Ablaufmodelle: Prozesse und Interaktionsdiagramme

Modellbildung in der Entwicklung

Prof. Dr. Dr. h.c. Manfred Broy
Gemeinsam mit Dr. Bernhard Schätz
Fakultät für Informatik, TU München

SS 2007

Dualität: Zustands- vs. Ablaufmodelle

- Zustandsmodelle von Systemen:
  Das Verhalten von Systemen wird durch Zustandsmaschinen und Zustandsübergänge beschrieben
- Ablaufmodelle von Systemen:
  Das Verhalten von Systemen wird durch Mengen von Abläufen beschrieben
  Zentrale Fragen:
  - Was ist ein Ablauf/ein Prozess
  - Was sind sequentielle und parallele Abläufe

Systemabläufe

Wir betrachten Zustandsmaschinen mit Ein/Ausgabe:

Sei

State ein Zustandsraum

IA eine Menge von Eingabe(aktion)en

OA eine Menge von Ausgabe(aktion)en

Δ eine Zustandsübergangsfunktion

$State_0$ eine Menge von Anfangszuständen

$$\Delta: (State \times IA) \rightarrow \wp(State \times OA)$$

Ablauf

Für jeden Anfangszustand $s_0 \in State_0$ und jede Folge von Eingaben

$$e_1\ e_2\ e_3\ e_4\ e_5\ e_6\ \ldots$$

erhalten wir eine Folge von Ausgaben

$$a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ \ldots$$

eine Folge von Zuständen

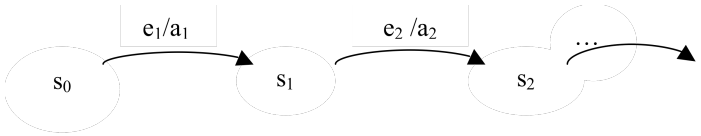
$$s_1\ s_2\ s_3\ s_4\ s_5\ s_6\ \ldots$$

durch

$$(s_{i+1}, a_{i+1}) \in \Delta(s_i, e_{i+1})$$

Ablauf

Ein Ablauf hat dann die folgende Form:



Abläufe sind somit sequentiell

Prozesse

Gegeben Aktionsmenge A: Ein Prozess besteht aus

E einer Menge von Ereignissen

≤ einer partiellen Ordnung auf E

α einer Zuordnung von Aktionen auf Ereignisse

$$\alpha : E \rightarrow A$$

Gilt für Ereignisse e1, e2 $\in$ E:

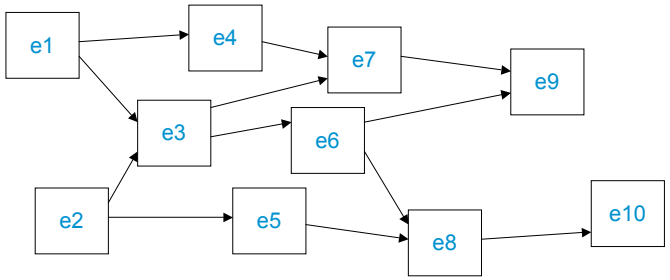
e1 ≤ e2 oder e2 ≤ e1

dann heißen e1 und e2 zueinander sequentiell

andernfalls parallel (nebenläufig)

Graphische Darstellung von Prozessen

Ein endlicher Prozess kann durch einen azyklischen Graphen dargestellt werden:

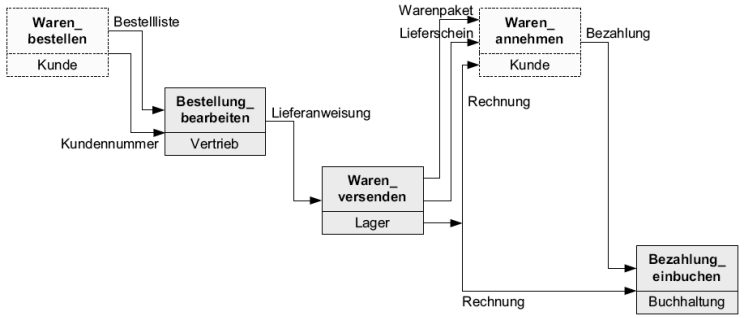


Prozesse von Zustandsmaschinen

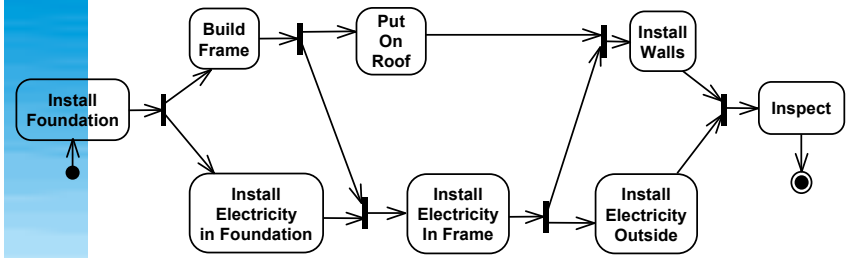
- Jeder Zustandsübergang entspricht einem Ereignis
- Es entsteht ein sequentieller Prozess von Ereignissen
- Die Aktionen sind die Ein-/Ausgabe

Geschäftsprozesse

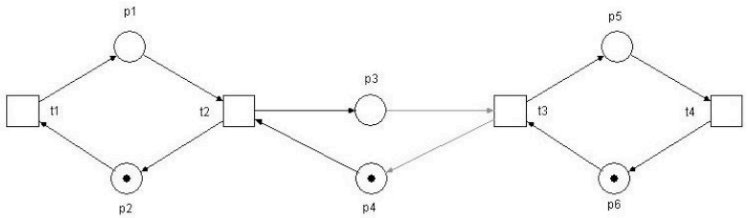
Nach Diss. Thurner



Activity Diagrams in UML



Zusammenhang zu Petri-Netzen



Zustandsmaschinen und Nebenläufigkeit

- Zustandsmaschinen haben zunächst sequentielle Abläufe
- Führt man mehrere Kommunikationskanäle ein, so kann eine Zustandsmaschine parallel Nachrichten empfangen und senden
- Komponiert man Zustandsmaschinen parallel so erhält man parallele Abläufe

Zustandsmaschinen mit Ein/Ausgabekanälen

- Seien I und O Mengen von Kanälen
  - Ein Kanal ist eine Kommunikationsverbindung
- Sei M Menge von Nachrichten
- Zustandsmaschine mit Ein- und Ausgabe

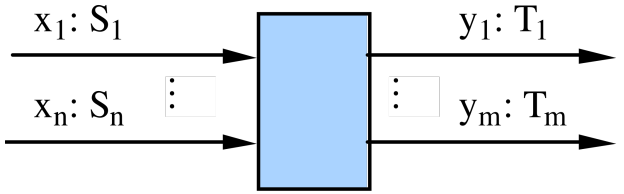
$$\Delta: (STATE \times (I \rightarrow M^*)) \rightarrow \wp(STATE \times (O \rightarrow M^*))$$

mit Anfangszustand $\sigma_0 \in STATE$ gegeben.

- Wir betrachten im Folgenden Moore Maschinen: Dabei hängt die Ausgabe y in (s', y) $\in$ Δ(s, x) nur vom Zustand s nicht aber von der Eingabe x ab.

Zustandsmaschinen mit Ein/Ausgabekanälen

- Seien I = {x_1, x_2, ..., x_n} und O = {y_1, y_2, ..., y_n}
- Die Kanäle können zusätzlich typisiert sein



Zustandsmaschinen mit Ein/Ausgabekanälen

- Seien I_j und O_j Mengen von Kanälen
- Sei M Menge von Nachrichten
- Seien Zustandsmaschinen

$$\Delta_j: (STATE_j \times (I_j \rightarrow M^*)) \rightarrow \wp(STATE_j \times (O_j \rightarrow M^*))$$

mit Anfangszuständen $\sigma 0_j \in STATE_j$ gegeben. Durch die Komposition der Übergangsfunktionen Δ_1 und Δ_2 erhalten wir die Übergangsfunktion:

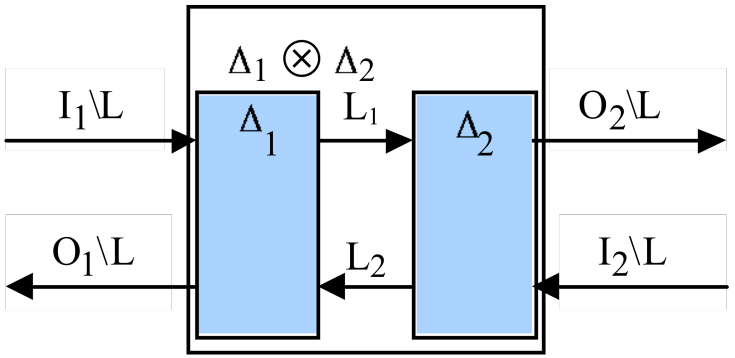
$$\Delta: (STATE \times (I \rightarrow M^*)) \rightarrow \wp(STATE \times (I_j \rightarrow M^*))$$

Definiert durch $\Delta = \Delta_1 \otimes \Delta_2$.

Komposition

$L_1 = I_2 \cap O_1$, $L_2 = I_1 \cap O_2$, $L = L_1 \cup L_2$

$I = (I_1 \cup I_2) \backslash L$, $O = (O_1 \cup O_2) \backslash L$



Komposition

- Als Zustandsmenge von $\Delta = \Delta_1 \otimes \Delta_2$ erhalten wir das Kreuzprodukt der Zustände

  $STATE = STATE_1 \times STATE_2$

  mit Anfangszustand $(\sigma 0_1, \sigma 0_2)$, und

  $\Delta((\sigma_1, \sigma_2), x) = \{ ((\sigma'_1, \sigma'_2), z|O) :$

  $\exists\, z \in (I \cup O \cup L \rightarrow M^*)) :$

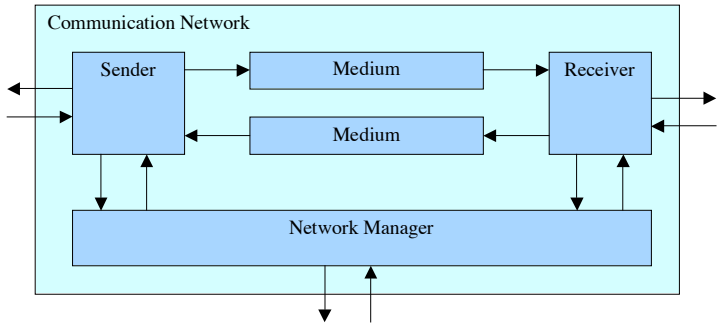
  $z|I = x \wedge (\sigma'_j, z|O_j) \in \Delta_j(\sigma_j, z|I_j)$ für $j = 1, 2 \}$
- Wir definieren für die Maschinen $Z_i = (\Delta_i, \sigma 0_i)$ für $i = 1, 2$ die Komposition $Z_1 \otimes Z_2$ durch die Gleichung

  $Z_1 \otimes Z_2 = (\Delta_1 \otimes \Delta_2, (\sigma 0_1, \sigma 0_2))$

Konsistenz der Komposition

- Da wir auf unendliche Zustandsräume
  - verallgemeinerte Moore-Maschinen betrachten und deshalb die Ausgabe nur von dem Zustand abhängt,
  - ist die Definition auch im Falle von Rückkopplung konsistent und insbesondere die Komposition totaler Zustandsmaschinen wieder total.
- Hier zahlt sich die Festlegung auf Moore-Maschinen aus.

Architekturen aus Zustandsmaschinen



Architekturen von Zustandsmaschinen

- In jedem Schritt
- empfängt jede Maschine eine Sequenz von Nachrichten über jeden ihrer Kanäle

und

- versendet jede Maschine eine Sequenz von Nachrichten über jeden ihrer Kannäle

Die Nachrichten können auch Methodenaufrufen oder Rückkehrnachrichten für Methoden entsprechen.

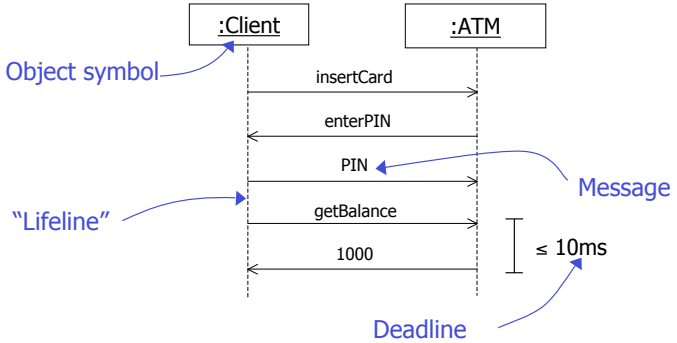
Interaction Diagrams

- Interaction is key ingredient of distributed system's SW architecture:
  - synchronous method-/operation calls
  - asynchronous communication via message exchange
- Coordination/Synchronization via interaction to establish use case
- Communication infrastructure:
  - Object/Component identifiers and interfaces
  - Communication paths (⇒ domain model)
  - Method call / message exchange mechanisms
- Communication defines:
  - Assumptions at environment
  - Commitment towards environment

Interaction Diagrams

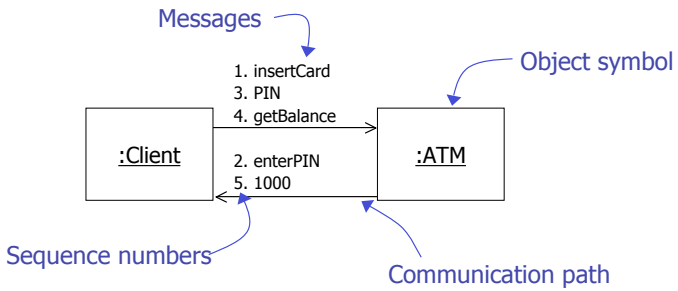
UML: Sequence and collaboration diagrams

Sequence Diagrams (SDs):



Interaction Diagrams

Collaboration diagrams:

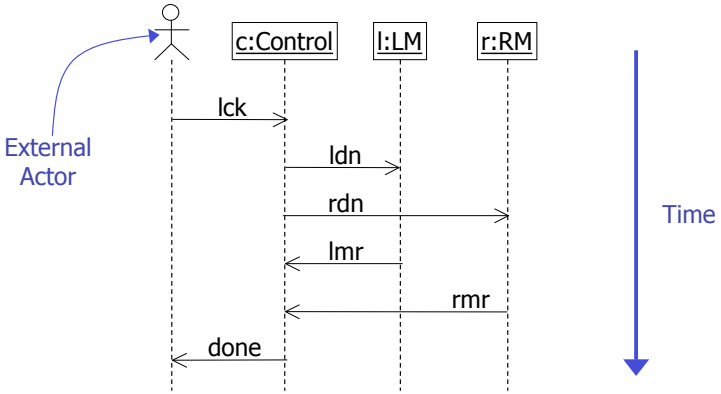


Sequence Diagrams

Roots of SDs:

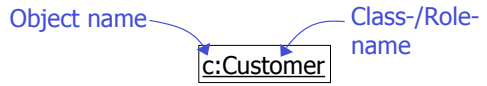
- MSC-96 (Message Sequence Charts, ITU-T Standard Z.120)
  - ⇒ Specification and documentation of asynchronous telecommunications protocols
- OMSCs (Object Message Sequence Charts, [POSA96])
  - ⇒ Specification and documentation of control-flow-oriented systems

Sequence Diagrams: Basic Notation

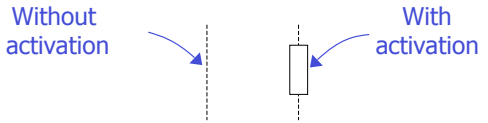


Sequence Diagrams: Basic Notation

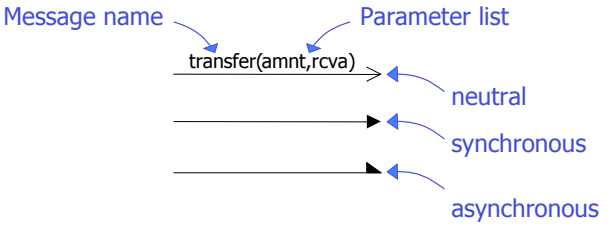
- object symbol:



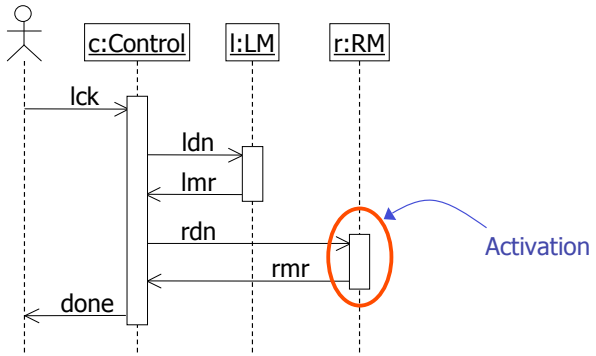
- lifeline:



- arrows:

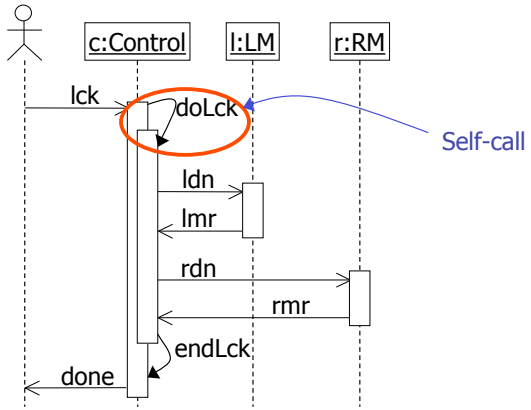


Sequence Diagrams: Activations



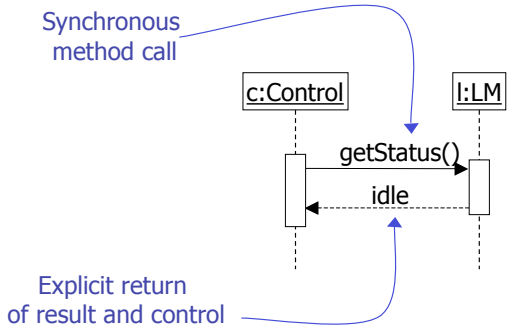
Sequence Diagrams: Activations

Internal Control Flow



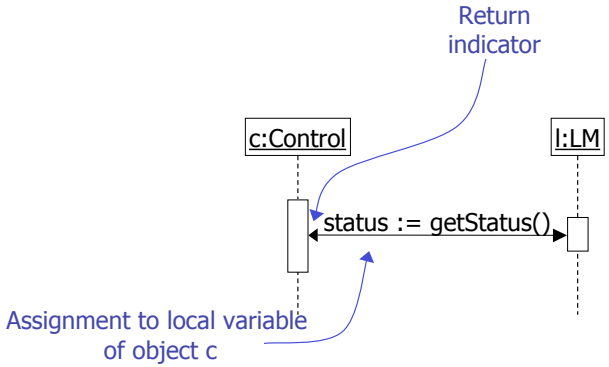
Sequence Diagrams: Returns

Explicit return arrows indicate return of result and control to caller



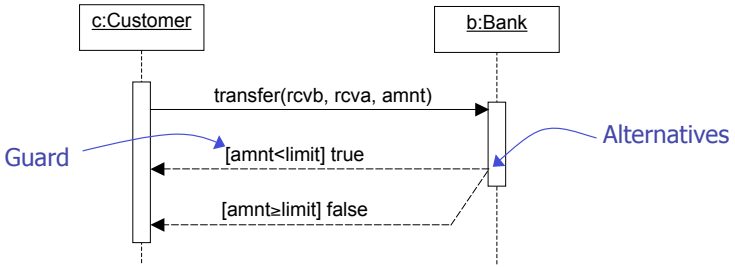
Sequence Diagrams: Returns

Implicit return of result and control



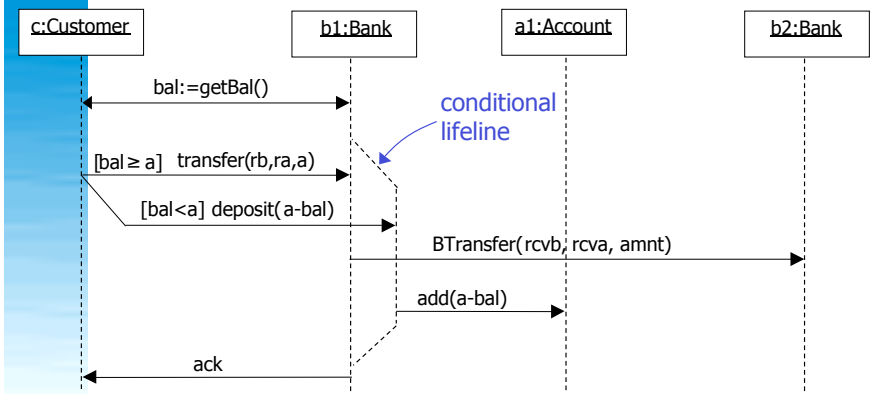
Sequence Diagrams: Alternatives

Boolean guard: occurrence condition for message



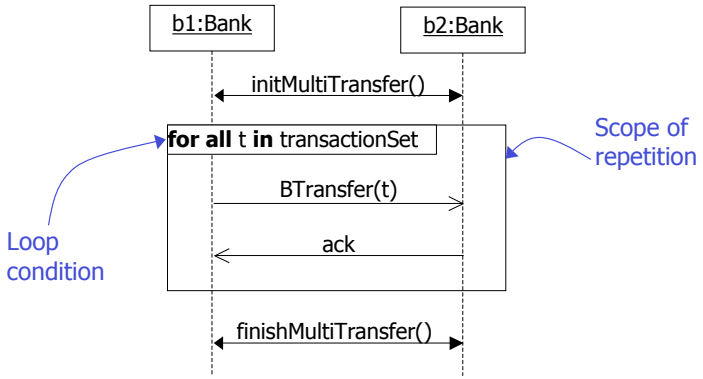
Sequence Diagrams: Alternatives

Conditional lifelines

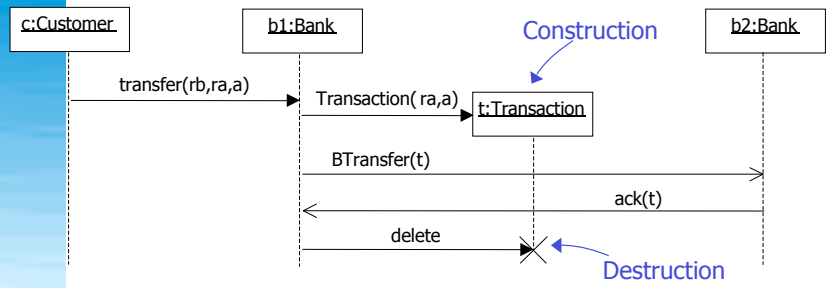


Sequence Diagrams: Repetition

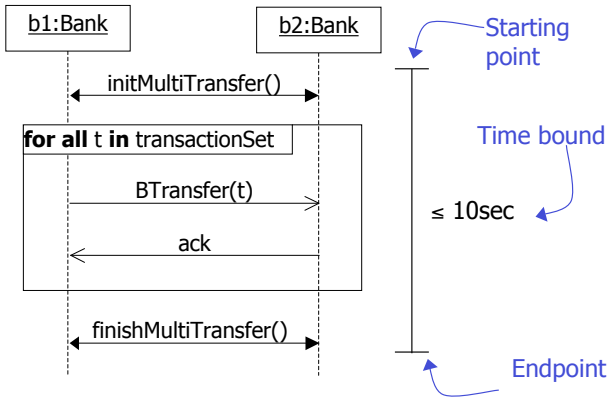
Enclosure with loop condition



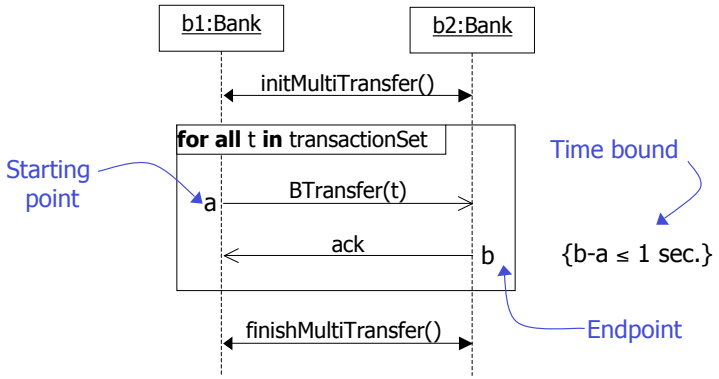
Sequence Diagrams: Construction and Destruction



Sequence Diagrams: Time Bounds

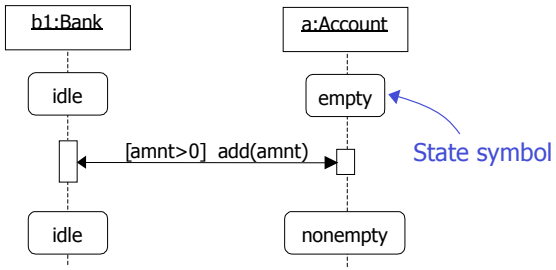


Sequence Diagrams: Time Bounds



Sequence Diagrams: State Markers

- Main use for SDs: “global” communication patterns
- Every interaction can change component state!



Summary Interaction Diagrams

- UML offers
  - Sequence diagrams
  - Collaboration diagrams

  for modeling communication patterns
- SDs support
  - neutral, asynchronous, synchronous communication
  - alternatives, repetition, object creation/deletion, time bounds, state symbols
- High potentials for interface specification (structure & behavior)