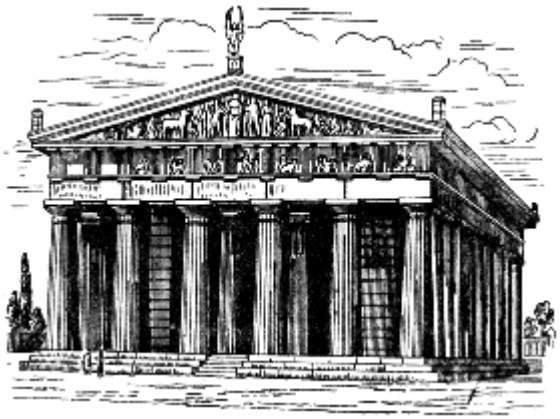
Softwarearchitektur

(Architektur: *αρχή* = Anfang, Ursprung + *tectum* = Haus, Dach)

2. Grundlagen der Softwarearchitektur



Vorlesung Wintersemester 2010 / 2011

Technische Universität München
Institut für Informatik
Lehrstuhl von Prof. Dr. Manfred Broy

Dr. Klaus Bergner, Prof. Dr. Manfred Broy, Dr. Marc Sihling

Inhalt

- Motivation und Darstellungsweise
  - Motivation und Darstellung
  - Komponenten und Instanzen
- Systemmodell für Architektur: Instanzen
  - Grundkonzepte und Strukturen
  - Zeit, Kommunikation und Interaktion
  - Strukturänderungen
  - Mobilität
- Systemmodell für Architektur: Beschreibungen
  - Definition und Bedeutung
  - Reflexivität
- Zusammenfassung

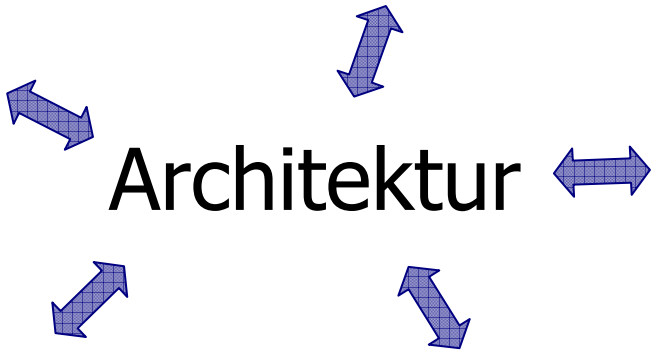
Inhalt

- Motivation und Darstellungsweise
  - Motivation und Darstellung
  - Komponenten und Instanzen
- Systemmodell für Architektur: Instanzen
  - Grundkonzepte und Strukturen
  - Zeit, Kommunikation und Interaktion
  - Strukturänderungen
  - Mobilität
- Systemmodell für Architektur: Beschreibungen
  - Definition und Bedeutung
  - Reflexivität
- Zusammenfassung

Wiederholung: Warum Softwarearchitektur

- Divide and Conquer (Teile und Herrsche):
  - Arbeitsteilung in der Projektorganisation
  - Komplexitätsreduktion durch Schnittstellenabstraktion und Kapselung
    - Austauschbarkeit, Portierbarkeit
    - Modularität
  - Beherrschbarkeit von Qualität
  - Integration
  - Wartung
- Wiederverwendung
  - Der Architektur (Referenzarchitektur)
  - Von Lösungsteilen (Design Patterns)
  - Von Komponenten
- Verteilung der Software auf der Hardware
  - Deployment

Einflussfaktoren



Architektur

Unterschiedliche Techniken und Komponenten

- COBOL, Java, .NET, ODBMS, XML, HTML, ...

Unterschiedliche Anwendungsgebiete

- Telekommunikation, eCommerce, Engineering,...

Unterschiedliche Programmierparadigmen

- Objektorientiert, Prozedural, Deklarativ, ...

Unterschiedliche Arten von Systemen

- Informationssysteme, eingebettete Systeme, mobile Systeme, ...

Unterschiedliche Methoden und Werkzeuge

- OOA/OOD, RUP, UML, CASE-Werkzeuge, IDEs, KM-Werkzeuge, ...

Problematik bei der Modellierung von SWAs

- SW-Architektur ist ein unreifes Gebiet
  - Grundlagen noch wenig wissenschaftlich aufgearbeitet
- Begriffe sind noch stark im Fluss
  - Oft angemessenen Abstraktionen noch nicht identifiziert
  - Konzepte stammen meist aus konkreten Systemen
  - Begriffe in unterschiedlichen Ansätzen unterschiedlich genutzt
  - Unterschiede werden betont (.NET versus Java)
- Darstellung erfolgt meist anhand von Beispielen
  - Irrelevante technische Details erschweren Verständnis
  - Übertragung auf andere Systeme schwierig
  - Darstellung kostet viel Zeit
- Zusammenhang zu Programmiersprachen oft ungenügend

Gesichtspunkte

- Daten - Kapselung
- Struktur: Gliederung des Systems
  - Horizontal: Aufspaltung eines Systems in Komponenten und ihre Verbindungen (Schnittstellen)
  - Vertikal: Weitere Untergliederungen der Komponenten
  - Es entsteht ein Struktur/Komponentenbaum (auch Struktur/Komponentenhierarchie genannt)
- Verhalten
  - Interaktion zwischen den Komponenten
  - Schnittstellenverhalten
- Verteilung

Darstellungsweise in der Vorlesung

- Definition eines abstrakten Systemmodells als Basis
  - Weglassen der technischen Details
  - Definition allgemeingültiger Begriffe und ihrer Zusammenhänge
  - Geeignet für sämtliche Anwendungsgebiete, Arten von Systemen, Programmierparadigma, etc.
- Bildung zusätzlicher Begriffs-Ebenen
  - Einführung zusätzlicher Konzepte auf Basis des Systemmodells
  - Angepasst an spezielle Anwendungsgebiete, Arten von Systemen, Programmierparadigma, etc.
- Veranschaulichung jeweils anhand konkreter Beispiele
  - Anwendung der Begriffe
  - Aufzeigen der technischen Umsetzung

Was ist ein Systemmodell?

Dient der Modellierung von Systemen:

- Definiert Grundkonzepte und ihre Zusammenhänge
  - Konzentration auf relevante Strukturen und Konzepte
  - Gibt ein bestimmtes Grundverständnis vor
- Abstrahiert von unwichtigen Details
  - Technische Details eines bestimmten Systems
  - Syntaktische Details einer bestimmten Beschreibung
- Verwendet mathematische Definitionen
  - Eindeutige und unmissverständliche Darstellung
  - Effiziente, sehr kompakte Darstellung
  - Möglichst einfach verständlich ☺
- Veranschaulichung durch Bilder und Diagramme

Begriff (Software-)System allgemein

Ein System hat

- eine festgelegte Systemgrenze
- die beschreibt, was Teil des Systems ist und was nicht (die Systemumgebung)
- über die Systemgrenze findet eine Wechselwirkung zwischen System und Umgebung statt
- die Wechselwirkung (auch beobachtbares Verhalten genannt) wird durch die Schnittstelle beschrieben

Schnittstelle eines Systems

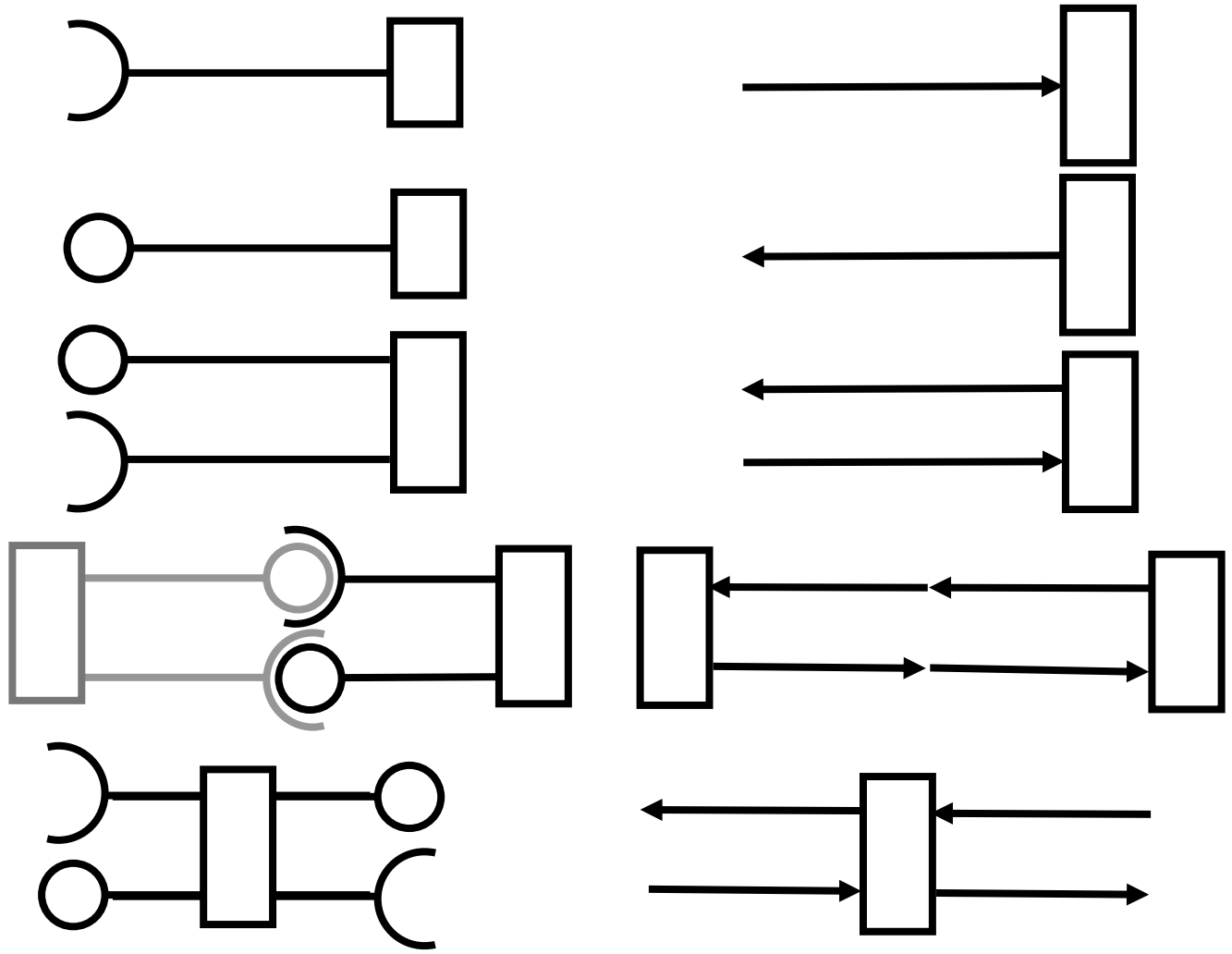
Die Schnittstelle gliedert sich in folgende Sichten

- syntaktische, statische Sicht
  - beschrieben durch die Möglichkeiten der Wechselwirkung („Interaktionsschritte“) des Systems mit seiner Umgebung
- semantische, dynamische Sicht (Schnittstellenverhalten)
  - beschrieben durch die Interaktionsfolgen zwischen System und seiner Umgebung

- Eine Schnittstelle kann in Unterschnittstellen gegliedert sein
  - Dann bietet ein System eine Familie von Teilschnittstellen an

Graphische Darstellung von Schnittstellen

- Eingabeport
- Ausgabeport
- Bidirektional
- Verbundene Ports
- Connector



Einschub: Familien von Systemen/Schnittstellen

- Sei eine Menge M von Systemen/Schnittstellen/... gegeben und eine Menge K von Bezeichnungen
- Eine Familie besteht aus einer Teilmenge

$$K' \subseteq K$$

und einer Abbildung

$$f: K' \rightarrow M$$

Architektur und die Rolle der Schnittstellen

- Aus Architektursicht sind wir primär an den Schnittstellen von Systemen und ihrer Komponenten interessiert
- Begründung
  - Die Architektur beschreibt die Gliederung eines Systems in Komponenten und deren Wirkung als Teil des Systems
  - Die Architektur hat zum Ziel die Schnittstelle des Systems nach außen zu realisieren
  - Die Wirkung der Komponenten als Teil der Architektur wird durch die Schnittstellen der Komponenten beschrieben

Arten von Schnittstellen - Schnittstellenkonzepte

Zusammenarbeit über „gemeinsamen Speicher“

- Gemeinsame Variable, die an der Schnittstelle gelesen oder geschrieben werden können
  - Syntaktische Schnittstelle: Variablen/Attribute an der Schnittstelle
  - Semantische Schnittstelle: mögliche Zustandsübergänge in Hinblick auf die Zustände der Variablen/Attribute an der Schnittstelle

Arten von Schnittstellen - Schnittstellenkonzepte

Zusammenarbeit über Methoden-/Prozeduraufrufe

- Schnittstelle:
  Mengen von Prozeduren oder Methoden -
  - oft gegliedert in Prozeduren/Methoden, die die über die Schnittstelle zur Verfügung gestellt werden an der Schnittstelle aufgerufen werden (Export) und in solche
  - die vom System nach außen (Import) aufgerufen werden
- Schnittstellensicht
  - Syntaktische Schnittstelle:
    Familien der Prozedur/Methodenkopfzeilen
    gegliedert in Export und Import
  - Semantische Schnittstelle:
    Folgen von Aufrufen und ihre Resultate

Arten von Schnittstellen - Schnittstellenkonzepte

- Mengen von Aktionen, die an den Schnittstellen ausführbar sind
  - Syntaktische Schnittstelle: Menge der Aktionen
  - Semantische Schnittstelle: Folgen von Aktionen - zusätzlich unter Umständen Angaben, nach welcher Folge von Aktionen welche Aktionen ausführbar sind und wann nicht

Arten von Schnittstellen - Schnittstellenkonzepte

- Datenfluss: Funktionen Datenströmen
- Datenfluss über Datenflussports - entspricht Maschinen mit Ein-/Ausgabe
  - Syntaktische Schnittstelle:
    Ports mit Angabe der Typen der Nachrichten, die über die Ports ausgetauscht werden (gegliedert in Ein- und Ausgabeports)
  - Semantische Schnittstelle:
    Relationen auf Folgen/Strömen von Daten auf den Ports - Zustandsmaschinen mit Ein-/Ausgabe

Gliederung von Schnittstellen

- Schnittstellen können in Unterschnittstellen gegliedert sein
- Syntaktisch heißt das, dass die syntaktischen Bestandteile der Schnittstellen in Teilmengen zerlegt werden
- Semantisch sind die Teilschnittstellen allerdings nicht einfach unabhängig beschreibbar, da es zwischen den Teilschnittstellen semantische Abhängigkeiten gibt

Schnittstellenkompatibilität

- Zwei (Unter-)Schnittstellen (zweier Systeme) heißen syntaktisch kompatibel, wenn die Systeme über die Schnittstellen zusammengesetzt/verbunden werden können

Architektur - Struktursicht - allgemein

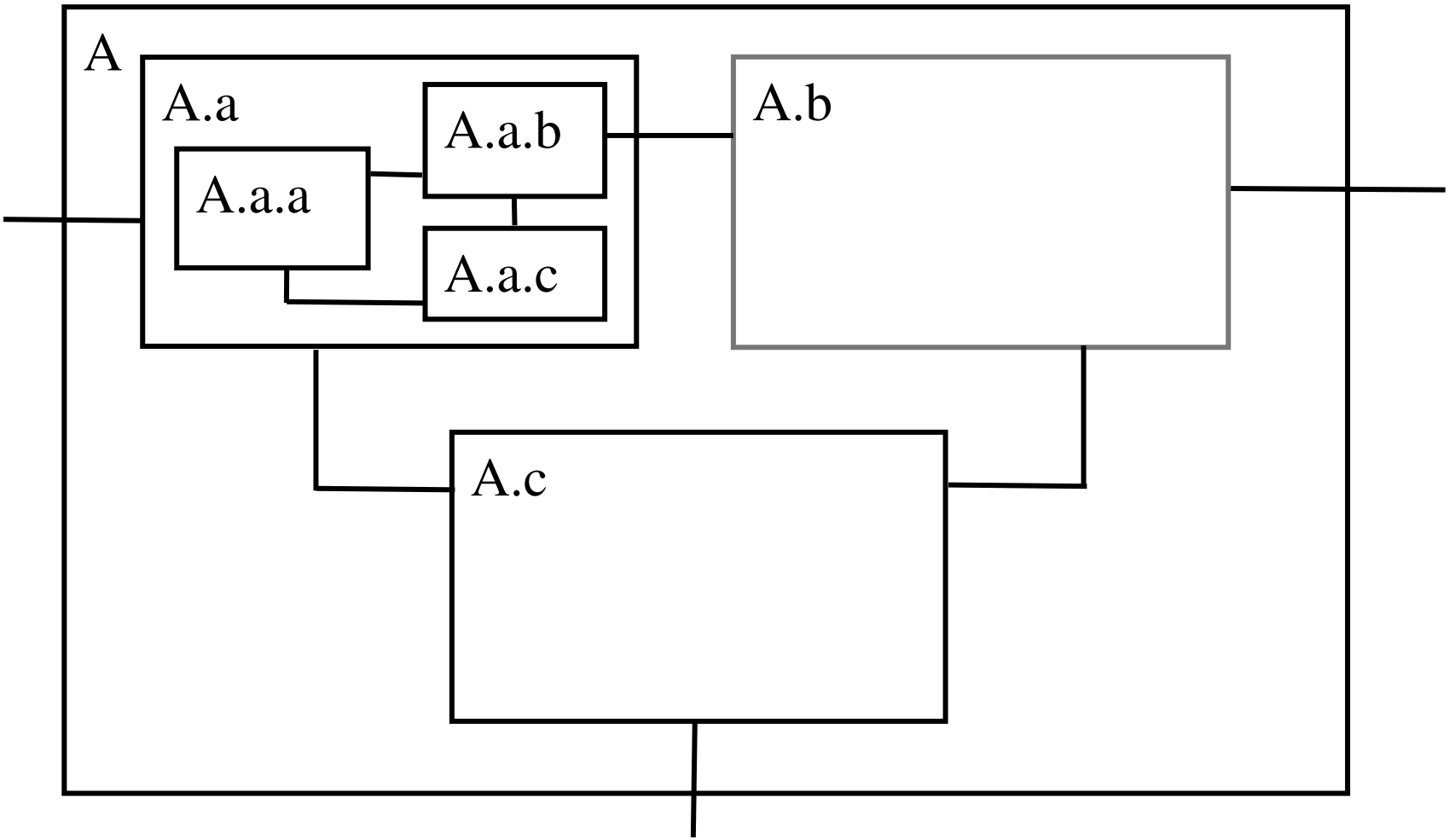
Eine Architektur für ein System gliedert ein System in

- eine Familie K von Komponenten (benannten Einheiten)
- eine Relation $R \subseteq K \times K$ zwischen den Komponenten
- stehen zwei Komponenten $(k, k') \in R$ in Relation zueinander, dann verfügen sie über eine gemeinsame verbundene Schnittstelle - bestehend aus kompatiblen Unterschnittstellen
- Zusätzlich existiert in der Regel Schnittstellen nach außen (zur „Umgebung")
  - Dazu kann man die Umgebung u zu K hinzunehmen
  - $(k, u) \in R$ bezeichnet dann eine Schnittstelle nach außen

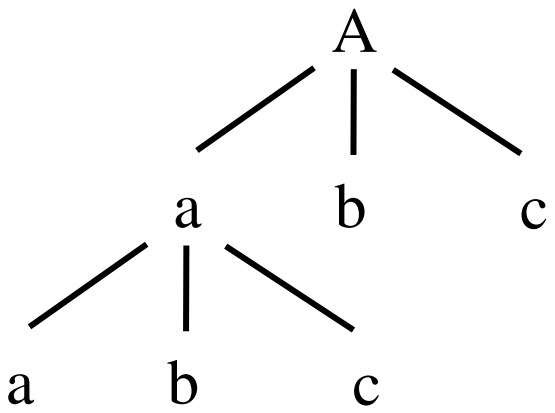
Hierarchie

- Ein Komponente ist wiederum selbst ein System
  - Ein System ist eine Komponente ist ein System
- Dadurch entsteht
  - eine Komponentenhierarchie (ein Komponentenbaum)
  - von in einander geschachtelten Komponenten

Architektur: Struktur



Komponentenbaum



Architektur und Komposition

- Eine Architektur beschreibt
  - eine Struktur
  - ein System-Verhalten: Schnittstellenverhalten
- Modularität:
  Das Schnittstellenverhalten InBeh(A) der Architektur A ist das Resultat der Komposition der Schnittstellerverhalten InBeh(A.?) seiner Komponenten

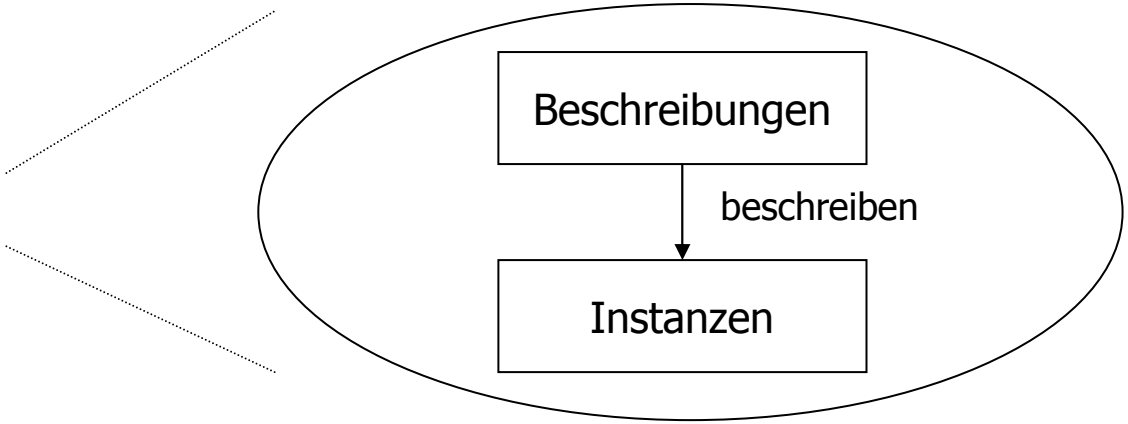
  $$InBeh(A) = InBeh(A.a) \otimes InBeh(A.b) \otimes InBeh(A.c)$$

  $$InBeh(A.a) = InBeh(A.a.a) \otimes InBeh(A.a.b) \otimes InBeh(A.a.c)$$
- Semantisch entspricht eine Architektur einer Komposition

Beschreibungen und Instanzen

- Architekten benötigen
  - Verständnis der Systemstrukturen und -Abläufe zur Laufzeit
  - Fähigkeit, diese Systeme auf unterschiedliche Art und Weise zu beschreiben
- Systemmodell enthält zwei Grundkonzepte
  - Komponenten-Beschreibungen repräsentieren Entwicklungssicht
  - Komponenten-Instanzen repräsentieren Laufzeitsicht

Mathematik
+ Bilder



Komponenten: Beschreibungen und Instanzen

Beschreibungen

- Komponenten mit deren Beschreibungen von Verhalten sind die Entwurfseinheiten zur Entwicklungszeit (Entwurfseinheiten).
- Entwurfseinheiten dienen der Strukturierung des Entwurfs.
- Entwurfseinheiten haben eine statische, gegebenenfalls hierarchische Struktur.
- Der Entwurf beschreibt die die Struktur und das Verhalten der Instanzen zur Ausführungszeit.

Instanzen

- Instanzen existieren zur Laufzeit. (Ausführungseinheiten)
- Ausführungseinheiten dienen der Organisation der Ausführung.
- In der Regel sind die Menge der Instanzen und ihre Verbindungen dynamisch (ändern sich während der Ausführungszeit).
- Das Verhalten der Instanzen wird durch die Entwurfs-einheiten beschrieben.

Beschreibung und Instanz

- Es gibt eine reiche Vielfalt unterschiedlicher Konzepte von Komponenten und ihres Verhaltens:
  - Funktion
  - Klasse
  - Zustandsmaschine
  - …
- Auf der Menge der Verhalten sind Kompositions-Operationen definiert (aus gegebenen Verhaltensbeschreibungen lässt sich eine komplexere Verhaltensbeschreibung zusammensetzen).
- Es gibt eine reiche Vielfalt von Möglichkeiten, Verhalten zu beschreiben:
  - Programmiersprachen: z.B. Assembler, C, Java
  - Modellierungssprachen: z.B. UML
  - Spezifikationen: z.B. Z, Zustandsmaschinen, Logik OCL

Beschreibung und Instanz

Eine Instanz entsteht in der Regel zur Laufzeit aus einer Entwurfseinheit:

- Eine Instanz besitzt
  - eine Identität (einen Identifikator)
  - einen Zustand (genauer Zustandsraum)
  - ein Laufzeit-Verhalten
- Zustände umfassen
  - lokale („verkapselte“) Daten- und Kontrollzustände
  - Verbindungen zu anderen Instanzen
  - enthaltene andere Instanzen
- Das Laufzeit-Verhalten umfasst
  - die Kommunikation mit anderen Instanzen
  - die Änderung des Zustands über die Zeit hinweg
  - Lokalisierung auf Hardwarekomponenten (Verteilung, Deployment)

Systeme: Mengen von Beschreibungen und Instanzen

- Ein Systembeschreibung umfasst
  - eine Menge von Verhaltensbeschreibungen (Entwurfseinheiten)
  - eine Beschreibung, wie aus den Verhaltensbeschreibungen (Entwurfseinheiten) Instanzen gebildet werden und wie diese verbunden sind (wie das System sich aus Instanzen zusammensetzt)
- Zur Ausführungszeit (Laufzeit) besteht ein System zu jedem Zeitpunkt aus
  - einer Menge von Instanzen
  - der Struktur ihrer Verbindungen
- Der Systemzustand ist zu jedem Zeitpunkt gegeben durch
  - die Menge der existierenden Instanzen und ihren Datenzuständen
  - die Nachrichten, die über die Verbindungen und Schnittstellen laufen
  - die Struktur der Verbindungen

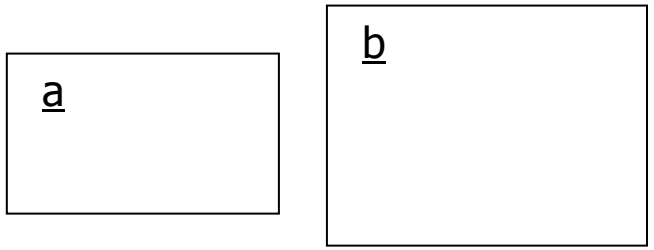
Inhalt

- Motivation und Darstellungsweise
  - Motivation und Darstellung
  - Komponenten und Instanzen
- Systemmodell für Architektur: Instanzen
  - Grundkonzepte und Strukturen
  - Zeit, Kommunikation und Interaktion
  - Strukturänderungen
  - Mobilität
- Systemmodell für Architektur: Beschreibungen
  - Definition und Bedeutung
  - Reflexivität
- Zusammenfassung

k4

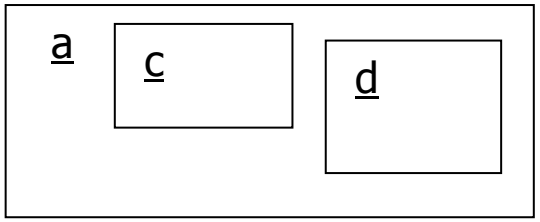
Komponenten-Instanzen

- Ein System besteht zur Laufzeit aus einer Menge von **Komponenten-Instanzen**.
- Definition:
  - COMPONENT: (unendliche) Menge aller nur möglichen („potentieller) Komponenteninstanzen (Identifikatoren und Verhalten)
- Beispiel:
  - $a, b \in \text{COMPONENT}$



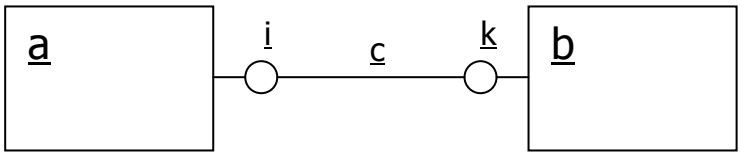
Komponentenhierarchie

- Komponenten(instanzen) können hierarchisch in andere Komponenten(instanzen) **zerlegt** bzw. aus anderen Komponenten(instanzen) **zusammengesetzt** sein.
- Definition:
  - parent : COMPONENT $\rightarrow$ COMPONENT
    Partielle Abbildung, ordnet einer Instanz ihre umschließende Eltern-Instanz zu. Instanzen auf oberster Ebene müssen keine Eltern-Instanz haben.
- Konsistenzbedingung:
  - parent ist azyklisch
- Beispiel:
  - parent(c) = parent(d) = a
  - parent(a) nicht definiert



Verbindungen und Schnittstellen

- **Komponenten**(instanzen) besitzen **Schnittstellen**(instanzen).
- Schnittstelleninstanzen sind untereinander **verbunden**.
- Definitionen:
  - INTERFACE : Menge aller Schnittstelleninstanzen
  - assigned : INTERFACE $\rightarrow$ COMPONENT
    Totale Abbildung, ordnet jeder Schnittstelle die Komponente zu, der sie zugeordnet ist. Freie Schnittstellen sind nicht erlaubt.
  - CONNECTION : Menge aller Verbindungsinstanzen
  - connects : CONNECTION $\rightarrow$ INTERFACE $\times$ INTERFACE
    Totale Abbildung, gibt für jede Verbindung an, welche Schnittstellen sie verbindet. Eine Schnittstelle kann viele Verbindungen haben.

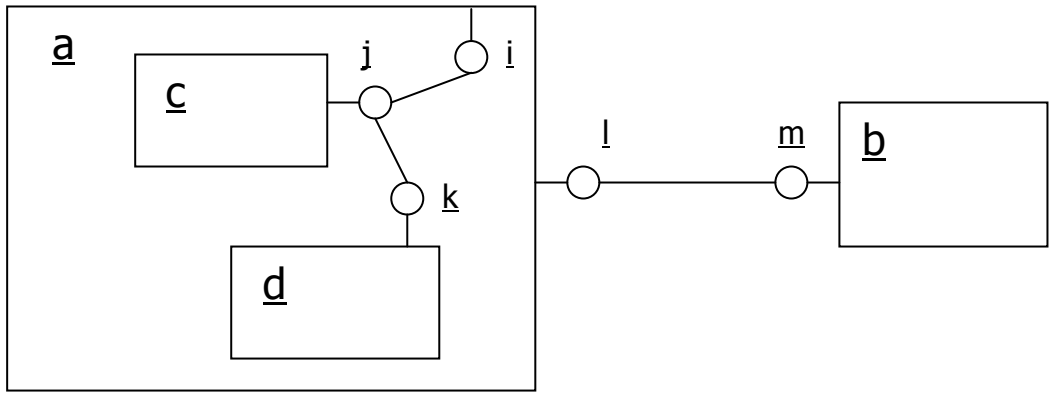


Systeme als Strukturen

- Komponenten, Schnittstellen und Verbindungen geben die Struktur eines Systems vor.
- Die Struktur ist einer der wichtigsten Aspekte bei der Betrachtung der Architektur eines Systems.
- Definitionen:
  - $\mathrm{INSTANCE} = \mathrm{COMPONENT} \cup \mathrm{INTERFACE} \cup \mathrm{CONNECTION}$
    Menge aller möglichen Instanzen
  - $\mathrm{alive} : \mathrm{INSTANCE} \rightarrow \mathrm{Boolean}$
    Totale Abbildung; gibt an, ob eine Instanz für die Systemstruktur des betrachteten Systems relevant/existent ist.
  - $\mathrm{STRUCTURE} = \mathrm{ASSIGNED} \times \mathrm{PARENT} \times \mathrm{CONNECTS} \times \mathrm{ALIVE}$
    Menge aller möglichen Systemstrukturen, wobei PARENT die Menge aller Funktionen parent ist, analog für $\mathrm{CONNECTS}$ und ALIVE.

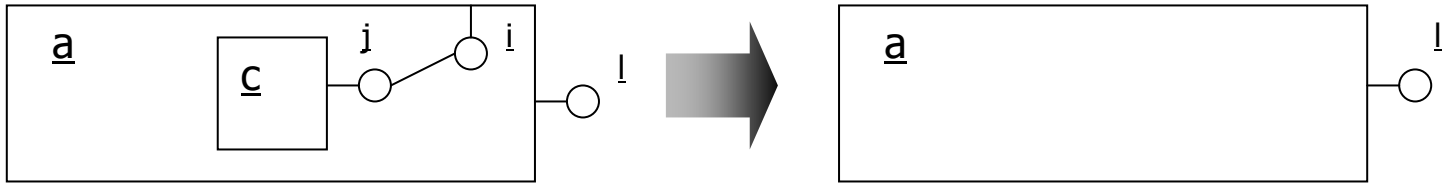
Systemstruktur - Beispiel

- a, b, c, d ∈ COMPONENT; i, j, k, l, m ∈ INTERFACE
- v, w, x ∈ CONNECTION
- ∀α ∈ INSTANCE : alive(α) ⇔ α ∈ {a, b, c, d, i, j, k, l, m, v, w, x}
- parent(c) = parent(d) = a
- assigned(i) = assigned(l) = a; assigned(m) = b
- assigned(j) = c; assigned(k) = d
- connects(v) = (i, j); connects(w) = (j, k); connects(x) = (l, m)



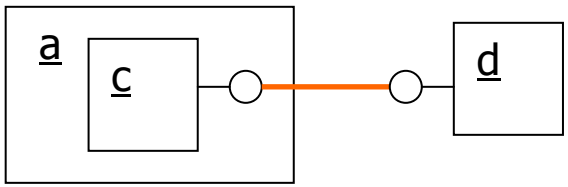
Schnittstellen und Implementierung

- Die Hierarchie und die Schnittstellen erlauben es, innere Strukturen auszublenden (Geheimnisprinzip).



white box *black box*

- Verbindungen dürfen deshalb keine Komponentengrenzen kreuzen
  - $\text{connects}(v) = (i, j) \wedge \text{assigned}(i) = c \wedge \text{assigned}(j) = d \Rightarrow$
    $c = d \vee \text{parent}(c) = d \vee \text{parent}(d) = c \vee \text{parent}(c) = \text{parent}(d)$



Architekturprinzipien zur Komplexitätsreduzierung

- Geheimnisprinzip - Information Hiding: Interne Strukturen und Abläufe werden vor der Außenwelt versteckt und sind nur über Schnittstellen zugreifbar.
  - Interne Strukturen sind für das Verständnis des Zusammenspiels der Komponente mit anderen Komponenten nicht relevant.
  - Interne Strukturen und Abläufe können geändert werden, ohne dass sich das Verhalten der Komponente nach außen ändert.
- Lose Kopplung – starke Bindung: Eine Komponente sollte möglichst viel Komplexität kapseln, indem sie
  - einen starken inneren Zusammenhang hat und
  - möglichst wenige, schmale Schnittstellen nach außen bereitstellt.

Inhalt

- Motivation und Darstellungsweise
  - Motivation und Darstellung
  - Komponenten und Instanzen
- Systemmodell für Architektur: Instanzen
  - Grundkonzepte und Strukturen
  - Zeit, Kommunikation und Interaktion
  - Strukturänderungen
  - Mobilität
- Systemmodell für Architektur: Beschreibungen
  - Definition und Bedeutung
  - Reflexivität
- Zusammenfassung

Verhaltensmodelle

- Grundsätzliche Ausführungskonzepte
  - Sequenzieller Kontrollfluss
  - Paralleler Kontrollfluss
- Grundsätzliche Interaktionskonzepte für Komponenten
  - gemeinsamer Speicher
    - gemeinsame Programmvariablen
  - gekapselter Speicher
    - Methoden/Prozeduraufruf
    - Nachrichtenaustausch

Verhaltensmodelle für Komponenten

- Zustandsmaschinen
  - Das Verhalten von Komponenten kann durch Zustandsmaschinen mit Ein-/Ausgabe modelliert werden
- Schnittstellenmodelle
  - Erfassung der Interaktion durch Daten-/Nachrichtenströme

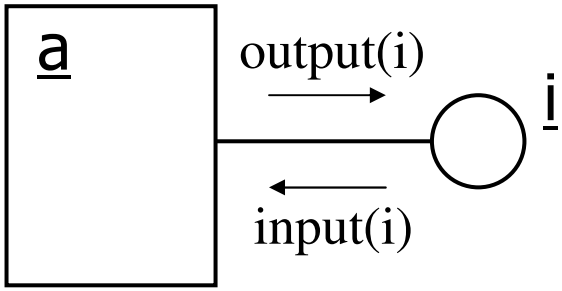
Verhalten und Zeitströme

- Bis jetzt haben wir nur statische Strukturen beschrieben … Verhalten bedeutet aber, dass es Veränderungen über die Zeit hinweg gibt.
- Was ist Zeit? Zeit hat in dem Modell folgende Eigenschaften
  - Es gibt eine globale Zeit, die überall gleich schnell läuft.
  - Es gibt eine kleinste, atomare Zeiteinheit.
  - Annahme dabei: Wählen wir die Zeiteinheit klein genug, dann können wir alles mitbekommen (analog wie beim Sampling von Tonfrequenzen).
- Wir betrachten Zeitströme als Sequenzen von einzelnen Zeitschritten und definieren:
  - Sei X eine Menge von Signalen, Nachrichten, Ereignissen, Zuständen.
  - $T = \{t_0, t_1, t_2, t_3, \ldots\}$: Zeit - Menge aller betrachteten Zeitpunkte
  - $X^T = T \rightarrow X$: Zeitstrom der Elemente aus der Menge X
  - x_t : das Element aus X, das in X^T dem Zeitpunkt t zugeordnet ist

Kommunikation und Interaktion I

- Miteinander verbundene Komponenteninstanzen können kommunizieren, indem sie Nachrichten austauschen.
- Nachrichten fließen von Schnittstellen zu Komponenten und umgekehrt.
- Definitionen:
  - $\mathrm{MESSAGE}$: Menge aller möglichen Nachrichten
  - $\varepsilon \in \mathrm{MESSAGE}$: leere Nachricht
  - $\mathrm{input} : \mathrm{INTERFACE} \rightarrow \mathrm{MESSAGE}$
  - $\mathrm{output} : \mathrm{INTERFACE} \rightarrow \mathrm{MESSAGE}$

Partielle Abbildungen; geben an, welche Nachrichten in eine bzw. aus einer Komponente fließen. Definition von Zeitströmen input^T und output^T wie eben.

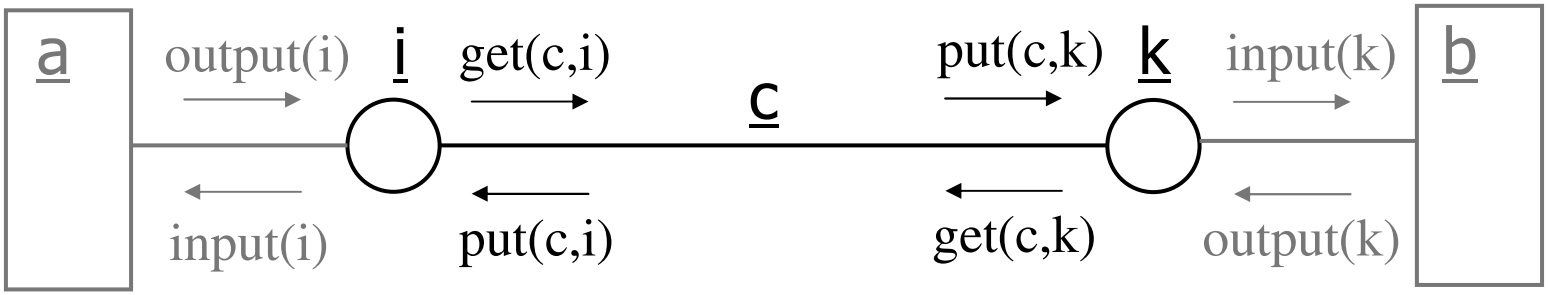


Kommunikation und Interaktion II: Connectoren

- Nachrichten fließen von Schnittstellen in Verbindungen und umgekehrt.
- Definitionen:
  - $\text{get} : \text{CONNECTION} \times \text{INTERFACE} \rightarrow \text{MESSAGE}$
  - $\text{put}: \text{CONNECTION} \times \text{INTERFACE} \rightarrow \text{MESSAGE}$

  Partielle Abbildungen; geben an, welche Nachrichten in eine bzw. aus einer Verbindung fließen.

  Definition von Zeitströmen get^T und put^T.



Verhalten von Verbindungsinstanzen I

- Das Verhalten einer Verbindungsinstanz lässt sich als Teilmenge aller Zeitströme der Ein- und Ausgaben beschreiben (genauer: aller Zeitströme der get- und put- Abbildungen, die diese Ein- und Ausgaben festlegen).
- Definitionen:
  - $\text{behavior} : \text{CONNECTION} \rightarrow \wp(\text{get}^T \times \text{put}^T)$
- Lokalität: Das Verhalten einer Verbindung c macht nur Aussagen über Nachrichten auf der Verbindung c (und nicht über Nachrichten auf anderen Verbindungen).
  - $(\text{gets}, \text{puts}) \in \text{behavior}(c) \wedge \text{connected}(c) = (i, k) \Rightarrow$
    $\forall t \in T: \quad \text{gets}_t \in \{ \{(c, i), (c, k)\} \rightarrow M \}$
    $\wedge \ \text{puts}_t \in \{ \{(c, i), (c, k)\} \rightarrow M \}$

s für Zeit**s**trom

Verhalten von Verbindungsinstanzen II

- Kausalität: Die Ausgaben einer Verbindung c zu einem bestimmten Zeitpunkt t können nur von den Eingaben bis zum Zeitpunkt t-1 abhängen (und insbesondere nicht von zukünftigen Eingaben).
  - $(gets1, puts1) \in behavior(c) \wedge (gets2, puts2) \in behavior(c) \wedge gets1{\downarrow}t\text{-}1 = gets2{\downarrow}t\text{-}1 \Rightarrow puts1_t = puts2_t$

    Notation: x↓t bezeichnet den Teil des Zeitstroms x bis einschließlich Zeitpunkt t
  - Die Formel bezeichnet den deterministischen Fall, bei dem die Eingabe bis zu einem Zeitpunkt die Ausgabe im nächsten Zeitschritt festlegt.
  - Im nichtdeterministischen Fall (mehrere Ausgabeströme zu einem Eingabestrom) müssen die Mengen der möglichen zur Zeit t ausgegebenen Zeichen gleich sein.

Verhalten von Verbindungsinstanzen III

- Selbst unter Berücksichtigung von Lokalität und Kausalität gibt es noch viele Möglichkeiten, wie sich eine Verbindung verhalten kann:
  - Sie kann Nachrichten verlieren oder nicht.
  - Sie kann Nachrichten verfälschen / umwandeln oder nicht.
  - Sie kann selbständig Nachrichten erzeugen oder nicht.
  - Sie kann Nachrichten puffern oder nicht.
  - Sie kann die Reihenfolge von Nachrichten bewahren oder nicht.
  - Sie kann unterschiedliche Laufzeiten für Nachrichten bieten (ggf. abhängig von der Größe der Nachrichten).
  - Sie kann Nachrichten nur auf Anforderung ausliefern.
- Der Architekt muss die Eigenschaften von Verbindungen in seinem Laufzeitsystem kennen, um die Architektur einer Anwendung erarbeiten zu können.

Inhalt

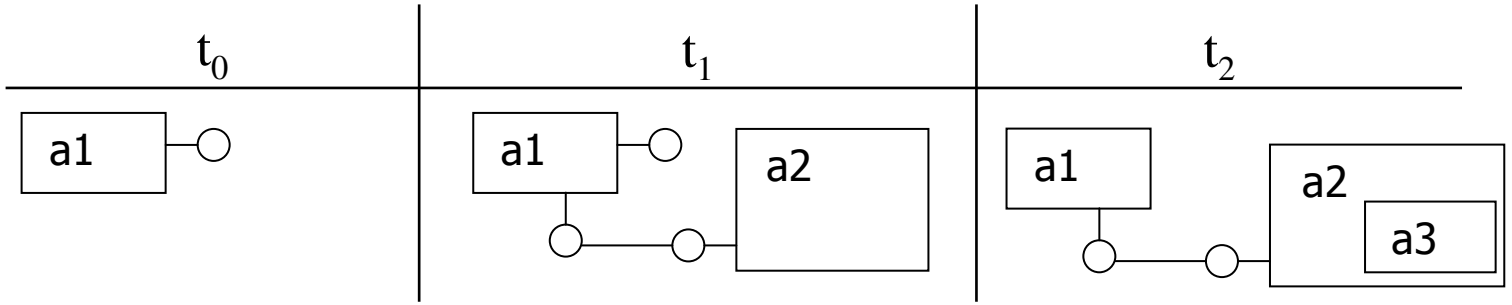
- Motivation und Darstellungsweise
  - Motivation und Darstellung
  - Komponenten und Instanzen
- Systemmodell für Architektur: Instanzen
  - Grundkonzepte und Strukturen
  - Zeit, Kommunikation und Interaktion
  - Strukturänderungen
  - Mobilität
- Systemmodell für Architektur: Beschreibungen
  - Definition und Bedeutung
  - Reflexivität
- Zusammenfassung

Strukturänderungen

- Bis jetzt haben wir nur statische Systemstrukturen betrachtet.
- Die Struktur vieler Systeme ist aber dynamisch:
  - Komponenten und Schnittstellen kommen hinzu oder fallen weg
  - Verbindungen werden gezogen oder gelöscht
  - Systemstrukturen ändern sich
- Die Strukturänderungen können dabei unterschiedliche Ursachen haben:
  - Das System selbst kann seine Struktur programmgesteuert verändern.
  - Instanzen des Systems können ausfallen und so die Struktur verändern.

Modellierung der Strukturänderungen

- Definitionen:
  - $\text{STRUCTURE}^T = \text{ALIVE}^T \times \text{ASSIGNED}^T \times \text{PARENT}^T \times \text{CONNECTS}^T$
- Ein Element aus STRUCTURE^T beschreibt einen möglichen Systemablauf aus Struktursicht.
- Beispiel:



Zwischenstand I

- Bis jetzt haben wir ein grundlegendes Vokabular für Architektur definiert.
  - Instanzen von Komponenten, Schnittstellen und Verbindungen
  - Strukturen aus Instanzen sowie Strukturänderungen
  - Kommunikation zwischen Komponenten über Schnittstellen und Verbindungen
  - Verhalten einzelner Instanzen sowie Komposition von Instanzen
- These: Das grundlegende Systemmodell ist allgemein genug, um über die Architektur aller möglichen Software-Systeme reden zu können.
- Es definiert also allgemeine Eigenschaften des Laufzeitsystems einer „Architekturmaschine“.

Zwischenstand II

- Auf Basis des grundsätzlichen Laufzeitsystems lassen sich eine Reihe von zusätzlichen Definitionen aufsetzen, die spezielle Architekturen beschreiben, zum Beispiel:
  - Nachrichtenlaufzeiten von Verbindungen
  - Schnittstellen, die als Nachrichtenverteiler dienen
  - Komponenten, die andere Komponenten erzeugen können
- Derartige zusätzliche Annahmen müssen explizit gemacht werden, wenn man über eine Architektur spricht.
- Die bisher gezeigten Konzepte sollen insbesondere dazu dienen, den Blick dafür zu schärfen, auf welche Aspekte man achten muss.

Inhalt

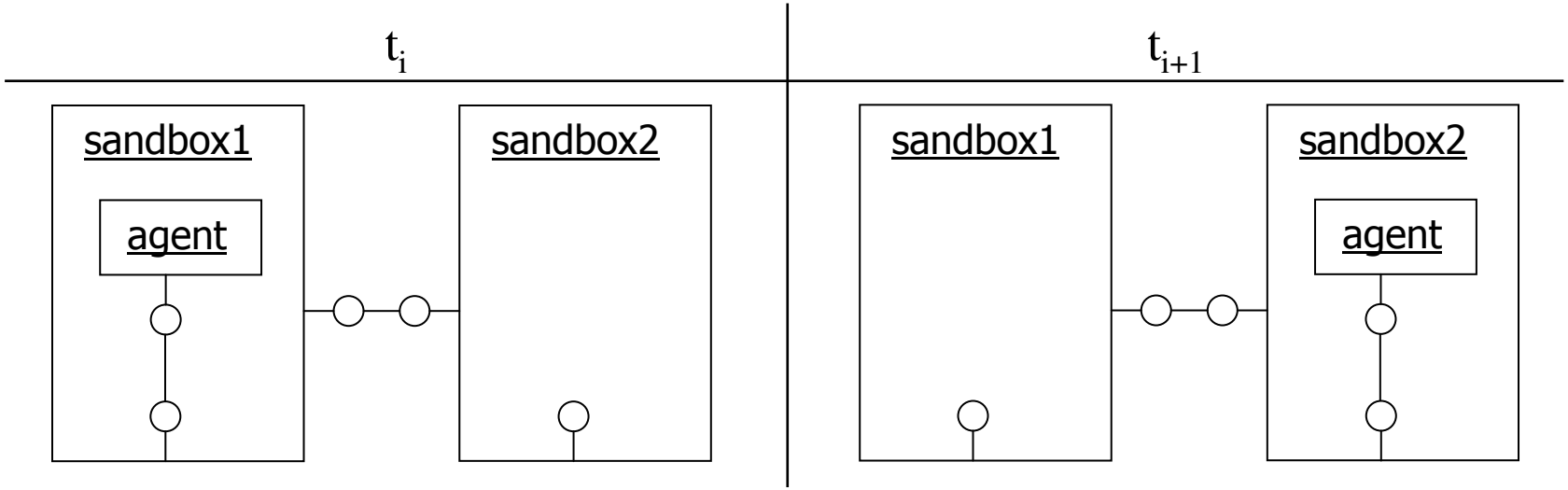
- Motivation und Darstellungsweise
  - Motivation und Darstellung
  - Komponenten und Instanzen
- Systemmodell für Architektur: Instanzen
  - Grundkonzepte und Strukturen
  - Zeit, Kommunikation und Interaktion
  - Strukturänderungen
  - Mobilität
- Systemmodell für Architektur: Beschreibungen
  - Definition und Bedeutung
  - Reflexivität
- Zusammenfassung

Mobilität

- Naive Definition: Eine Komponente ist mobil, wenn sie innerhalb eines Systems ihren Ort wechseln kann.
- Beispiele:
  - Ein mobiler Software-Agent, der innerhalb eines Netzwerks von einem Rechner zum nächsten wandern kann.
  - Ein Handy (Cell Phone) mitsamt seiner Software, das innerhalb des GSM-Netzes seine Funkzelle wechselt.
- Mobilität lässt sich auf mehrere Arten definieren.
  - Implizit: Wir fassen Orte als (ggf. spezielle) Komponenten auf. Mobilität lässt sich dann als zeitliche Änderung der Abbildung parent begreifen.
  - Explizit: Wir führen Orte als explizites Konzept neben den anderen Basiskonzepten ein.

Mobilität – implizite Variante

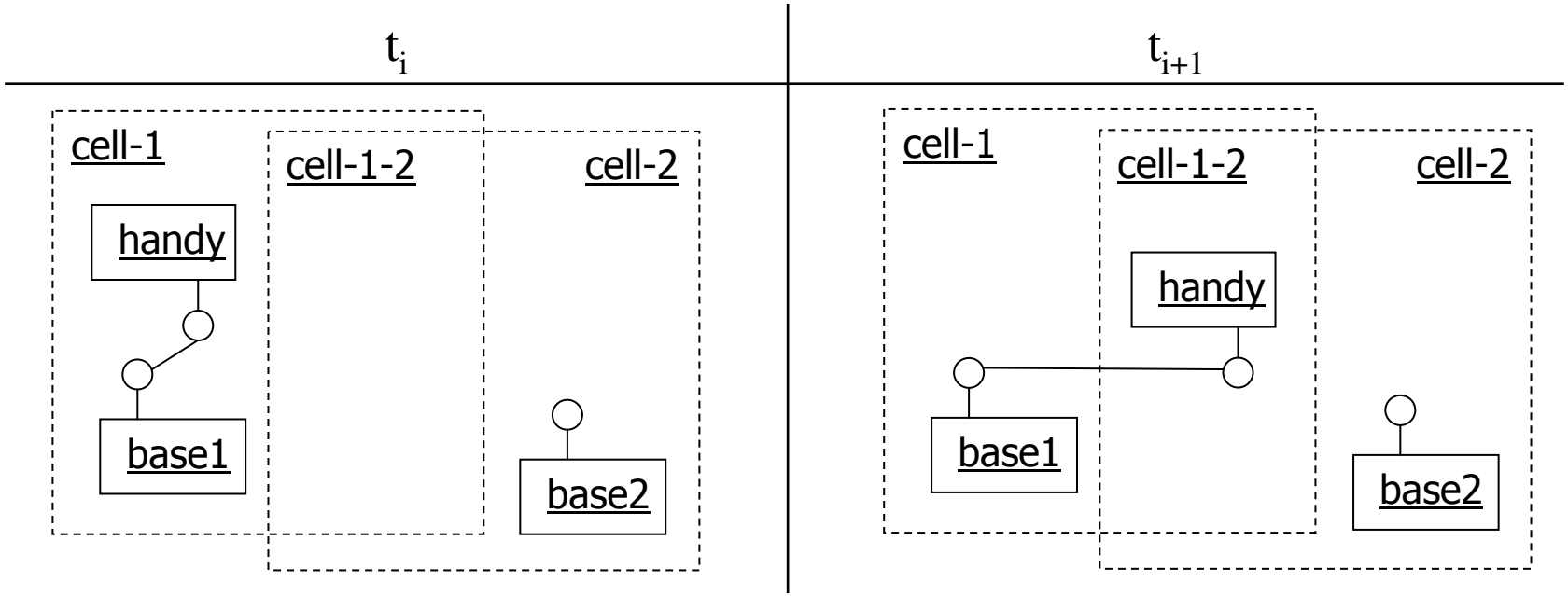
- Orte sind Komponenten
- Mobilität ist die zeitliche Änderung der Abbildung parent.



- Einfaches Modell, nur für Spezialfälle geeignet:
  - Modellierung spezieller Eigenschaften von Komponenten-Orten (beispielsweise räumliche Koordinaten) unhandlich
  - keine Darstellung überlappender Orte möglich

k1

Mobilität – Beispiel für explizite Variante



Inhalt

- Motivation und Darstellungsweise
  - Motivation und Darstellung
  - Komponenten und Instanzen
- Systemmodell für Architektur: Instanzen
  - Grundkonzepte und Strukturen
  - Zeit, Kommunikation und Interaktion
  - Strukturänderungen
  - Mobilität
- Systemmodell für Architektur: Beschreibungen
  - Definition und Bedeutung
  - Reflexivität
- Zusammenfassung

Beschreibungen - Motivation

- Der Architekt arbeitet aber meist nicht mit einem laufenden System (es sei denn, er erstellt einen Prototyp), sondern mit Beschreibungen des Systems.
- Hier stellen sich insbesondere folgende Fragen:
  - Was sind Beschreibungen?
  - Was sagen Beschreibungen über die Instanzen aus?
  - Welche Beziehungen gibt es zwischen den Beschreibungen?

Beschreibungen – Definition und Beispiele

- Beispiele für Beschreibungen:
  - Spezifikationen (*.doc-Dateien)
  - UML-Diagramme (*.mdl-Dateien)
  - Schnittstellendefinitionen (*.idl-Dateien)
  - Quellcode (*.java-Dateien)
  - Binaries (*.class-Dateien)
- Zwischen Beschreibungen gibt es viele mögliche Beziehungen (import, derived, etc.)

Bedeutung von Beschreibungen

- Beschreibungen entsprechen Prädikaten; sie machen Vorgaben für Struktur und Verhalten der Instanzen.
- Beispiel: IDL-Datei Counter.idl

```
interface Counter {
   long getSum();
   void setSum(long x);
   long increment();
}
```

Counter.idl ist eine Beschreibung:
$\text{CounterIDL} \in \text{DESCRIPTION}$

Reflexivität

- Ein reflexives System ist ein System, bei dem Instanzen zur Laufzeit auf (bestimmte) Informationen aus den Beschreibungen der Komponenten zugreifen können.
- Typische Modellierung: Die Beschreibungen sind zur Laufzeit als Instanzen repräsentiert (z.B. Meta-Klassen).
  - Erfordert keine Erweiterung des Verhaltensbegriffs.
- Komplexe Modellierung: Instanzen greifen zur Laufzeit auf Beschreibungen zu.

Inhalt

- Motivation und Darstellungsweise
  - Motivation und Darstellung
  - Komponenten und Instanzen
- Systemmodell für Architektur: Instanzen
  - Grundkonzepte und Strukturen
  - Zeit, Kommunikation und Interaktion
  - Strukturänderungen
  - Mobilität
- Systemmodell für Architektur: Beschreibungen
  - Definition und Bedeutung
  - Reflexivität
- Zusammenfassung

Zusammenfassung

- Systemmodelle erlauben eine abstrakte Darstellung der wesentlichen Aspekte von Systemen.
- Komponenten, Schnittstellen und Verbindungen sind die Grundbausteine jeder Architektur.
- Der Architekt muss die jeweiligen Eigenschaften dieser Grundbausteine in seinem System kennen, um die Architektur einer Anwendung erarbeiten zu können.
- Für das Verhalten eines Systems sind sowohl Struktur als auch Kommunikation relevant.
- Für den Architekten sind sowohl Instanzen als auch Beschreibungen relevant.

Literaturhinweise

- [BR95] M. Broy: *Mathematical System Models as a Basis of Software Engineering*, J. van Leeuwen (ed.), Computer Science Today, Springer-Verlag 1995.
- [BHH+] R. Breu, U. Hinkel, C. Hofmann, C. Klein, B. Paech, B. Rumpe, V. Thurner: *Towards a Formalization of the Unified Modeling Language*, In: Proceedings of the ECOOP´97, Springer-Verlag, LNCS, 1997.
- [BRS+] K. Bergner, A. Rausch, M. Sihling, A. Vilbig, M. Broy: *A Formal Model for Componentware*, Gary T. Leavens, Murali Sitaraman (eds.), Foundations of Component-Based Systems, Cambridge University Press, 2000.
- [BR] M. Broy, B. Rumpe: Modulare hierarchische Modellierung als Grundlage der Software- und Systementwicklung. Informatik-Spektrum Band 30, Heft 1, Februar 2007, Springer-Verlag