Repository-basierte Anwendungssystem-Architekturen

(Von einem sprachlogischen Standpunkt)

Erich Ortner
Entwicklung von Anwendungssystemen
Technische Universität Darmstadt

INHALT

1. Wann ist in der Informatik und Wirtschaftsinformatik von Architekturen die Rede?

2. Sprachbasierte Architekturen (Grundlagen und Anwendungen)

3. Repository-Anwendungen

   3.1 Entwicklungsrepositorien und Kataloge

   3.2 Dynamik von Anwendungssystemen

4. Tobias Grollius (Vortrag)

1.

Wann ist… von Architekturen die Rede?

Härder/Rahm: Datenbanksysteme – Konzepte und Techniken der Implementierung, Springer 1999.

1.3.2 Architekturprinzipien

Ziel unserer Überlegungen ist die Entwicklung einer Systemarchitektur für ein datenunabhängiges DBS. Da es **keine Architekturlehre** für den Aufbau großer Software-Systeme gibt, können wie keine konkreten Strukturierungsvorschläge heranziehen. Es existieren aus dem Bereich Software Engineering lediglich Empfehlungen, **allgemeine Konzepte wie das Geheimnisprinzip (*information hiding* nach Parnas [PARN72]) und eine hierarchische Strukturierung [PARN75] zu nutzen**. Daraus lassen sich wichtige Hinweise ableiten, große SW-Systeme aus hierarchisch angeordneten Schichten aufzubauen, wobei Schicht i+1 die Operatoren und Datenobjekte „benutzt“[6], die Schicht i „realisiert“. Dieses Aufbauprinzip ist in Abb. 1.3 veranschaulicht.

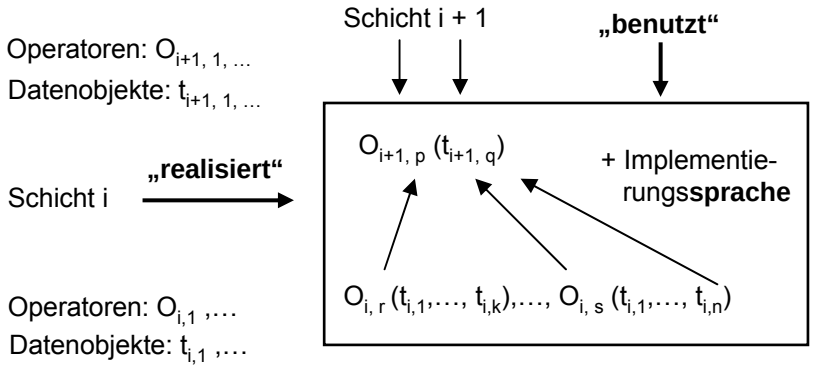


Abb. 1.3: Aufbauprinzip für eine Schicht

1.3.4 Integration von Metadaten- und Transaktionsverwaltung

Beschreibungsdaten werden zur Realisierung der Aufgaben **in jeder Schicht** benötigt. Um dies zu untersteichen, illustrieren wir in Abb. 1.6 die **Metadatenverwaltung** als konzeptionell eigenständige Komponente, welche die **Abstraktionsebenen** aller unserer Modellschichten überdeckt. Das soll nicht bedeuten, dass sie immer auch als eigenständige Komponente realisiert ist. Später werden wir die Beschreibungs- und Abbildungsinformationen **wiederum den einzelnen Schichten zuordnen** und im Rahmen der Konkretisierung der jeweiligen Schicht beschreiben (siehe **Abb. 1.7**).

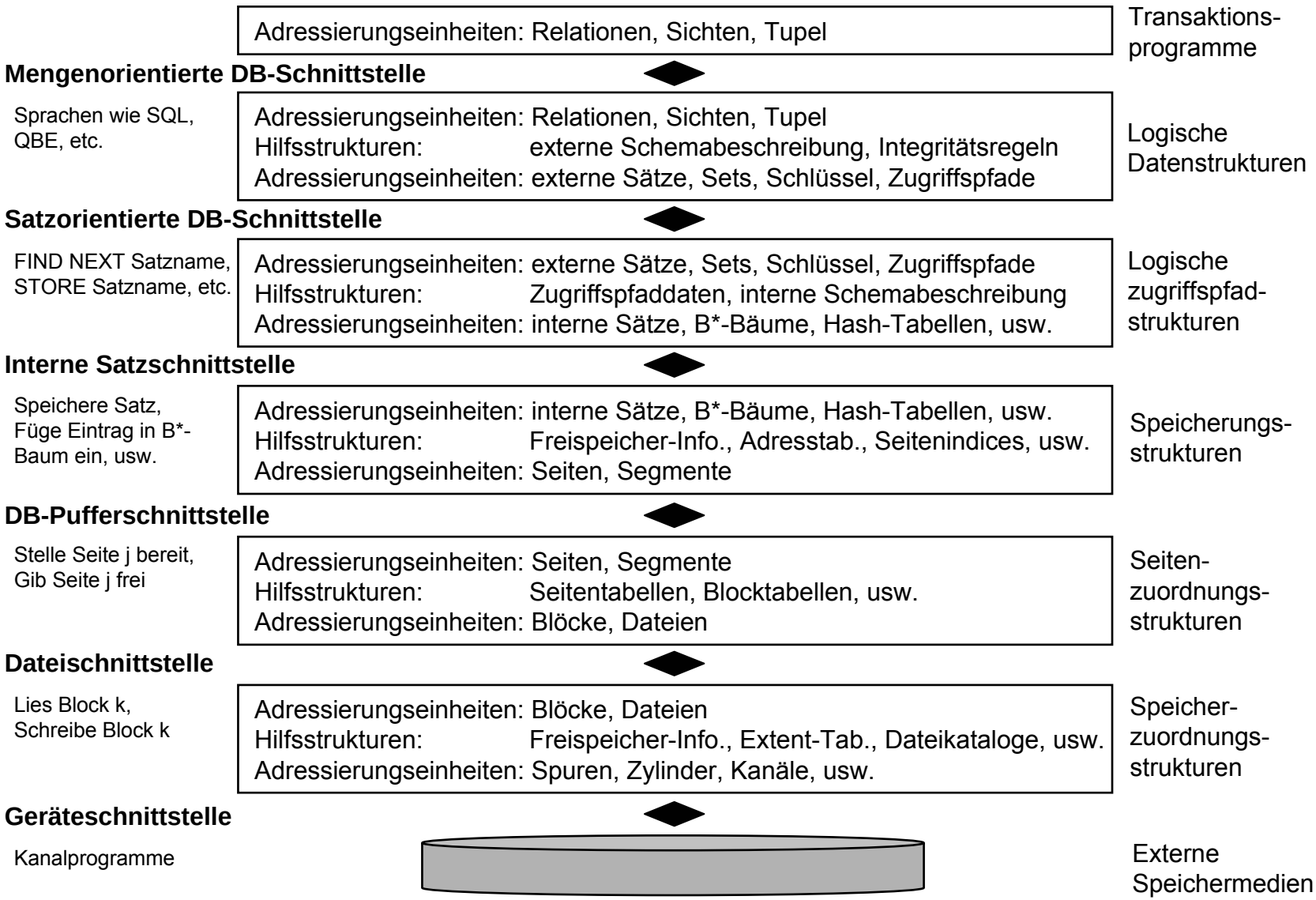


Abb. 1.7: Schichtenmodell für ein datenunabhängiges Datenbanksystem (**Härder/Rahm**)

EAWS

Was bringt die Abstraktionsebenenbildung (Implementierung) in der rechnerunterstützten Informationsverarbeitung?

Nach Parnas ergeben sich unmittelbar eine Reihe von Vorteilen für die Entwicklung des SW-Systems, die als Konsequenzen der Nutzung hierarchischer Strukturen und durch die Benutzt-Relation erzielte Kapselung angesehen werden können:

- Höhere Ebenen (Systemkomponenten) werden einfacher, weil sie tiefere Ebenen (Systemkomponenten) benutzen können.
- Änderungen auf höheren Ebenen sind ohne Einfluss auf tieferen Ebenen.
- Höhere Ebenen lassen sich abtrennen, tiefere Ebenen bleiben trotzdem funktionsfähig.
- Tiefere Ebenen können getestet werden, bevor die höheren lauffähig sind.

Weiterhin lässt sich jede Hierarchieebene als abstrakte oder virtuelle Maschine auffassen. Solche Abstraktionsebenen erlauben, dass

- Programme (Module) der Schicht i als abstrakte Maschine die Programme der Schicht i-1, die als Basismaschine dienen, benutzen und
- die abstrakte Maschine der Schicht i wiederum als Basismaschine für die Implementierung der abstrakten Maschine der Schicht i+1 dient.

(**Härder/Rahm**)

Zwei Anmerkungen:

1. Geheimnisprinzip $\Rightarrow$ Abstraktionsprinzip!
2. Abstraktionsebenen $\Rightarrow$ Sprachebenen?

Abstraktionsprinzip

Leibniz (Gleichheit in jeder Hinsicht von etwas Einmaligem, Individuen)

$$n = m =_{DF} \bigwedge_A (A(n) \leftrightarrow A(m))$$

(extensional identisch)

Lorenzen (Gleichheit von verschiedenen Objekten in gewisser, ausdrücklich mit angegebener Hinsicht)

$$\tilde{a} = \tilde{b} \leftrightarrow a \sim b \quad ; \quad a \sim b \leftrightarrow (P(a) \leftrightarrow P(b))$$

Beschränkung auf Prädikatoren (P), die in folgendem Sinne invariant sind:

(intensional identisch)

„façon de parler“ (Quine: Innocent Abstraction)

Eine Rede über abstrakte Objekte (ã) wird als eine besondere Rede – nämlich als eine invariante Rede – über konkrete Objekte (a) aufgefasst.

(**Wedekind/Ortner/Inhetveen**: Informatik als Grundbildung, Teil II: Gleichheit und Abstraktion, in: Informatik-Spektrum, Heft 3, 2004, S. 337-342.)

2.

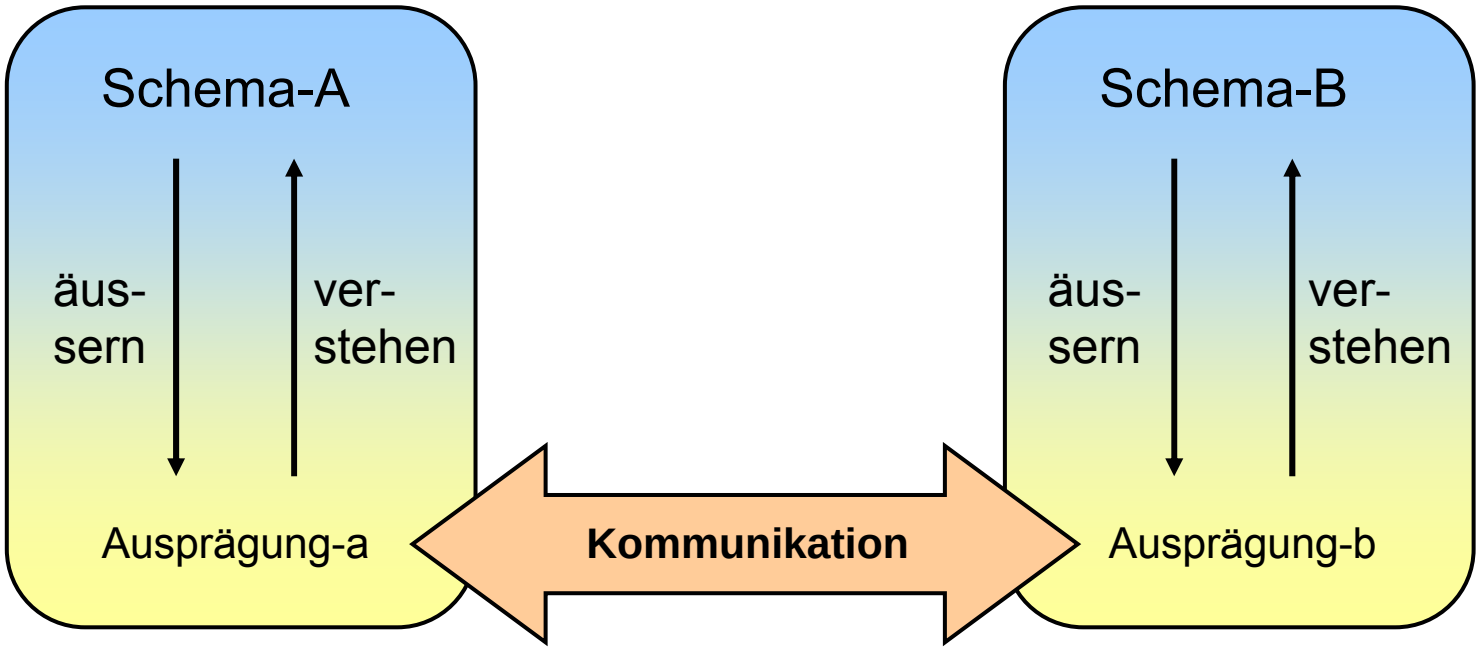
Sprachbasierte Architekturen

Lorenzscher „Fundamentalsatz“ der menschorientierten Symbolverarbeitung (dialogische Situation):

Von demjenigen, für den es die Handlung aktiv gibt – der sie „tut“ – sagt man auch, dass er ein „token“ oder eine „Aktualisierung“, die Handlung als etwas Einzelnes (Singulares; single act) erzeugt habe; von derjenigen, für die es die Handlung passiv gibt – die sie „erleidet“– , sagt man hingegen, dass sie ein „type“ oder ein „Schema“, die Handlung als etwas Allgemeines (Universales; generic action) erkannt habe.

Wir unterscheiden physische Handlungen, Trägerhandlungen und Sprachhandlungen (mit „denken“ als reden mit sich selbst).

Sprachlogischer Kommunikationsbegriff



Teilnehmer A **Teilnehmer B**

Schema und Ausprägung

- Ein Schema stellt den **universellen Aspekt** eines Objekts (Ding oder Geschehnis) dar

  ⇒ Ein Kunde hat eine Kundennummer.

- Eine Ausprägung stellt den **singulären Aspekt** eines Objekts (Ding oder Geschehnis) dar

  ⇒ 4711 ist ein Kunde.

(Schemata – wie Ausprägungen – sind „material“: Form + „Inhalt“)

Erklärung:

Im Kennen wird deutlich, dass Sein und Erkennen ebenso wie Einzelnes und Allgemeines jeweils aufeinander bezogen und unabhängig voneinander unbestimmbar sind: **Jedes Singulare ist nur als Aktualisierung eines Schemas verständlich, so wie jedes Universale nur als aktualisiertes Schema vorhanden ist.**

Kant:

„Diese Vorstellung nun von einem allgemeinen **Verfahren** der Einbildungskraft, einem Begriff sein Bild zu verschaffen, nenne ich das Schema zu diesem Begriff."

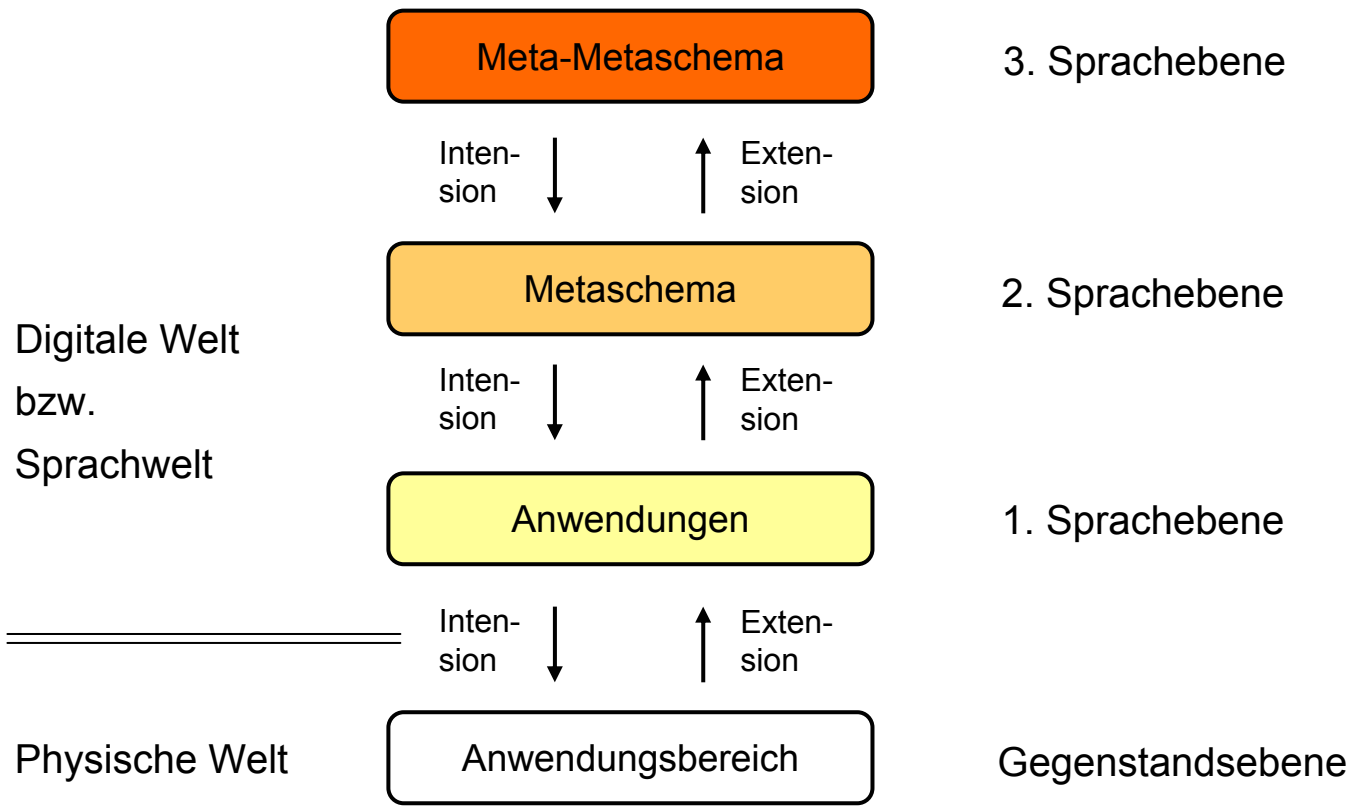
(KdrV, 1976, B180, S. 199)

I.A. an Kamlah/Lorenzen:

Indem ein **Geschehnis** (sprachlich) auf ein vereinbartes Schema gebracht wurde, ist es verfügbar wie ein verwendbares Gerät.

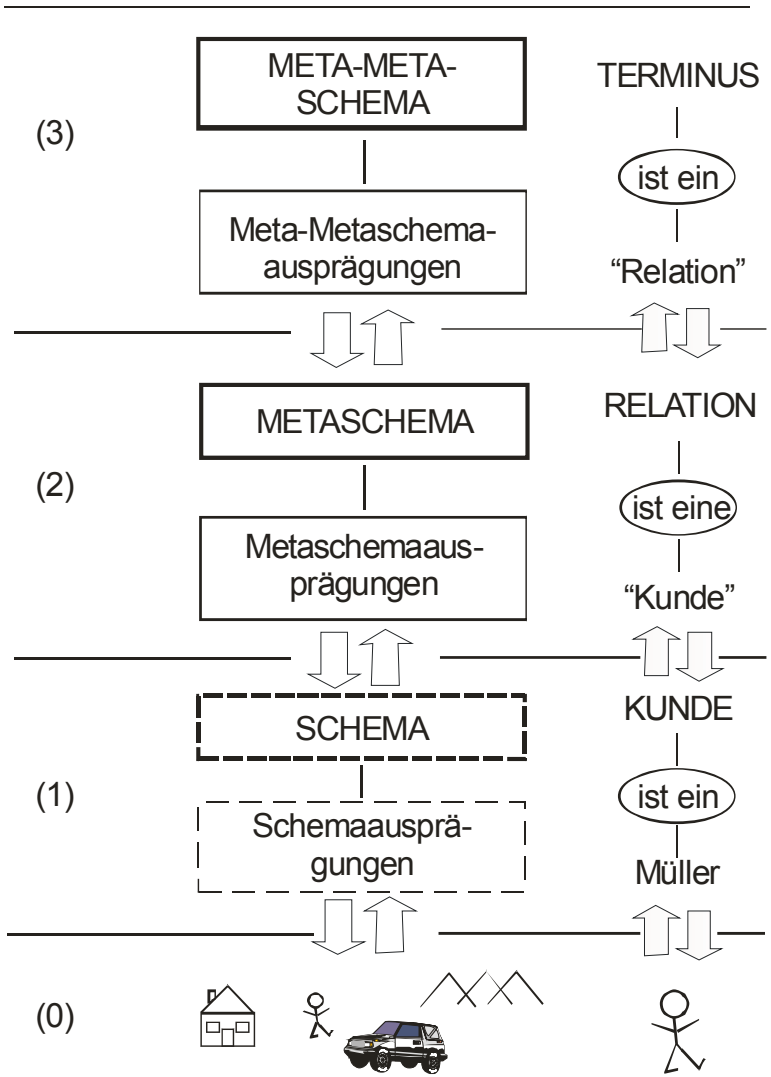
(LP, 1967, B.I. 227, S. 56)

Sprachebenen



(Intensionen ⇒ Schemata ; Extensionen ⇒ Ausprägungen, Objekte)

Sprachebenen-Architektur (Integration)



Legende:

: Referenzbeziehung (herstellend/ feststellend)

: Äußerungs-/Verstehensverbindung

ist ein : Subordination/singuläre Aussage/in-stance-of/Element von

: gehört nicht zum Inhalt des Repo-sitoriums

3. Sprachebene (Integration)

Komparativitätsgesetz:

Sind zwei Größen (a, b) einer dritten (u) gleich (R), so sind die auch untereinander gleich.

$$aRu \wedge bRu \longrightarrow aRb \wedge bRa$$

Prädikatorenregeln:

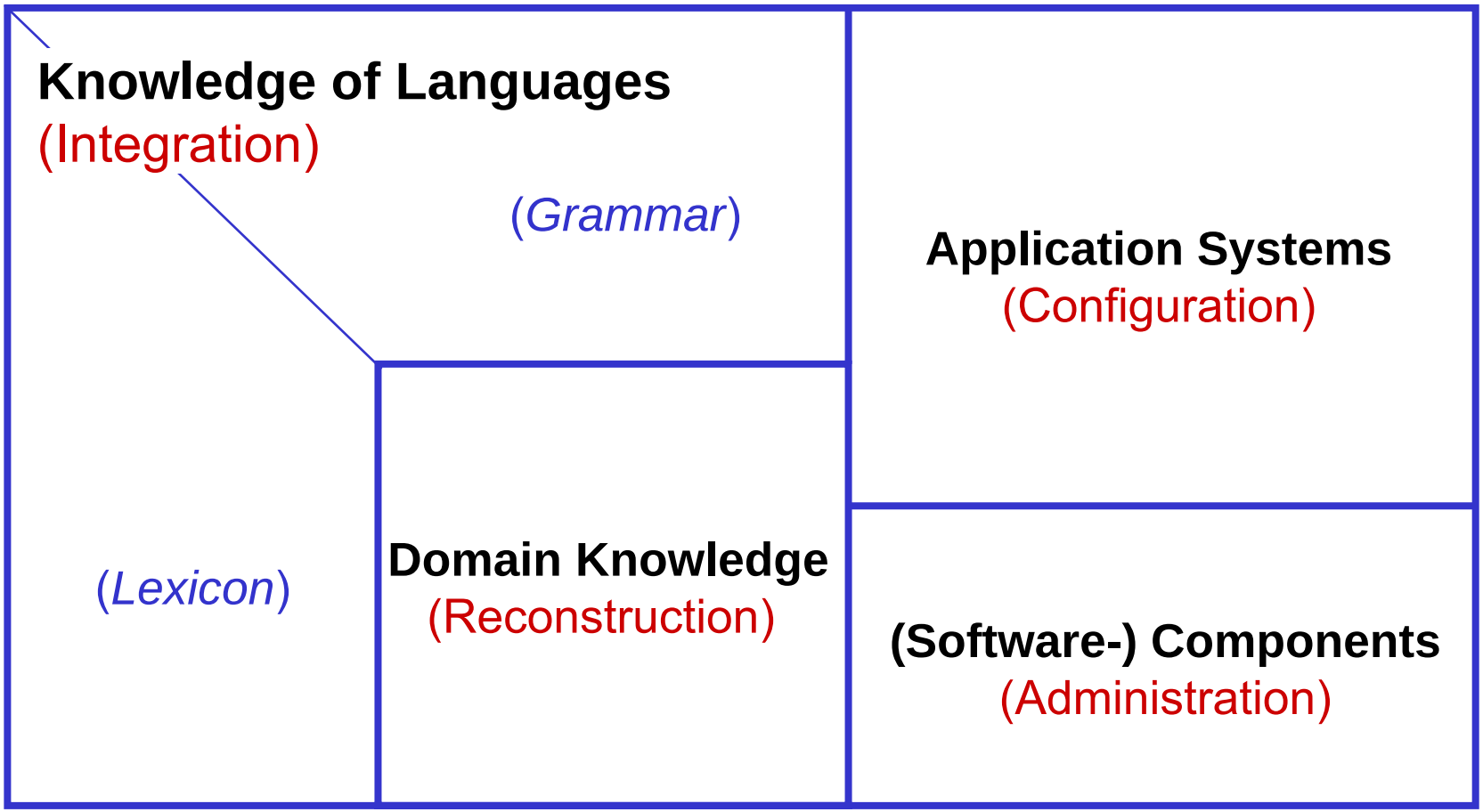
Kunde R abcd	$\Rightarrow$	Kunde R Customer
$\wedge$ Customer R abcd		$\wedge$ Customer R Kunde
$\wedge$ Client R abcd		$\wedge$ Kunde R Client
		$\wedge$ Client R Kunde
		$\wedge$ Customer R Client
		$\wedge$ Client R Customer

Rekonstruktion (Grammatik und Terminologie) einer Norm- oder Zwischensprache (Paul Lorenzen: Orthosprache, interlingual) für das Netz.

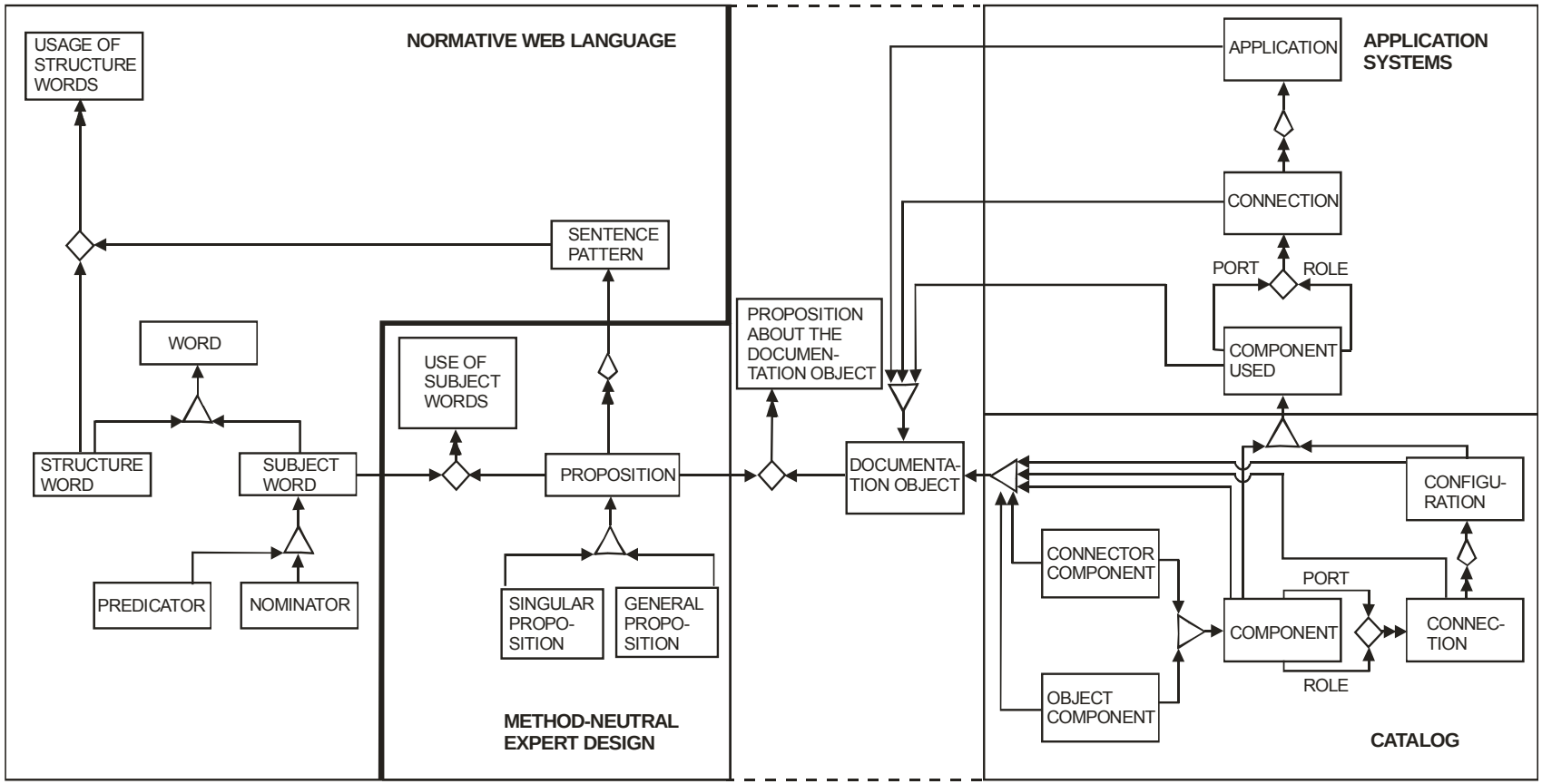
3.

Repository-Anwendungen

Repository Framework of an Electronic Marketplace for (Software-) Components



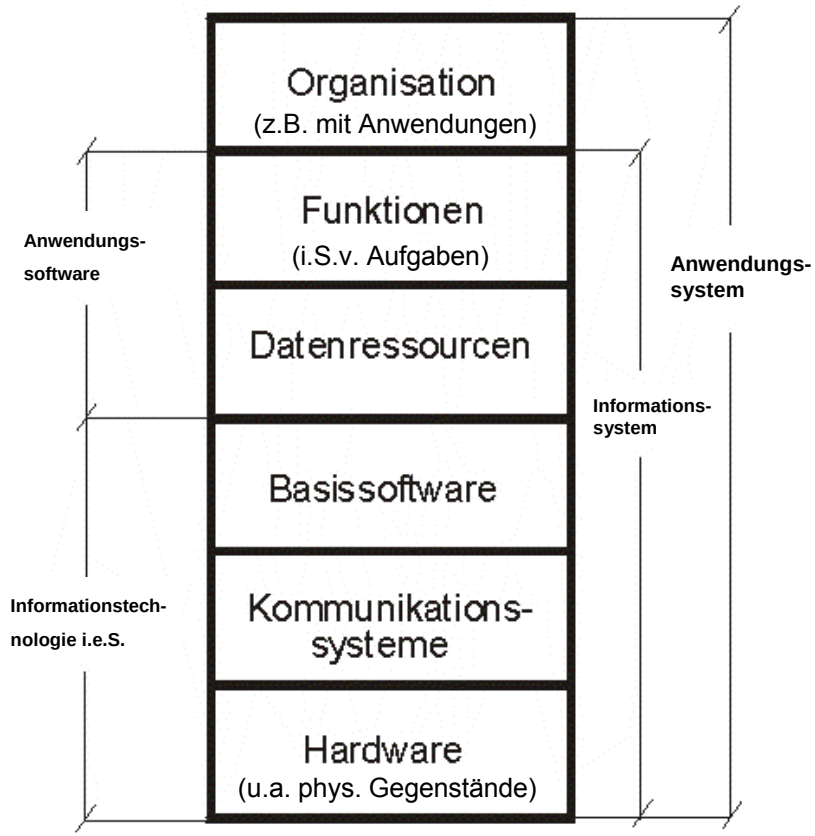
Meta Schema (Description Structure) of an Enterprise Repository



Quelle: ISTA 2005, Palmerston North, 23/5/2005 Erich Ortner, Tobias Grollius

Was ist ein Anwendungssystem?

Ein Anwendungssystem ist eine/ein **mit rechnerunterstützter Symbolverarbeitung versehene**(r) **Organisation** oder Gegenstand (Ding, Geschehnis) **jedweder Art**: eine Person, eine Familie, ein Unternehmen, ein Staat, eine Auto-fahrt, eine Auktion, ein Gebäude, eine Milchtüte, etc.

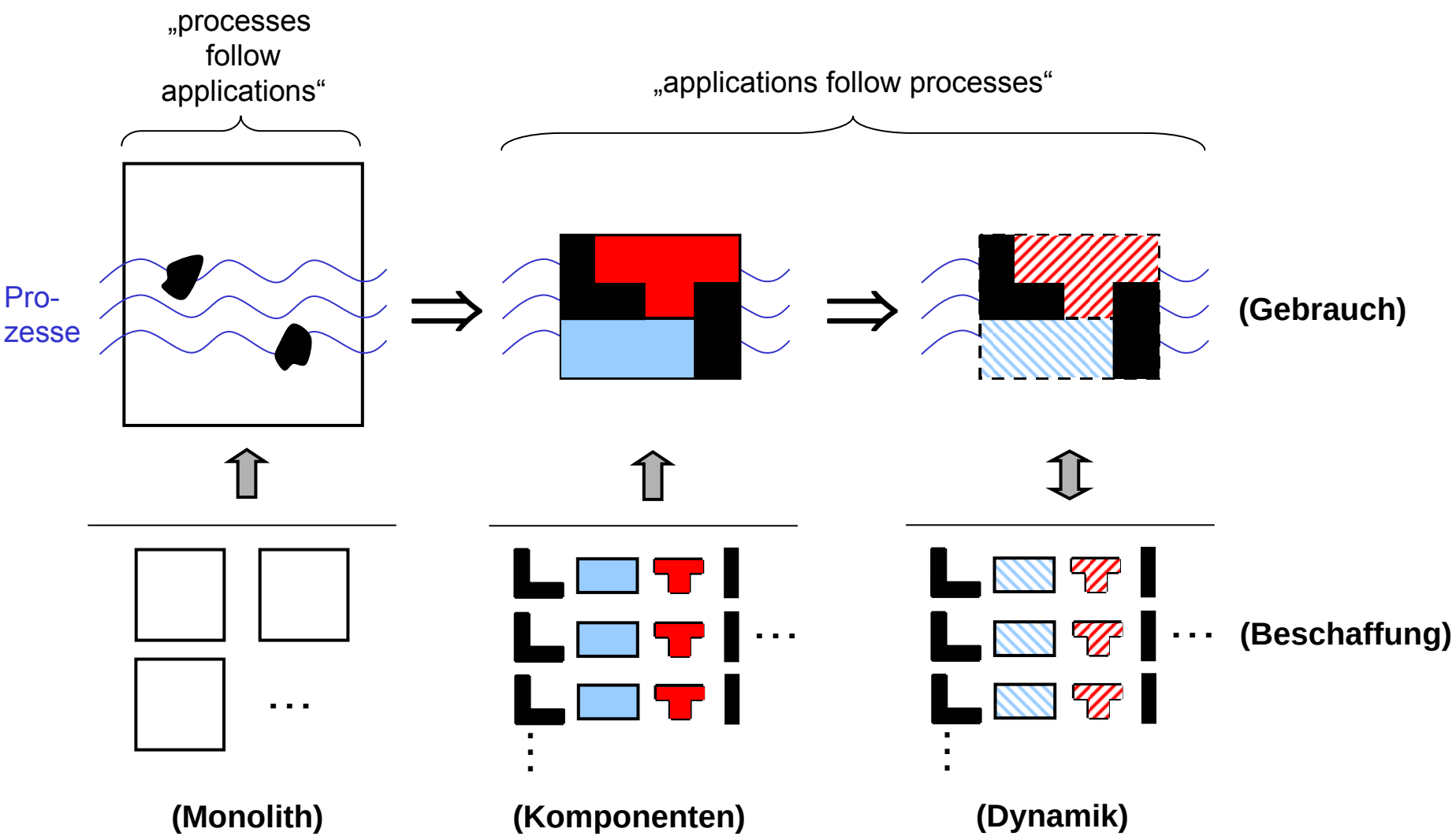


Dynamik und Ubiquität

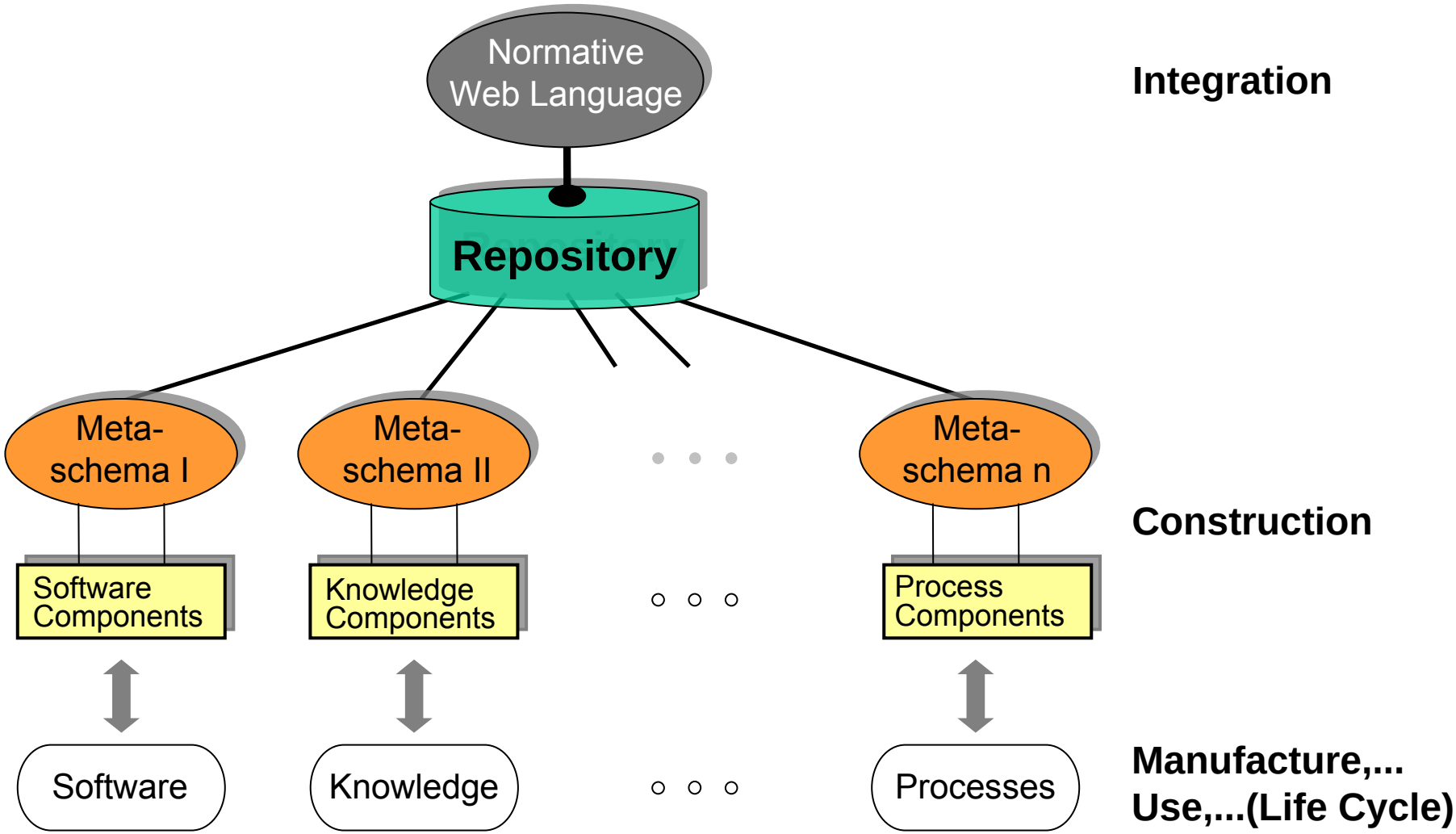
Dynamik: Vermögen, sich anzupassen $\Rightarrow$ Mobilität **nach innen** (einer Anwendung)

Ubiquität: Allgegenwart $\Rightarrow$ „Mobilität“ **nach außen** (einer Anwendung)

Was ist Anwendungssystemdynamik?



Workbench for Component-oriented, Self-configured, Integrated, Ubiquitous, … Application Systems



4.

Vortrag: Tobias Grollius