

Softwarearchitektur verteilter Systeme

3. Grundlagen der Softwarearchitektur

Vorlesung Wintersemester 2002 / 03

Technische Universität München

Institut für Informatik

Lehrstuhl von Prof. Dr. Manfred Broy

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

Dr. Andreas Rausch, Dr. Marc Sihling

Inhalt



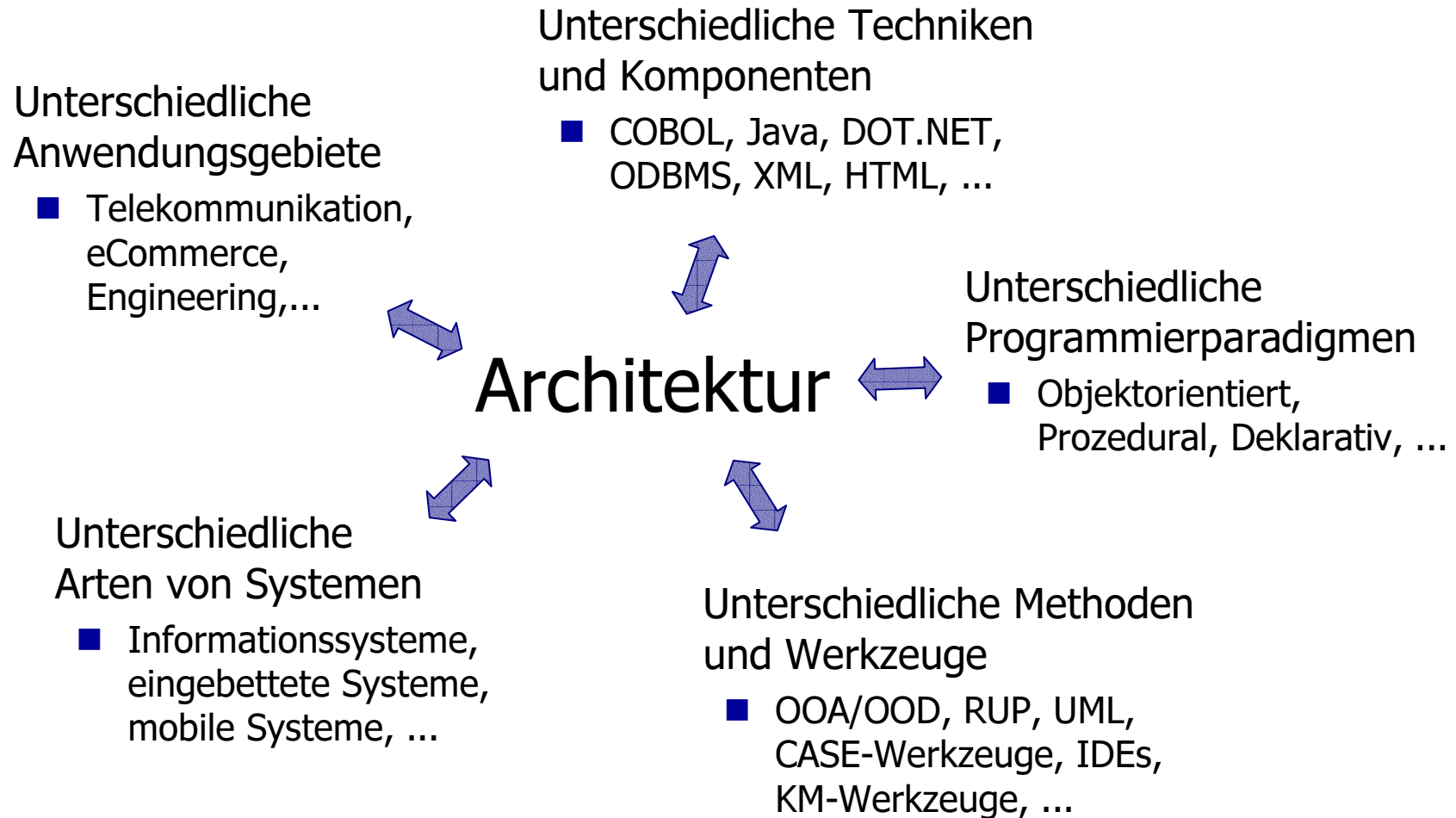
- Motivation und Darstellungsweise
- Systemmodell für Architektur: Instanzen
 - Grundkonzepte und Strukturen
 - Zeit, Strukturänderungen und Mobilität
 - Kommunikation und Interaktion
 - Verhalten von Systemen und Instanzen
- Systemmodell für Architektur: Beschreibungen
- Zusammenfassung

Inhalt



- Motivation und Darstellungsweise
- Systemmodell für Architektur: Instanzen
 - Grundkonzepte und Strukturen
 - Zeit, Strukturänderungen und Mobilität
 - Kommunikation und Interaktion
 - Verhalten von Systemen und Instanzen
- Systemmodell für Architektur: Beschreibungen
- Zusammenfassung

Einflussgebiete



Problematik bei der Darstellung

- SW-Architektur ist ein relativ neues Gebiet
 - Erst seit relativ kurzer Zeit als eigene Disziplin erkannt
 - Grundlagen noch wenig wissenschaftlich aufgearbeitet
- Begriffe sind noch stark im Fluss
 - Oft gibt es noch keine Abstraktionen
 - Konzepte stammen meist aus konkreten Systemen
 - Begriffe in unterschiedlichen Ansätzen unterschiedlich genutzt
 - Unterschiede werden betont (DOT.NET versus Java)
- Darstellung erfolgt meist anhand von Beispielen
 - Irrelevante technische Details erschweren Verständnis
 - Übertragung auf andere Systeme schwierig
 - Darstellung kostet viel Zeit

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?

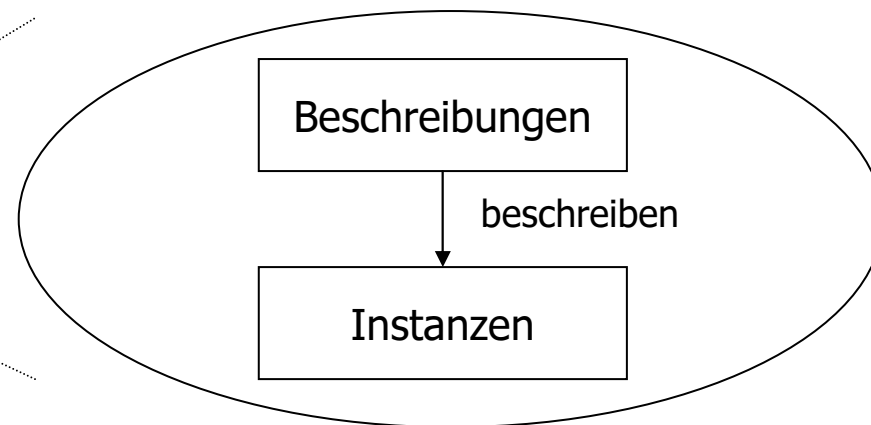
- Definiert Grundkonzepte und ihre Zusammenhänge
 - Konzentration auf relevante Strukturen und Konzepte
 - Gibt eine bestimmte Sichtweise auf ein System vor
- Abstrahiert von unwichtigen Details
 - Technische Details eines bestimmten Systems
 - Syntaktische Details einer bestimmten Beschreibung
- Verwendet mathematische Definitionen
 - Eindeutige und unmissverständliche Darstellung
 - Effiziente, sehr kompakte Darstellung
 - Einfach verständlich 😊
- Veranschaulichung durch Bilder und Diagramme

Instanzen und Beschreibungen

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



Mathematik
+ Bilder



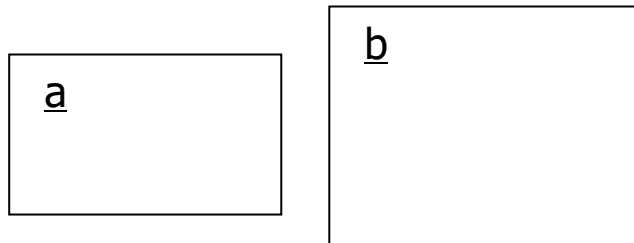
Inhalt



- Motivation und Darstellungsweise
- **Systemmodell für Architektur: Instanzen**
 - Grundkonzepte und Strukturen
 - Zeit, Strukturänderungen und Mobilität
 - Kommunikation und Interaktion
 - Verhalten von Systemen und Instanzen
- Systemmodell für Architektur: Beschreibungen
- Zusammenfassung

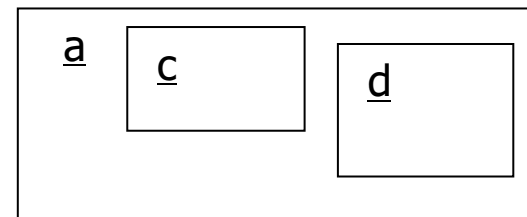
Komponenteninstanzen

- Architektur beschäftigt sich mit Systemen aus Bestandteilen.
- Die Bestandteile nennen wir **Komponenten**.
- Definitionen:
 - COMPONENT : unendliche Menge aller nur möglichen Komponenteninstanzen
- Beispiel:
 - $a, b \in \text{COMPONENT}$



Komponentenhierarchie

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



Verbindungen und Schnittstellen

- Komponenteninstanzen besitzen **Schnittstellen**instanzen.
- Schnittstelleninstanzen sind untereinander **verbunden**.
- Definitionen:

- INTERFACE : Menge aller Schnittstelleninstanzen

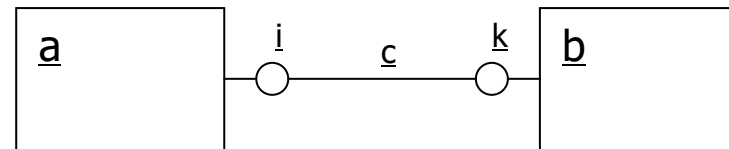


- assigned : INTERFACE \rightarrow COMPONENT

Totale Abbildung, ordnet jeder Schnittstelle die Komponente zu, der sie zugeordnet ist. Freie Schnittstellen sind nicht erlaubt.

- connects : CONNECTION \rightarrow INTERFACE \times INTERFACE

Totale Abbildung, gibt für jede Verbindung an, welche Schnittstellen sie verbindet. Eine Schnittstelle kann viele Verbindungen haben.

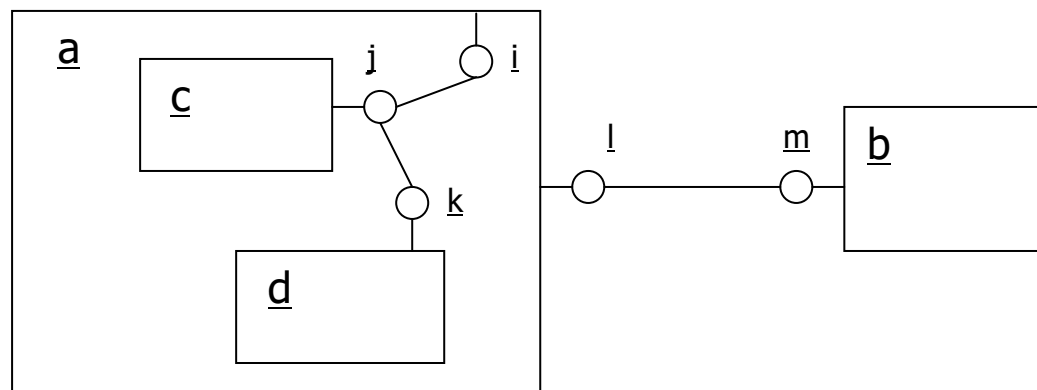


Systeme als Strukturen

- 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:
 - $INSTANCE = COMPONENT \cup INTERFACE \cup CONNECTION$
Menge aller möglichen Instanzen
 - $alive : INSTANCE \rightarrow Boolean$
Totale Abbildung; gibt an, ob eine Instanz für die Systemstruktur des betrachteten Systems relevant ist.
 - $STRUCTURE = alive \times assigned \times parent \times connects$
Menge aller möglichen Systemstrukturen.

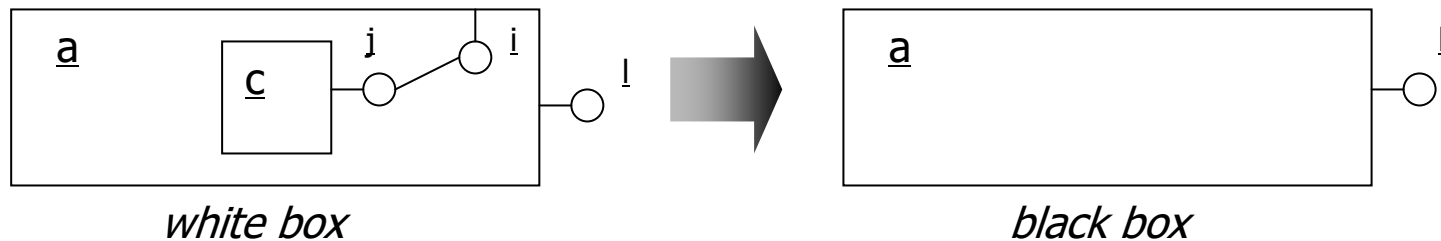
Systemstruktur - Beispiel

- $a, b, c, d \in \text{COMPONENT}; i, j, k, l, m \in \text{INTERFACE}$
- $v, w, x \in \text{CONNECTION}$
- $\forall \alpha \in \text{INSTANCE} : \text{alive}(\alpha) \Leftrightarrow \alpha \in \{a, b, c, d, i, j, k, l, m, v, w, x\}$
- $\text{parent}(c) = \text{parent}(d) = a$
- $\text{assigned}(i) = \text{assigned}(l) = a; \text{assigned}(m) = b$
- $\text{assigned}(j) = c; \text{assigned}(k) = d$
- $\text{connects}(v) = (i, j); \text{connects}(w) = (j, k); \text{connects}(x) = (l, m)$



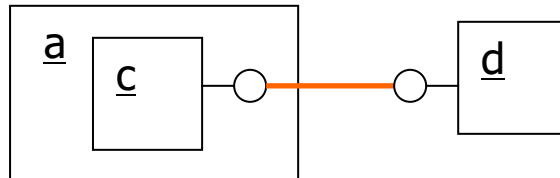
Schnittstellen und Implementierung

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



- Verbindungen dürfen dazu 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:** 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
- Systemmodell für Architektur: Instanzen
 - Grundkonzepte und Strukturen
 - **Zeit, Strukturänderungen und Mobilität**
 - Kommunikation und Interaktion
 - Verhalten von Systemen und Instanzen
- Systemmodell für Architektur: Beschreibungen
- Zusammenfassung

Zeit und Strukturänderungen



- Bis jetzt konnten wir nur statische Systemstrukturen beschreiben.
- 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.

Zeitströme

- Wir betrachten Zeitströme als Sequenzen von einzelnen Zeitschritten und definieren:
 - $T = \{t_0, t_1, t_2, t_3, \dots\}$: Menge aller betrachteten Zeitpunkte
 - $X^T = T \rightarrow X$: Zeitstrom der Elemente aus X
 - x_t : das Element aus X , das in X^T dem Zeitpunkt t zugeordnet ist

- Beispiel Gelage:

- Runde = {Bier, Wein, Nix}
- Gelage = Runde^T : Menge aller möglichen Gelage, enthält beispielsweise die Abläufe
- $g1 \in \text{Gelage}$ $g2 \in \text{Gelage}$

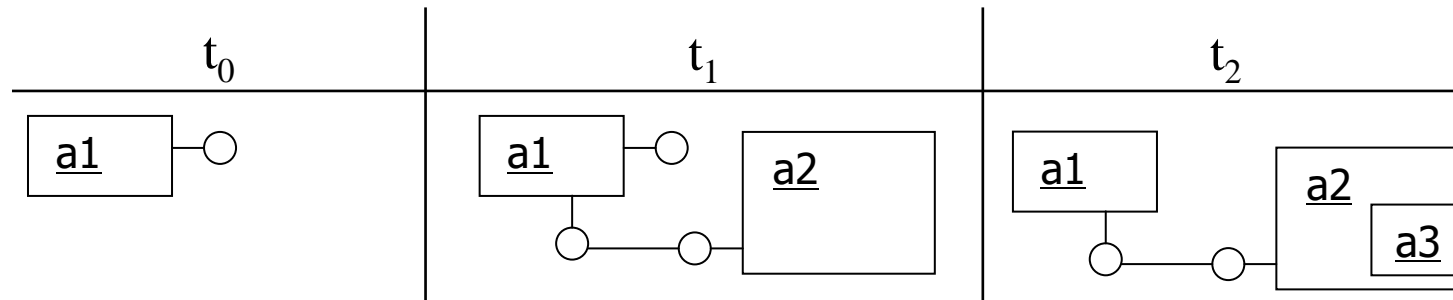
| | |
|-------|-----|
| t_0 | Nix |
| t_1 | Nix |
| t_2 | Nix |
| t_3 | Nix |
| ... | ... |

| | |
|-------|------|
| t_0 | Wein |
| t_1 | Bier |
| t_2 | Wein |
| t_3 | Bier |
| ... | ... |

- $g1_1 = \text{Nix}$ $g2_1 = \text{Bier}$

Modellierung der Strukturänderungen

- Definitionen:
 - $STRUCTURE^T = alive^T \times assigned^T \times parent^T \times connects^T$
- Ein Element aus $STRUCTURE^T$ beschreibt einen möglichen Systemablauf aus Struktursicht.
- Beispiel:



- Das Strukturverhalten eines Systems ist darstellbar als die Menge all seiner möglichen Systemabläufe.
 - $systembehavior \in \wp(STRUCTURE^T)$

Zeitmodell und Konsistenzbedingungen

- 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).
- Nicht sämtliche möglichen Abläufe sind erlaubt. Eine Vielzahl von Konsistenzbedingungen regelt, welche Abläufe zulässig sind:
 - Tote kehren nicht wieder:
 $\forall i \in \text{INSTANCE} : \neg \exists r, s, t \in T : r < s < t \wedge \text{alive}_r(i) \wedge \neg \text{alive}_s(i) \wedge \text{alive}_t(i)$
 - Interfaces wechseln ihre Komponente nicht:
 $\text{assigned}_t(i) = c \Rightarrow \text{assigned}_{t+1}(i) = c$
 - Interfaces leben nicht ohne ihre Komponente:
 $\text{assigned}(i) = c \Rightarrow \forall t \in T : \text{alive}_t(i) \Rightarrow \text{alive}_t(c)$

Zeitmodell und Konsistenzbedingungen

- Weitere grundlegende Konsistenzbedingungen für Systemabläufe:
 - Verbindungen verbinden immer die selben Schnittstellen.
 - Am Anfang der Zeit leben nur endlich viele Instanzen.
 - Von einem Zeitschritt zum nächsten gibt es nur endlich viele Änderungen an der Struktur.

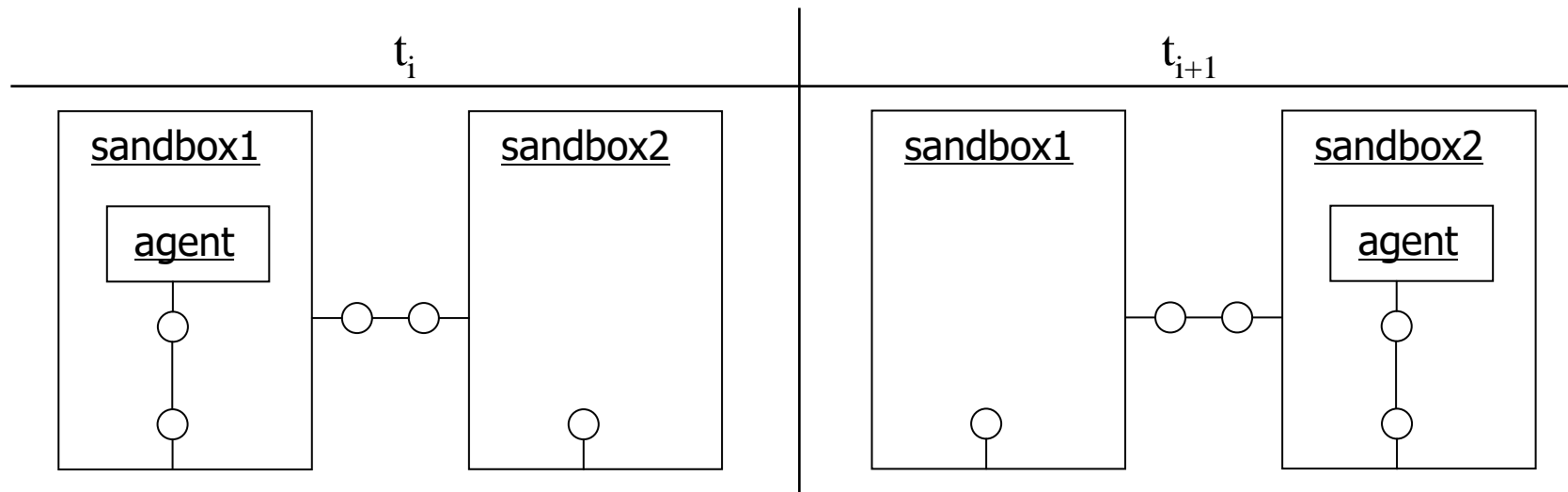
Mobilität



- 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

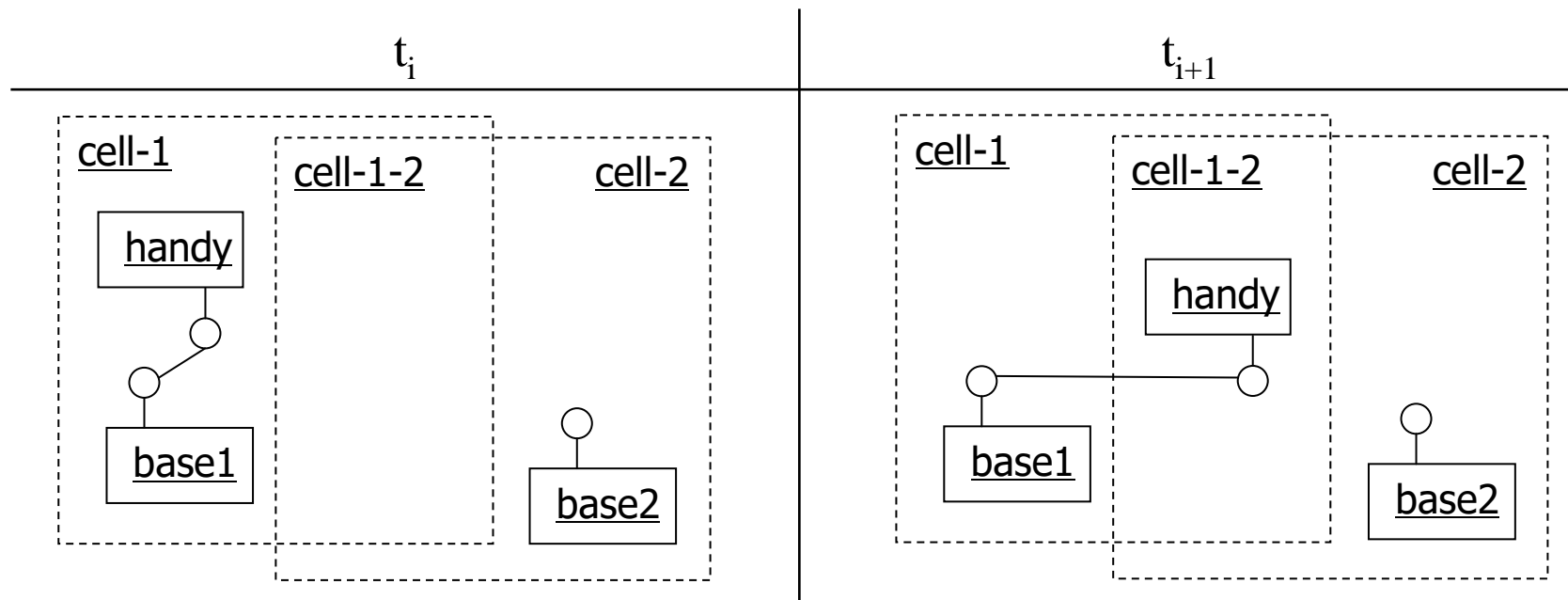
Mobilität – explizite Variante

- Es gibt ein spezielles Konzept für Orte.
- Grundlegende Definitionen:
 - LOCATION : Menge aller Orte
 - located : COMPONENT \rightarrow LOCATION
Totale Abbildung; ordnet jeder Komponente den Ort zu, an dem sie sich befindet.
 - sublocation : LOCATION \rightarrow \wp (LOCATION)
Totale Abbildung, definiert die Enthaltensein-Struktur der Orte.
- Modell ist komplexer, bietet aber höhere Mächtigkeit:
 - benötigt zusätzliche Konsistenzbedingungen z.B. für Azyklizität von sublocation, für Änderungen der Ortsstruktur, für Zusammenhang zwischen parent, sublocation, located etc.)
 - erlaubt natürliche Modellierung der Eigenschaften von Orten
 - erlaubt überlappende Orte – Komponente kann sich so an mehreren Orten gleichzeitig befinden

loc

Mobilität – Beispiel für explizite Variante

- Redefinition (auch located ist veränderlich): $STRUCTURE^T = alive^T \times assigned^T \times parent^T \times connects^T \times located^T$
- $cell-1, cell-2, cell-1-2 \in LOCATION$; $handy \in COMPONENT$
- $sublocation(cell-1-2) = \{cell-1, cell-2\}$
- $located_{t_i}(handy) = cell-1$; $located_{t_{i+1}}(handy) = cell-1-2$



Inhalt

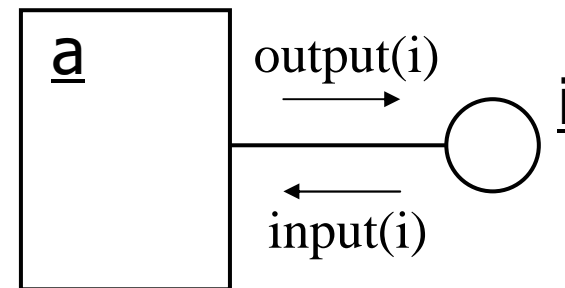


- Motivation und Darstellungsweise
- Systemmodell für Architektur: Instanzen
 - Grundkonzepte und Strukturen
 - Zeit, Strukturänderungen und Mobilität
 - **Kommunikation und Interaktion**
 - Verhalten von Systemen und Instanzen
- Systemmodell für Architektur: Beschreibungen
- Zusammenfassung

Kommunikation und Interaktion I

- Bis jetzt haben wir nur die Struktur von Systemen betrachtet ... aber Komponenten sollen auch miteinander kommunizieren.
- Nachrichten fließen von Schnittstellen zu Komponenten und umgekehrt.
- Definitionen:
 - MESSAGE : Menge aller möglichen Nachrichten
 - $\varepsilon \in \text{MESSAGE}$: leere Nachricht
 - $\text{input} : \text{INTERFACE} \rightarrow \text{MESSAGE}$
 - $\text{output} : \text{INTERFACE} \rightarrow \text{MESSAGE}$

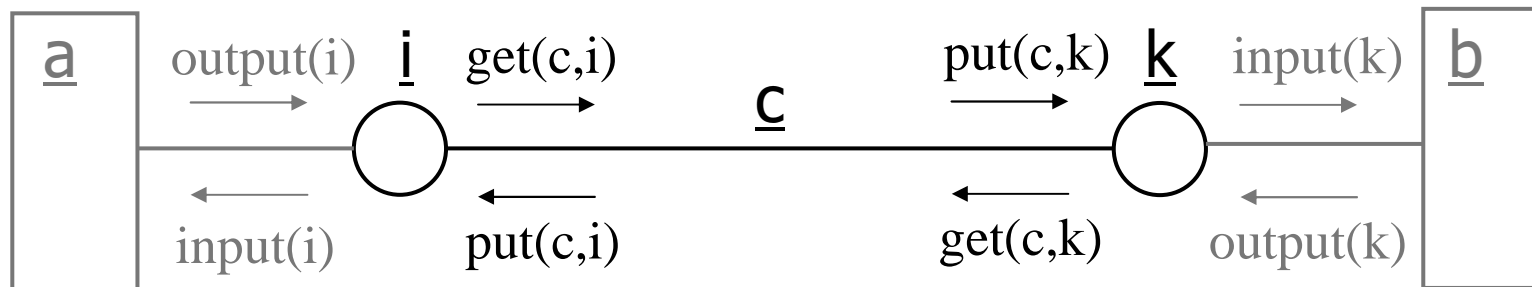
Partielle Abbildungen; geben an, welche Nachrichten in eine bzw. aus einer Komponente fließen. Definition von Zeitströmen input^T und output^T wie üblich.



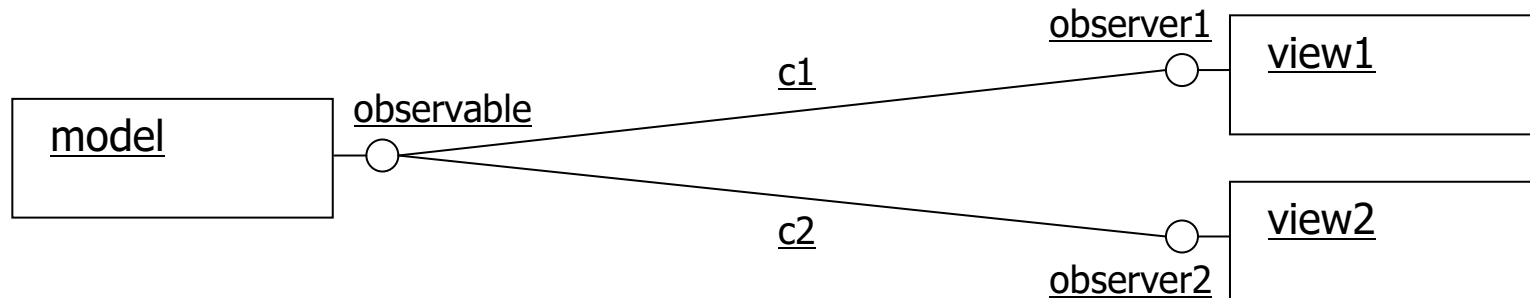
Kommunikation und Interaktion II

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

Partielle Abbildungen; geben an, welche Nachrichten in eine bzw. aus einer Verbindung fließen. Definition von Zeitströmen get^T und put^T wie üblich.



Kommunikation und Interaktion – Beispiel Observer

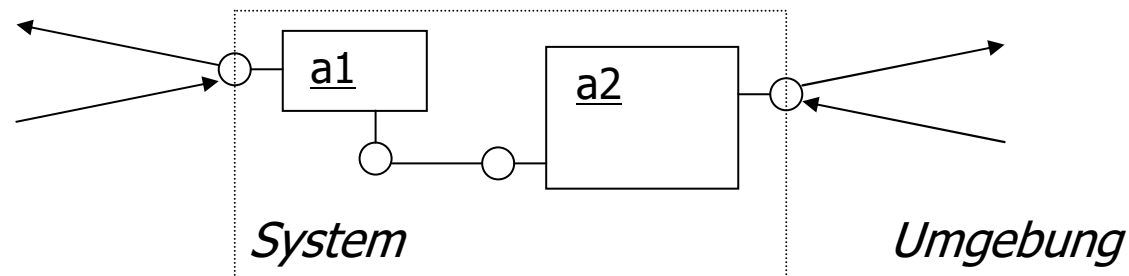


■ Beispielablauf:

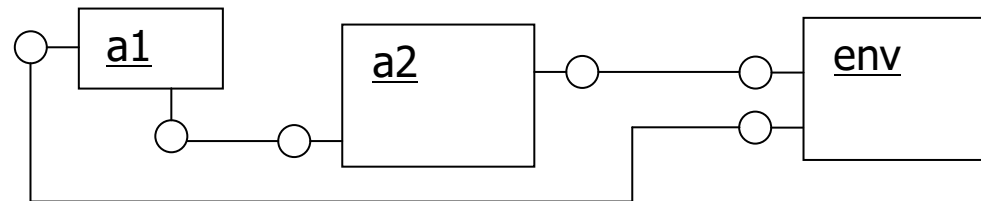
- t_1 : $\text{output}_{t_1}(\text{observer1}) = \text{„changeTo(7)“}$
- t_2 : $\text{get}_{t_2}(c1, \text{observer1}) = \text{„changeTo(7)“}$
- t_3 : $\text{put}_{t_3}(c1, \text{observable}) = \text{„changeTo(7)“}$
- t_4 : $\text{input}_{t_4}(\text{observable}) = \text{„changeTo(7)“}$
- t_5 : $\text{output}_{t_5}(\text{observable}) = \text{„notify(danger)“}$
- t_6 : $\text{get}_{t_6}(c1, \text{observable}) = \text{get}_{t_6}(c2, \text{observable}) = \text{„notify(danger)“}$
- t_7 : $\text{put}_{t_7}(c1, \text{observer1}) = \text{„notify(danger)“}$
- t_8 : $\text{put}_{t_8}(c2, \text{observer2}) = \text{„notify(danger)“}$

Kommunikation mit der Außenwelt

- In einem System kommunizieren typischerweise einige Komponenten mit der Umgebung.
- Darstellungsmöglichkeiten:
 - Umgebung kommuniziert über freie Schnittstellen:



- Außenwelt wird als eigene Komponente repräsentiert:



Exkurs: Zustand

- In dem Systemmodell gibt es zwei Arten von Zustand.
- Struktur-Zustand des Systems
 - Falls die Struktur dynamisch ist, kann das System sie zur Speicherung von Informationen nutzen.
 - Beispiel: Referenzen zwischen Kunden- und Konten-Objekten
- Daten-Zustand von Instanzen
 - Instanzen können Zustand halten.
 - Beispiel: Speicherung des Kontostands in einem Konto-Objekt.
 - Den Daten-Zustand modellieren wir nur implizit über das Kommunikationsverhalten: Die Ausgabe einer Instanz zu einem Zeitpunkt t kann grundsätzlich von allen früheren Eingaben abhängen.
 - Instanzen, deren Ausgaben jeweils nur von der letzten nichtleeren Eingabe abhängen, nennen wir zustandslos.

Inhalt

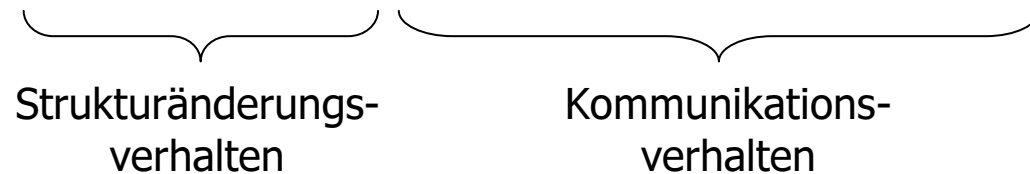


- Motivation und Darstellungsweise
- Systemmodell für Architektur: Instanzen
 - Grundkonzepte und Strukturen
 - Zeit, Strukturänderungen und Mobilität
 - Kommunikation und Interaktion
 - Verhalten von Systemen und Instanzen
- Systemmodell für Architektur: Beschreibungen
- Zusammenfassung

Systemverhalten

- Auf der Basis von Struktur und Kommunikation lässt sich das Verhalten eines Systems nun wie folgt darstellen:

- $\text{systembehavior} \in \wp(\text{STRUCTURE}^T \times \text{get}^T \times \text{put}^T \times \text{input}^T \times \text{output}^T)$



- Das Gesamtverhalten eines Systems ist also die Menge all seiner möglichen Systemabläufe im Hinblick auf die Struktur der Instanzen und die Kommunikation zwischen ihnen.
- These: Mit Hilfe der bis jetzt dargestellten Konzepte lassen sich alle architektur-relevanten Aspekte eines Systems erfassen.

Vom Systemverhalten zum Instanzenverhalten

- Bis jetzt haben wir immer nur das Verhalten des gesamten Systems beobachtet:
 - Wie sieht die Gesamtstruktur zu einer bestimmten Zeit aus?
 - Welche Veränderungen im Gesamtsystem haben sich gegenüber dem vorigen Zeitpunkt ergeben?
 - Welche Nachrichten werden im Gesamtsystem verschickt bzw. empfangen?
- In der Folge schauen wir uns das Verhalten einzelner Instanzen an:
 - Welche Ausschnitte aus dem Systemverhalten sind relevant für das Verhalten einzelner Instanzen?
 - Was können einzelne Instanzen im Gesamtsystem bewirken?
 - Welches Verhalten ergibt sich, wenn man Instanzen zu einem System zusammensetzt?

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).
 - $(\text{gets}, \text{puts}) \in \text{behavior}(c) \wedge \text{connected}(c) = (i,k) \Rightarrow$
 $\forall t \in T: \text{gets}_t \in \{ \{(c,i),(c,k)\} \rightarrow M \} \wedge \text{puts}_t \in \{ \{(c,i),(c,k)\} \rightarrow M \}$

s für Zeitstrom

Verhalten von Verbindungsinstanzen II

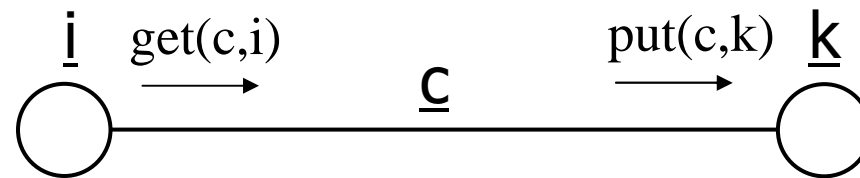
- Kausalität: Die Ausgaben einer Verbindung c zu einem bestimmten Zeitpunkt t können nur von den Eingaben bis zum Zeitpunkt $t-1$ abhängen (und insbesondere nicht von zukünftigen Eingaben).
 - $(\text{gets1}, \text{puts1}) \in \text{behavior}(c) \wedge (\text{gets2}, \text{puts2}) \in \text{behavior}(c)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

Notation: $x \downarrow t$ bezeichnet den Teil des Zeitstroms x bis einschließlich Zeitpunkt t

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.

Verhalten von Verbindungsinstanzen IV - Beispiele



- A) Sämtliche Nachrichten werden im nächsten Zeitschritt am anderen Ende ausgeliefert.
 - $get_t(c,i) = m \Rightarrow put_{t+1}(c,k) = m$
 - z.B. Objekte als Komponenten, Verweise als Verbindungen
- B) Nachrichten werden gepuffert und müssen vom Empfänger explizit angefordert werden (durch Senden einer speziellen Anforderungsnachricht).
 - über Definition einer Queue, Definition spezieller Nachrichten in der Menge M , Definition des Protokolls
 - z.B. Objekte als Komponenten, JMS-Kanäle als Verbindungen

Verhalten von Schnittstelleninstanzen I

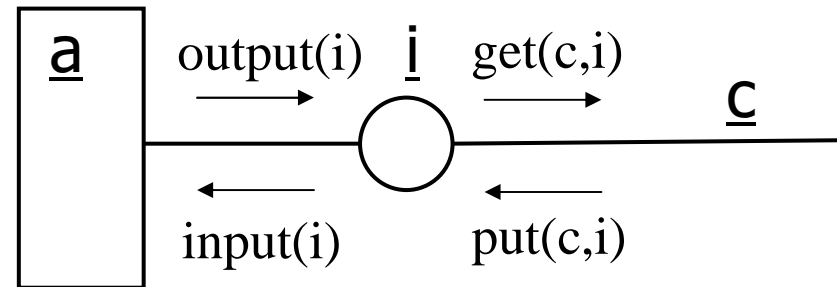
- Das Verhalten einer Schnittstelleninstanz lässt sich ähnlich wie das Verhalten einer Verbindung definieren und bezieht sich auf dieses.
- Definitionen:
 - $\text{behavior} : \text{INTERFACE} \rightarrow \wp(\text{get}^T \times \text{put}^T \times \text{input}^T \times \text{output}^T)$
- Lokalität: Das Verhalten einer Schnittstelle i macht nur Aussagen über Nachrichten an der Schnittstelle und an den Enden der Verbindungen, die mit der Schnittstelle verbunden sind.
 - $(\text{gets}, \text{puts}, \text{inputs}, \text{outputs}) \in \text{behavior}(i) \Rightarrow \forall t \in T:$
 $\text{gets}_t, \text{puts}_t \in \{ \{(c,i)\} \rightarrow M \mid c \in \text{connections}(i) \} \wedge$
 $\text{inputs}_t, \text{outputs}_t \in \{ \{i\} \rightarrow M \}$

Menge der Verbindungen an i

Verhalten von Schnittstelleninstanzen II

- Kausalität wird ähnlich definiert wie bei Verbindungen.
- Wie bei Verbindungen gibt es viele Möglichkeiten, wie sich eine Schnittstelle verhalten kann, insbesondere:
 - Sie kann Nachrichten von mehreren Verbindungen mischen (nach einer bestimmten Strategie, beispielsweise fair).
 - Sie kann Nachrichten an Verbindungen schicken (an eine bestimmte Verbindung oder an alle).
- Das Verhalten einer Schnittstelleninstanz muss mit dem Verhalten der angrenzenden Verbindungsinstanzen zusammenpassen.
- Der Architekt muss die Eigenschaften von Schnittstellen in seinem Laufzeitsystem kennen, um die Architektur einer Anwendung erarbeiten zu können.

Verhalten von Schnittstelleninstanzen III – Beispiel



- „Vielzweck-Schnittstelle“:

- hinausgehende Nachrichten gehen an alle Verbindungen (Schnittstelle als Verteiler)

$c \in \text{connections}(i) \wedge \text{alive}_t(i) \wedge \text{alive}_{t+1}(c) \Rightarrow \text{get}_{t+1}(c,i) = \text{output}_t(i)$
z.B. verwendbar, um JMS-Topics zu modellieren

- hineingehende Nachrichten kommen aus einer Verbindung (Schnittstelle als Mischer):

$\text{alive}_t(i) \wedge \text{input}_t(i) = m \Rightarrow$

$\exists c \in \text{connections}(i) : \text{alive}_{t-1}(c) \wedge \text{put}_{t-1}(c,i) = m$

Verhalten von Komponenten

- Das Verhalten einer Komponente hat zwei Aspekte:
 1. Die Komponente kann Nachrichten empfangen und versenden.
 2. Die Komponente kann die Struktur des Systems verändern.
- Definitionen:
 - $\text{behavior} : \text{COMPONENT} \rightarrow \wp(\underbrace{\text{input}^T \times \text{output}^T}_{\text{Kommunikationsverhalten}} \times \underbrace{\text{STRUCTURE}^T}_{\text{Strukturänderungsverhalten}})$
- Sowohl das Kommunikationsverhalten als auch das Strukturänderungsverhalten sind lokal und kausal.

Kommunikationsverhalten von Komponenten

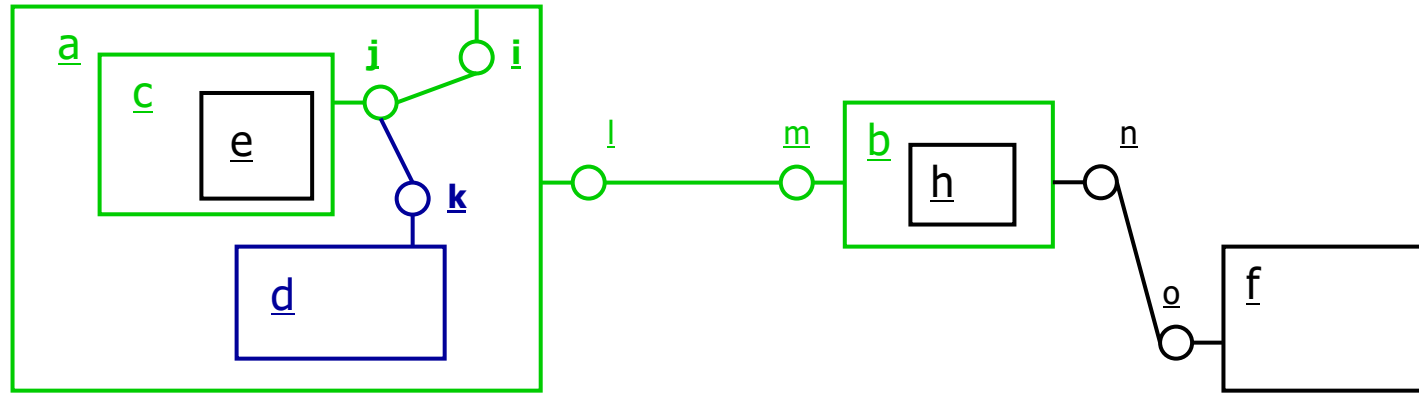
- **Lokalität:** Das Kommunikationsverhalten einer Komponente c macht nur Aussagen über input- und output-Nachrichten von Schnittstellen von c .
 - analog zu Lokalität bei Verbindungen und Schnittstellen
- **Kausalität:** Die Ausgaben einer Komponente c zu einem bestimmten Zeitpunkt t können nur von den Eingaben und dem Strukturverhalten von c bis zum Zeitpunkt $t-1$ abhängen.
 - $(inputs1, outputs1, structures1) \in behavior(c) \wedge$
 $(inputs2, outputs2, structures2) \in behavior(c) \wedge$
 $inputs1 \downarrow_{t-1} = inputs2 \downarrow_{t-1} \wedge$
 $structures1 \downarrow_{t-1} = structures2 \downarrow_{t-1} \Rightarrow$
 $outputs1_t = outputs2_t$

deterministischer
Fall, vgl. Folie 2.37

Strukturänderungsverhalten von Komponenten I

- **Lokalität:** Das Strukturänderungsverhalten einer Komponente *c* macht nur Aussagen über Instanzen, auf die die Komponente zugreifen kann.
- Hier gibt es unterschiedliche Möglichkeiten. Zugreifbar sind im Normalfall mindestens:
 1. die Komponente *c* selbst
 2. die Schnittstellen der Komponente *c*
 3. (bestimmte) Verbindungen an Schnittstellen von *c*
 4. (bestimmte) Schnittstellen und Komponenten, die unmittelbar über hinausgehende Verbindungen mit *c* verbunden sind
- Zusätzlich können zugreifbar sein:
 5. die Kind-Komponenten von *c* und Verbindungen zwischen ihnen

Strukturänderungsverhalten von Komponenten II



- Die zugreifbaren Instanzen zu einem Zeitpunkt t können jeweils durch eine Formel beschrieben werden:
 - $\text{reachable}_t(c) =$
 1. $\{c\} \cup$
 2. $\{i \mid \text{assigned}_t(i) = c\} \cup$
 3. $\{\text{con} \mid \text{connected}_t(\text{con}) = (i,j) \wedge \text{assigned}_t(i) = c\} \cup$

...

Strukturänderungsverhalten von Komponenten III

- Lokalität als Formel: Das Strukturänderungsverhalten einer Komponente c macht nur Aussagen über Instanzen, auf die die Komponente zugreifen kann.
 - $(inputs, outputs, structures) \in behavior(c) \wedge$
 $structures = (alives, assigned, parent, connects) \Rightarrow \forall t \in T:$
 - $alives_t \in \{$
 $\{ x \mid x \in COMPONENT \cap reachable_t(c) \} \rightarrow M \} \wedge$
 - $assigned_t \in \{$
 $\{ x \mid x \in INTERFACE \cap reachable_t(c) \} \rightarrow COMPONENT \} \wedge$
 - $parent_t \in \{$
 $\{ x \mid x \in COMPONENT \cap reachable_t(c) \} \rightarrow COMPONENT \} \wedge$
 - $connects_t \in \{$
 $\{ x \mid x \in CONNECTION \cap reachable_t(c) \} \rightarrow$
 $INTERFACE \times INTERFACE \}$

Komposition von Instanzen

- Instanzen lassen sich in ein System einbauen, wenn ihr Verhalten kompatibel mit dem Systemverhalten ist.
- Beispiel: Verbindungsverhalten
 - Wenn bei einer Verbindung eines Systems eine bestimmte Eingabesequenz `gets` möglich ist, müssen sämtliche dazugehörigen Ausgabesequenzen `puts` aus dem Komponentenverhalten im System möglich sein.
 - $(\text{gets}, \text{puts}) \in \text{behavior}(\text{conn}) \wedge (\text{gets}, \text{puts}) \leq \text{systembehavior} \Rightarrow \forall x \in \{ (g, p) \in \text{behavior}(\text{conn}) \mid g = \text{gets} \} : x \leq \text{systembehavior}$

Verhalten des Komponente findet sich im Systemverhalten wieder – Definition ähnlich wie Definition der Lokalität

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.

Beispiele



- Komponenten sind große Anwendungen, die sich über http-Verbindungen XML-Dokumente schicken.
 - Übermittlung von Nachrichten nur mit (ungewisser) Verzögerung
 - Verbindungen können ausfallen
 - Nachrichten werden bei der Übertragung nicht verfälscht
 - Komponentenstruktur ändert sich nicht (oder nur selten)
 - ...
- Komponenten sind Objekte innerhalb einer Anwendung, die über Referenzen miteinander verbunden sind und Methoden aufeinander aufrufen.
 - Übermittlung von Nachrichten im Wesentlichen verzögerungslos
 - Verbindungen können nicht ausfallen
 - Nachrichten gehen nicht verloren
 - Nachrichten sind Aufrufe (Request/Reply)
 - Komponentenstruktur ändert sich häufig
 - ...

Inhalt



- Motivation und Darstellungsweise
- Systemmodell für Architektur: Instanzen
 - Grundkonzepte und Strukturen
 - Zeit, Strukturänderungen und Mobilität
 - Kommunikation und Interaktion
 - Verhalten von Systemen und Instanzen
- **Systemmodell für Architektur: Beschreibungen**
- Zusammenfassung

Beschreibungen - Motivation

- Bis jetzt haben wir nur Instanzen betrachtet.
- Die Sicht war im Wesentlichen die eines Tracers:
 - Was passiert alles im System?
 - Was tun die einzelnen Komponenten?
 - Wie arbeiten sie zusammen?
- 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

- Definitionen:
 - DESCRIPTION : Menge aller möglichen Beschreibungen
 - describedBy : DESCRIPTION \rightarrow \wp (INSTANCE)
Jede Beschreibung kann viele Instanzen beschreiben.
- 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 I

- 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:
 $\text{CounterIDL} \in \text{DESCRIPTION}$

- Standardumsetzung in Prädikate:
 - Die Beschreibung beschreibt nur (bestimmte) Schnittstellen:
 $i \in \text{describedBy}(\text{CounterIDL}) \Rightarrow i \in \text{INTERFACE}$

Bedeutung von Beschreibungen II

- Über Schnittstellen, die der Beschreibung gehorchen, laufen nur bestimmte Nachrichten:
 $i \in \text{describedBy}(\text{CounterIDL}) \wedge$
 $(\text{gets}, \text{puts}, \text{inputs}, \text{outputs}) \in \text{behavior}(i) \Rightarrow \forall t \in T:$
 $\text{outputs}_t(c,i) \in \{ \text{getSum}, \text{setSum}(n), \text{increment} \mid n \in \text{Nat} \} \wedge$
 $\text{inputs}_t(c,i) \in \text{Nat}$
- Weiterhin kann noch die Semantik des entfernten Methodenaufrufs vorgegeben werden: Nach Senden von `getSum` muss zunächst eine Antwort erfolgen, bevor eine weitere Ausgabe möglich ist.
- Weitere Möglichkeiten: Exception-Handling spezifizieren, Anreicherung mit Vor- und Nachbedingungen, etc. etc.

Reflexivität

- Ein reflexives System ist ein System, bei dem Instanzen zur Laufzeit auf (bestimmte) Informationen aus den Beschreibungen 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.
 - Erfordert Erweiterung des Verhaltensbegriffs
systembehavior \in
 $\wp(\text{STRUCTURE}^T \times \text{get}^T \times \text{put}^T \times \text{input}^T \times \text{output}^T \times \text{descriptions}^T)$

Inhalt



- Motivation und Darstellungsweise
- Systemmodell für Architektur: Instanzen
 - Grundkonzepte und Strukturen
 - Zeit, Strukturänderungen und Mobilität
 - Kommunikation und Interaktion
 - Verhalten von Systemen und Instanzen
- Systemmodell für Architektur: Beschreibungen
- **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.