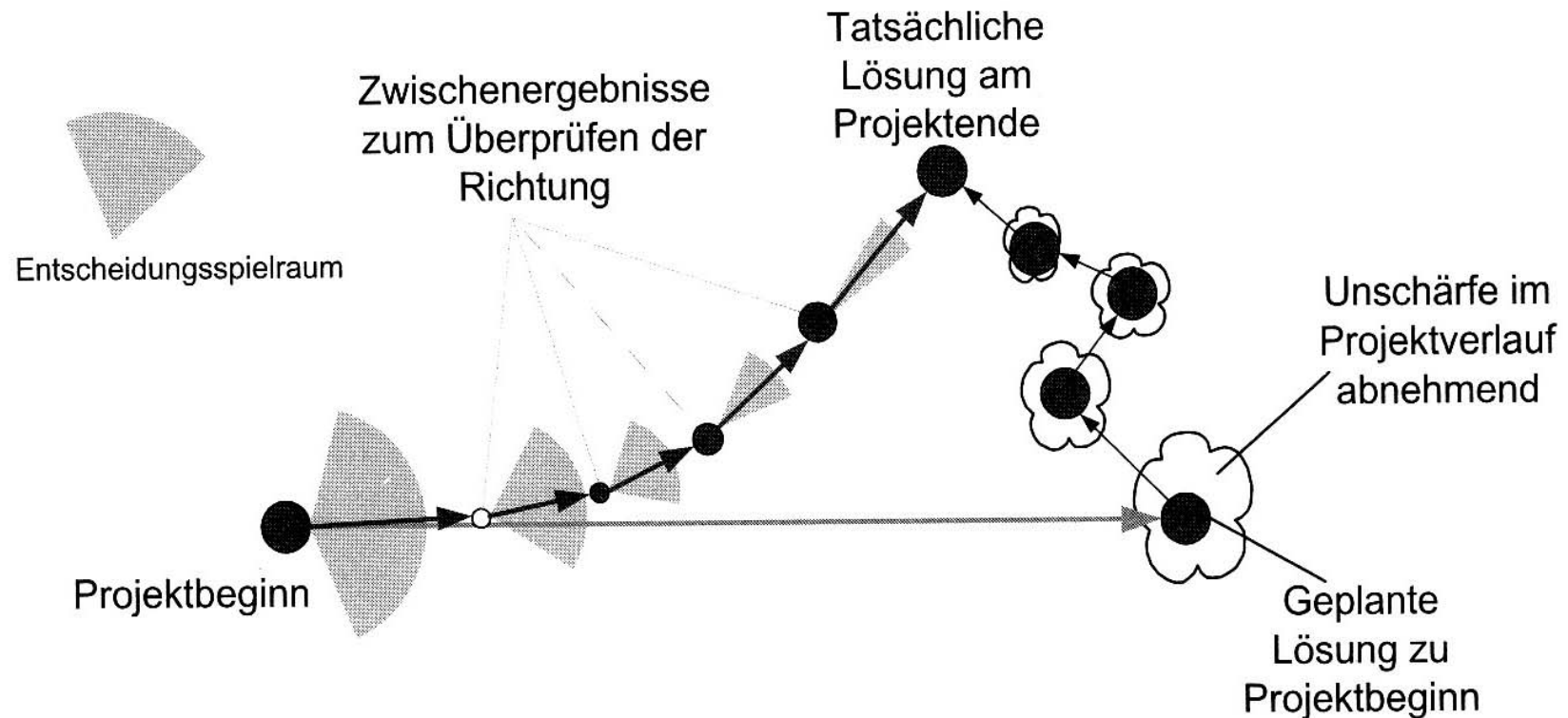
2: Ständig steigende Komplexität, wenn nicht gegengesteuert wird (Patches)

6: Ständige Erweiterung der Funktionalität wird für Kundenzufriedenheit benötigt



(2) Agilität statt Vorhersagbarkeit

13



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?

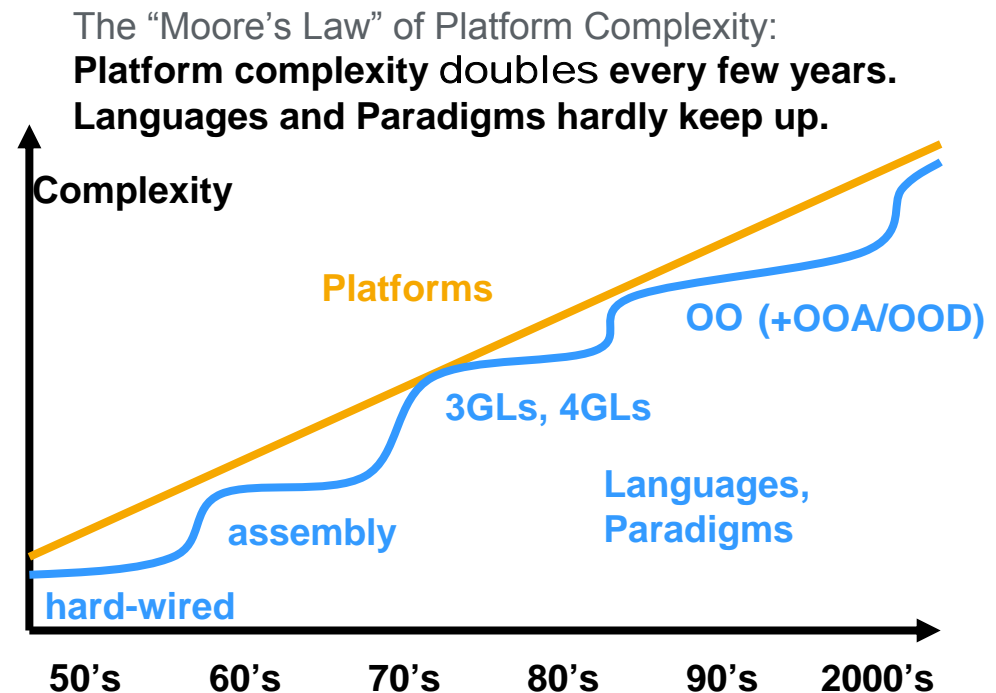
(3) Exponentiell wachsende Komplexität

14

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?

15

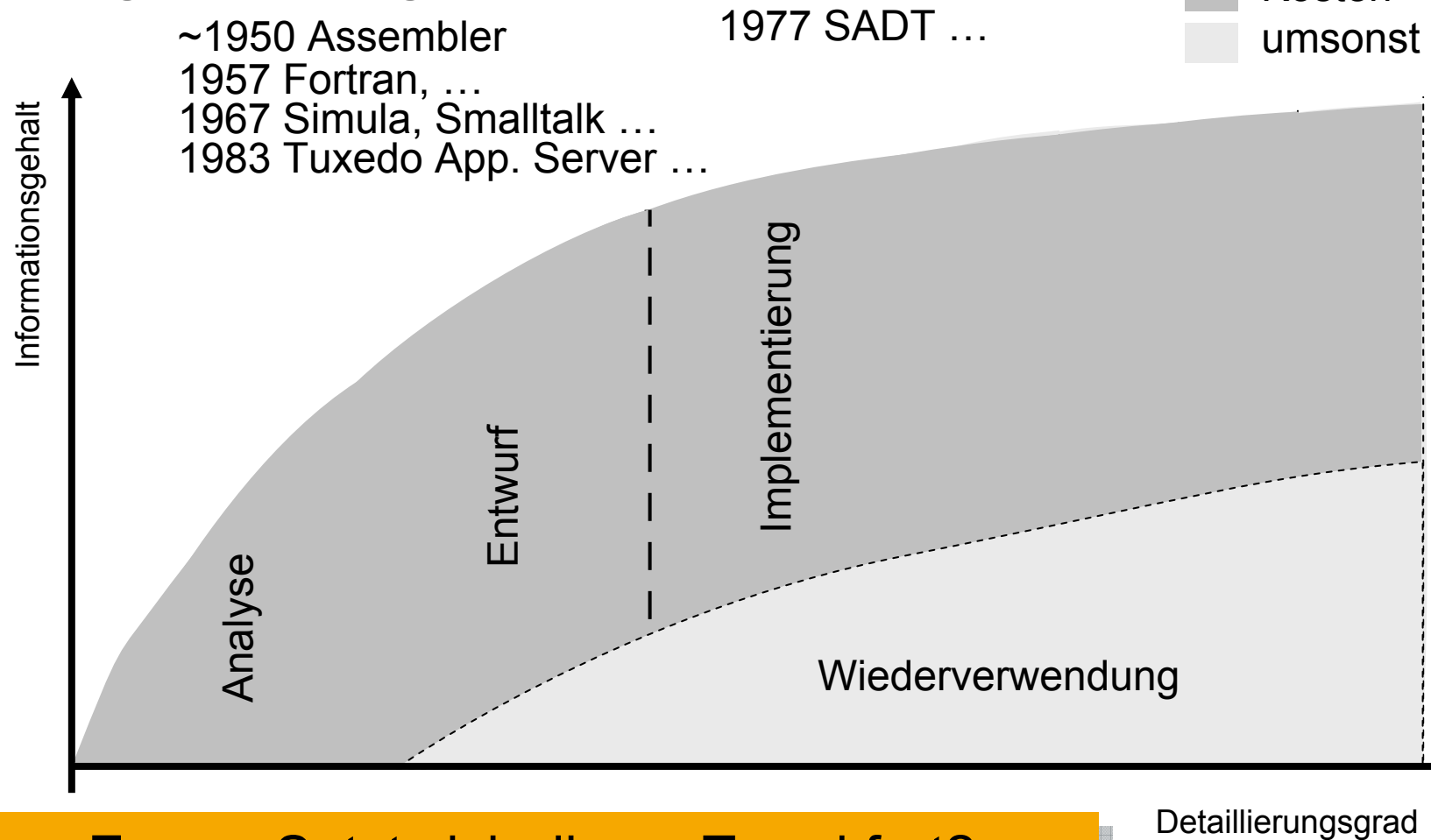
- Ständige Evolution statt Revolution
 - ⇒ **Problem:** Es gibt nur Programme und **keine Modelle**?
 - ⇒ **Problem:** **Konsistenz** der Modelle und des Programms?
- Agilität statt Vorhersagbarkeit?
 - ⇒ **Problem:** **Frühes Feedback** für Anforderungen wichtig!
 - ⇒ **Problem:** Zusätzliche **Änderung** 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?

16

Bisherige Entwicklung:



Frage: Setzt sich dieser Trend fort?

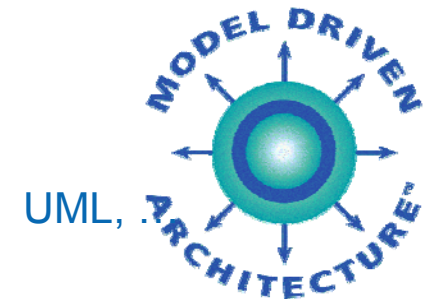
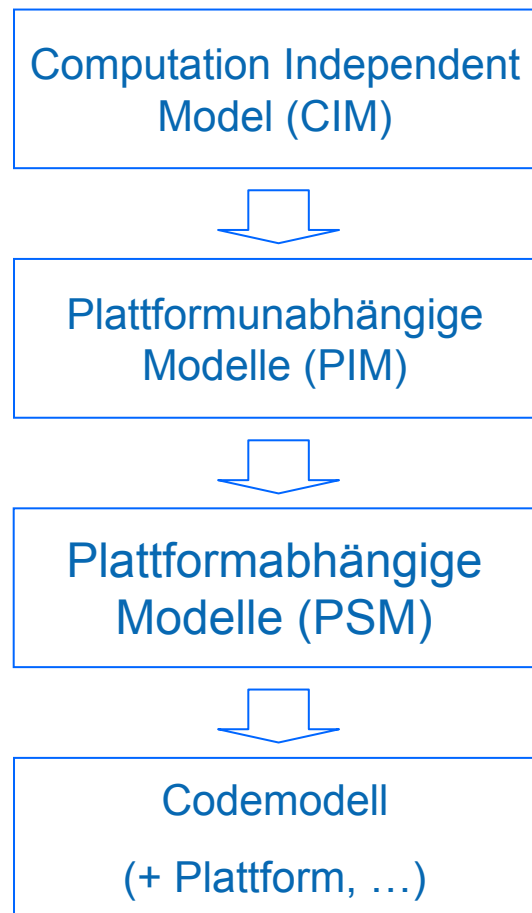
(1) Model-Driven Architecture (MDA)

17

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”**



UML, ...

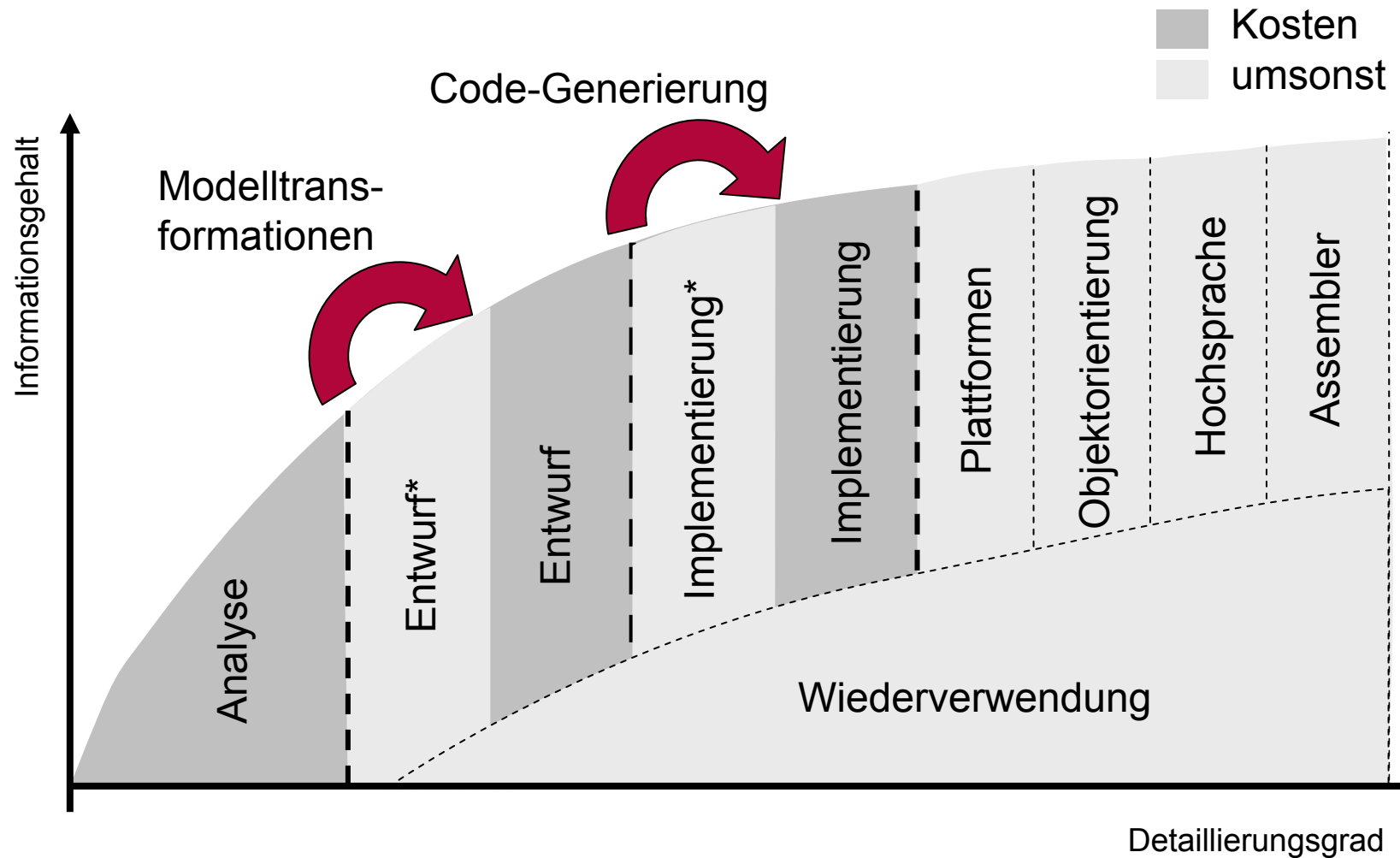
UML, ...

Plattformen (e.g., EJB Profile, ...)

Java, (+ EJB, ...)

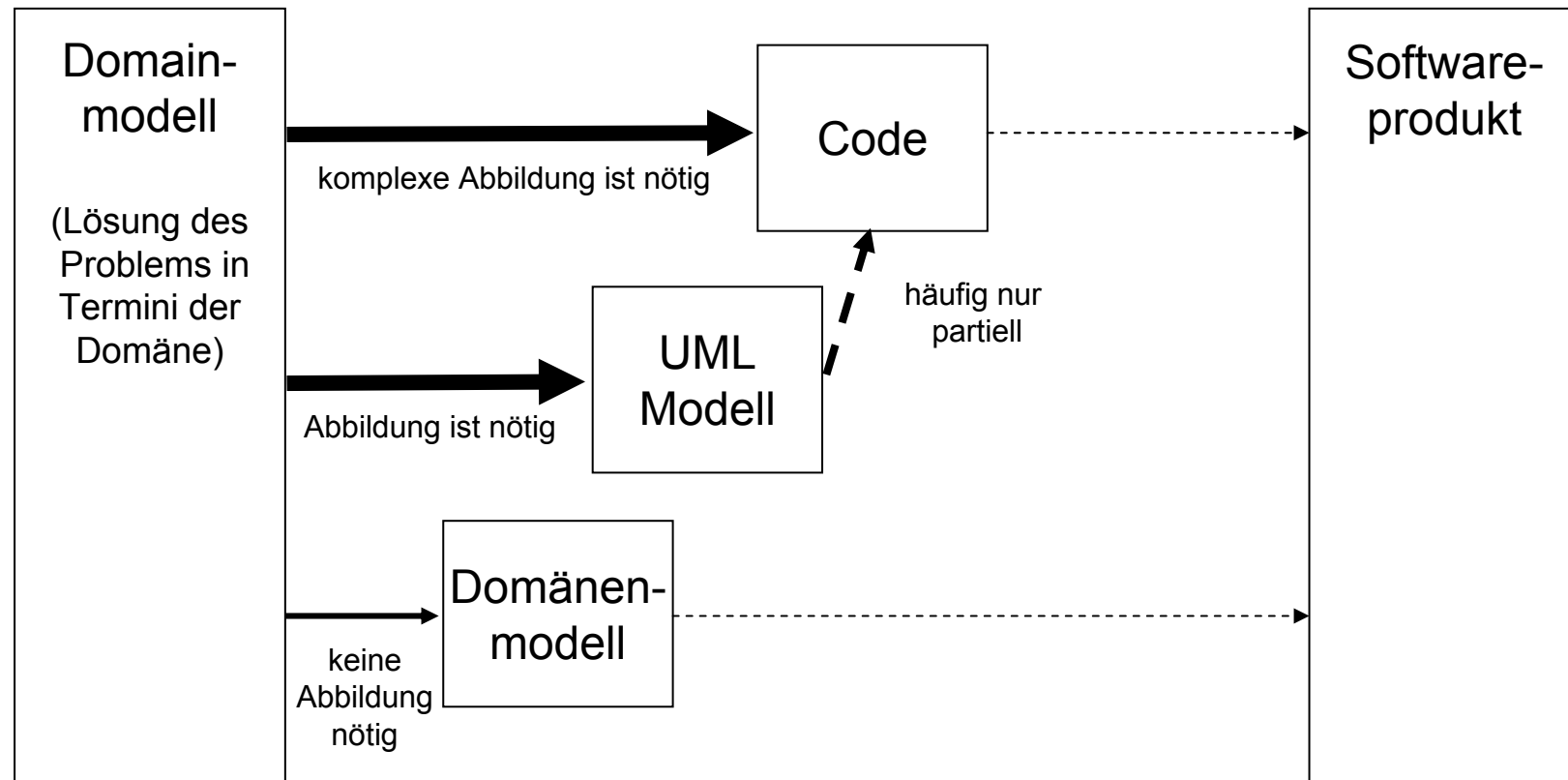
Zukünftig Entwicklung: Ansatz der MDA

18



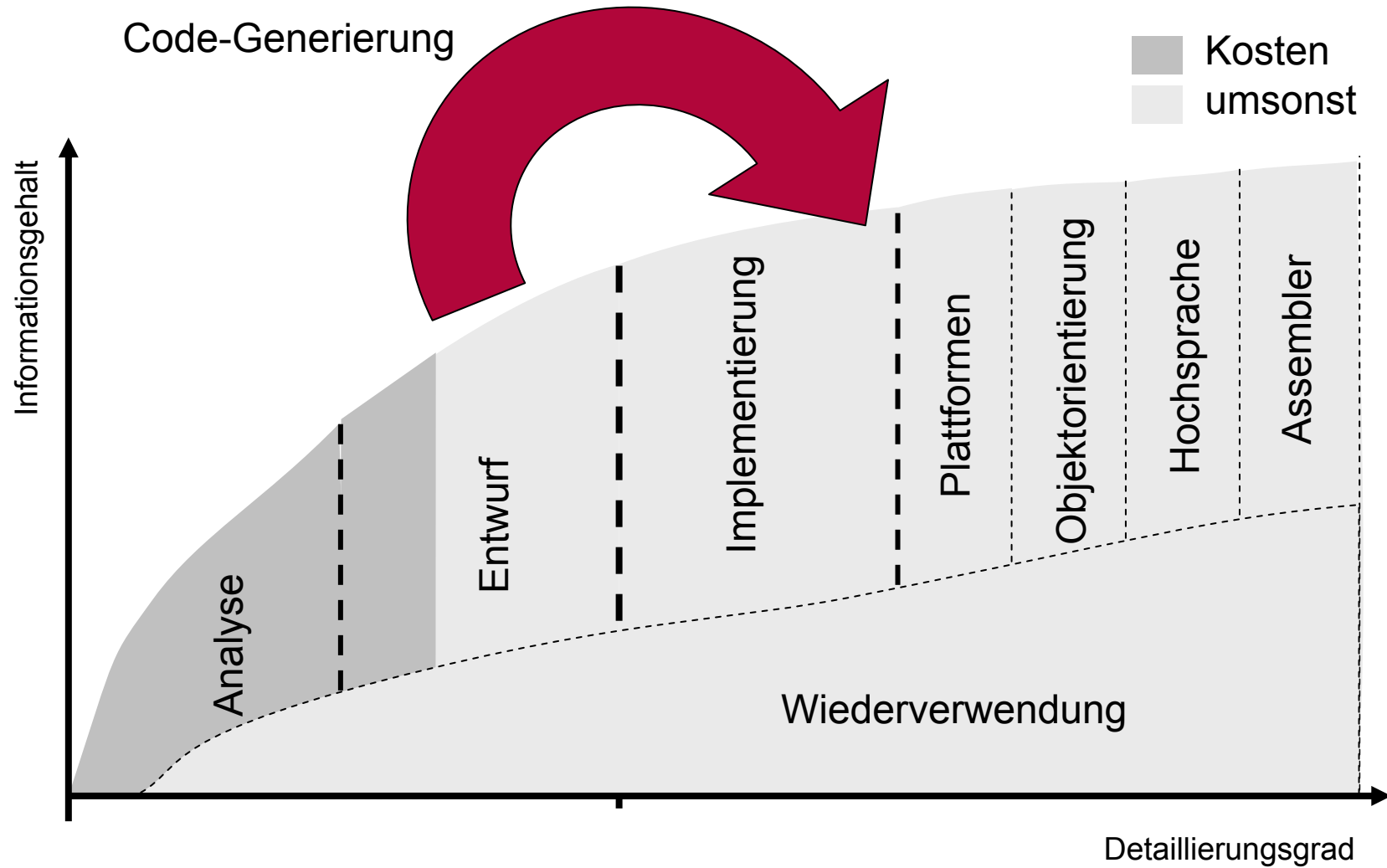
(2) Domänenspezifische Sprachen (DSL)

19



Zukünftig Entwicklung: Ansatz für DSLs

20



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

21

Werkzeug:

Safety Critical Applications
Development Environment
(SCADE)

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



Source: Esterel Technologies

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

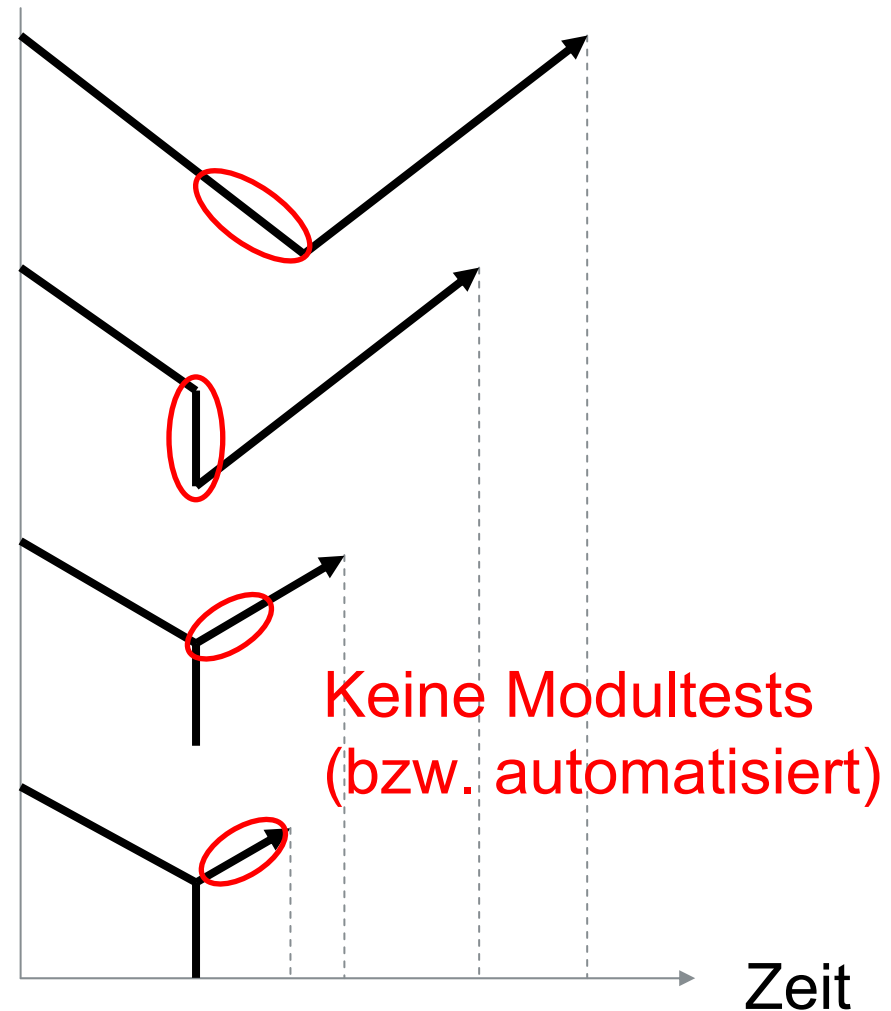
22

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](http://www.safeair.org/http://www.safeair.org/description/D4-2_final_Report_v1.7.pdf)

Luft- und Raumfahrt: Einsparungspotential

23

Einsparung durch die modellbasierte Softwareentwicklung

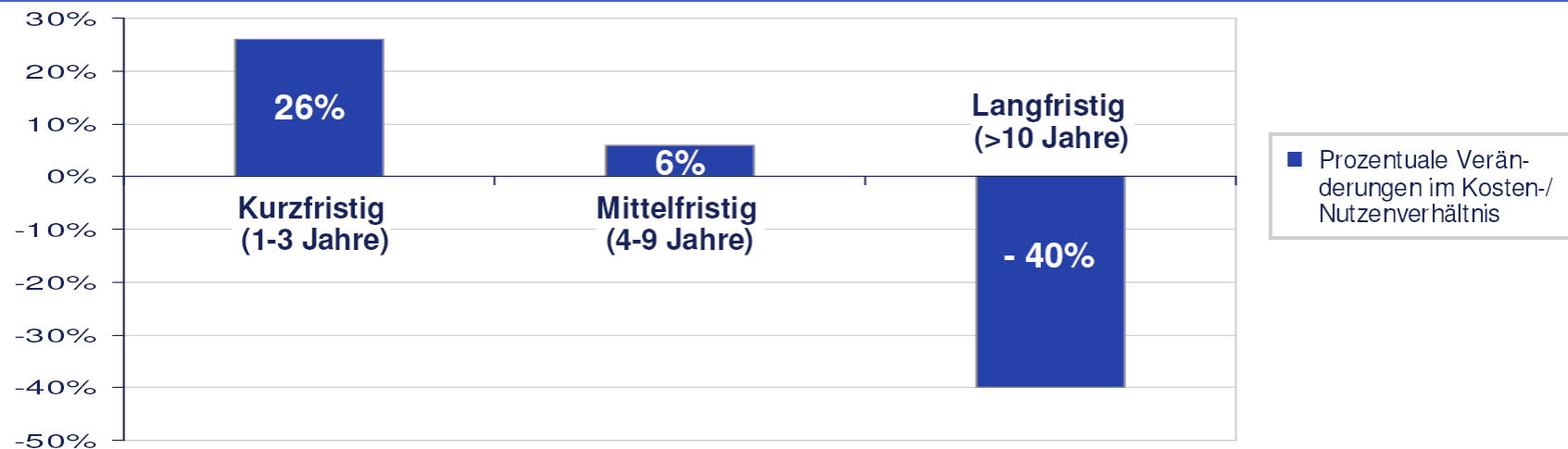
35 %

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

24

| 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

25

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)



26

Wesentliche offene Punkte:

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

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



Thomas Vogel



Basil Becker



Stefan Neumann



Andreas Seibel



Fr. Miers

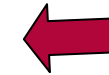
Prototyping mit Modellen



Modellsynchronisation
(vorwärts, rückwärts)

Modellkonsistenz

Modellanalysen



N.N.



Stephan Hildebrandt



**Hasso
Plattner
Institut**

IT Systems Engineering | Universität Potsdam

Fragen?

Prof. Dr. Holger Giese

Fachgebiet Systemanalyse und Modellierung