Aktivitätsdiagramme (Stichworte)

Udo Kelter

12.01.2005

Zusammenfassung dieses Lehrmoduls

Aktivitätsdiagramme (in UML Version 2) bieten vielfältige Sprachkonzepte, mit denen Abläufe und Algorithmen graphisch spezifiziert werden können, teilweise so detalliert und präzise, daß man die Spezifikationen ausführen kann. Dieses Lehrmodul stellt die wichtigsten Konzepte vor.

Vorausgesetzte Lehrmodule:

obligatorisch:
- Die Unified Modelling Language (UML) Version 2
- Petri-Netze

Stoffumfang in Vorlesungsdoppelstunden: 1.5

Inhaltsverzeichnis

1 Zweck von Aktivitätsdiagrammen

Zweck von Aktivitätsdiagrammen:

- Verhalten detailliert spezifizieren durch eine *Implementierung* (algorithmisch)
- sowohl sequentielle wie auch parallele Abläufmöglichkeiten darstellen (nicht nur einen konkreten Ablauf)

  z.B. Implementierung einer Operation darstellen

  $\sim$ visuelle Programmierung
- als Ergänzung zu Use-Cases oder Diagrammen anderen Typs

beinhalten Mischung diverser Konzepte:

- Programmablaufpläne und Nassi-Shneiderman-Diagramme
- Datenflußdiagramme
- erweiterte Petri-Netze

In UML 1.5 und 2 völlig verschieden

Diagrammelemente

1. Aktivitäten
2. Aktionen
3. Objektknoten
4. diverse Kontrollelemente: bedingte Verzweigungen, Schleifen, Transitionen

2 Aktionen

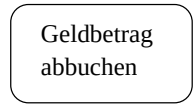
modellieren einen atomaren Vorgang, Strukturen innerhalb einer Aktion werden nicht mehr modelliert!

Darstellung: Rechteck mit gerundeten Ecken, innen:

- eine Bezeichnung oder

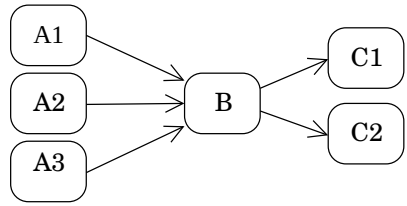
- eine Kurzbeschreibung oder
- Pseudocode für innere Details

Beispiel:



UML 1.5 bezeichnet dies als Aktivität, Verwechslungsgefahr!

einfacher Kontrollfluß zwischen Aktionen durch Pfeil, über das "Ausführungstoken" wandern können; Beispiel:



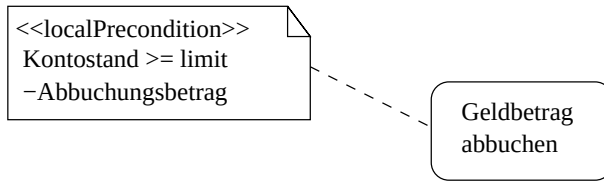
mehrere Eingangspfeile zu einer Aktion: es muß an jedem Eingang ein Token anliegen

mehrere Ausgangspfeile: auf allen Ausgängen werden Token erzeugt

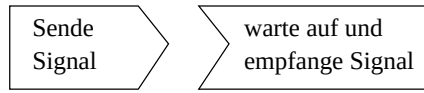
Vor- und Nachbedingungen

... müssen vor bzw. nach einer Aktionsausführung erfüllt sein. Notation: Notizzettel mit Stereotyp «`localPrecondition`» bzw. «`localPostcondition`»

Beispiel:



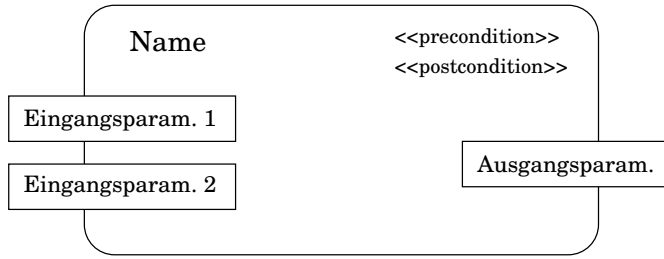
Signale und Ereignisse Senden und Warten werden durch spezielle Aktionen dargestellt (*SendSignalAction* bzw. *AcceptEventAction*):



ein Empfänger ist *immer empfangsbereit* (braucht keinen Eingangspfeil) und generiert einen Ablauftoken, wenn ein Signal eintritt

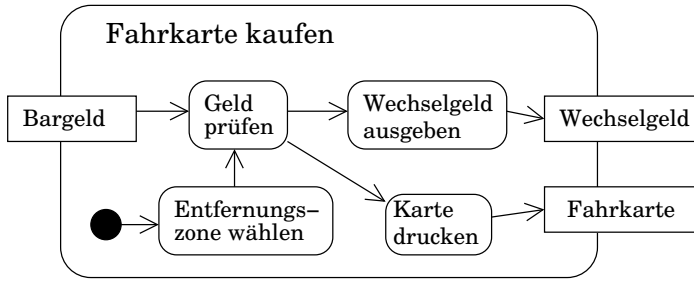
3 Aktivitäten

modellieren: innere, eingeschachtelte Aktivität in einem Aktivitätsdiagramm



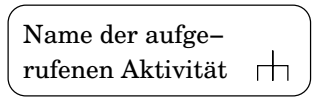
Darstellung:
- Kasten mit gerundeten Ecken, Name links oben
- Eingangs- bzw. Ausgangsparameter durch Objektsymbole
- Vor- und Nachbedingungen rechts oben, mit Stereotypen
- Innenbereich = Zeichenfläche für Realisierung der Aktivität

Beispiel:



statt eingeschachtelter Aktivität oft übersichtlicher: separates Diagramm und Aufruf

Aufruf einer Aktivität: durch Aktion, in der der Name der Aktivität steht + Aufrufsymbol ("Harke")

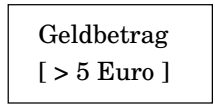


Parameterübergabe: durch "Pins" (Anzahl und Typ muß passen)

4 Objektknoten

repräsentiert einen "Kanal" / Datenfluß, durch den Daten bestimmten Typs wandern → tritt i.d.R. nur zwischen zwei Aktionen auf

Darstellung:Kasten, innen der Typname, ggf. in [...] zusätzliche Einschränkungen
Beispiel:

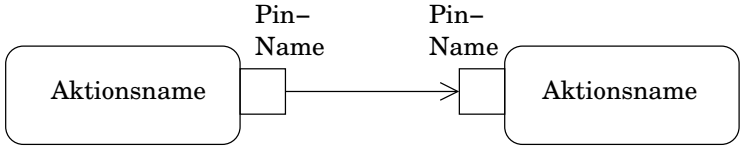


nichtatomarer Typ wird in Klassendiagramm spezifiziert

(möglich, aber hier nicht behandelt: Mischung von Datenflüssen, streaming Knoten u.a.)

4.1 Pin-Notation von Parametern

... für Objektknoten, die Ein- oder Ausgabeparameter repräsentieren



Pin ist nur andere Notation für einen Objektknoten

Objektknoten als Puffer
Normalverhalten: genau 1 Wert zu einem Zeitpunkt
Objektknoten als Puffer: Angabe der Größe durch eine Merkmalsangabe {upperBound=...}

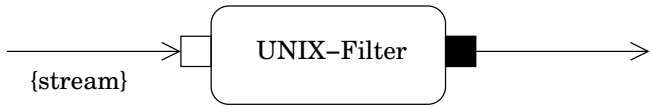
4.2 Streaming-Parameter

Normalverhalten von Ein- oder Ausgabeparametern: Datentoken wird beim Start der Aktion verbraucht bzw. am Ende erzeugt, zwischendurch passiert nichts an den Schnittstellen

Streaming-Parameter: verarbeiten bzw. erzeugen einen Strom von Datentoken *während der Aktionsausführung*

Darstellung:
- Pin schwarz ausgefüllt oder
- Merkmalsangabe {stream} an der Ein- oder Ausgangskante

Beispiel:

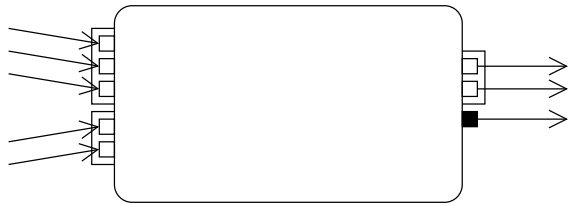


4.3 Parametersätze

Motivation:

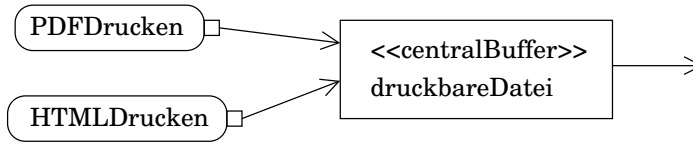
- wenn eine Aktion mehrere Eingangsparameter hat, müssen *überall* Token anliegen, damit die Aktion starten kann
- oft reicht eine "sinnvolle Teilmenge" der Parameter = **Parametersatz**

Darstellung: durch Einrahmen der Eingangsparameter

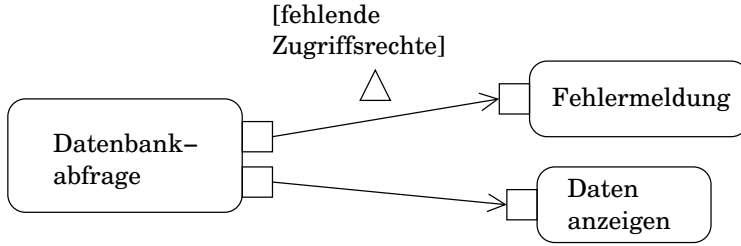


Parameter, die *keinem* Parametersatz zugehören, müssen als Streaming-Parameter angegeben sein

Objektknoten, der mehrere Datenflüsse vereinigt
Knoten mit Stereotyp: «`centralBuffer`»
Darf nur mit anderen Objektknoten verbunden sein



Exception-Objekte werden durch Dreieck gekennzeichnet

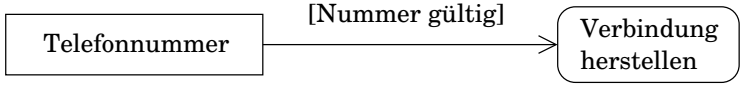


5 Kanten

Arten von (stets gerichteten) Kanten:

- Kontrollflußkanten: zwischen zwei Aktionen oder einer Aktion und einem Kontrollflußelement
  tragen keine Daten, triggern nur die Ausführung der Folgeaktion
- Objektflußkanten: mindestens ein Objektknoten involviert

Bedingungen
Angabe in eckigen Klammern; Beispiel:



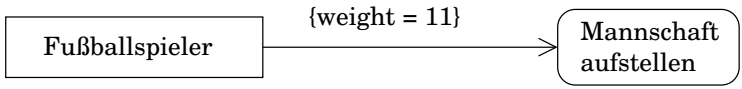
Ablauf über diese Kante nur erlaubt, wenn die Bedingung erfüllt ist

Bedingungen treten meist an Kanten hinter Verzweigungsknoten auf

Gewichtete Kanten

Bedeutung wie bei Stellen-Transitions-Netzen

Angabe des Gewichts einer Kante mit Schlüsselwort *weight* als Merkmalsangabe an der Kante; Beispiel:



Sprungmarken für Kanten

- haben eindeutige Namen
- treten immer paarweise auf
- Darstellung: kleiner Kreis

Beispiel:

6 Kontrollelemente

leiten Token zwischen Aktionen weiter, können aber selbst keine Token speichern[1]

[1] mit einer Ausnahme: in einem Startknoten entsteht ein neuer Token, der u.U. nicht weitergeleitet werden kann – wahrscheinlich ein falsch konstruiertes Diagramm

6.1 Startknoten

Notation: dicker schwarzer Punkt

erzeugt beim Start einen Token; bei mehreren ausgehenden Kanten 1 Token pro Kante, die alle parallel losgeschickt werden

6.2 Endknoten

Arten von Endknoten:

- für Aktivitäten: beendet Aktivität komplett

  Darstellung: eingekreister dicker schwarzer Punkt

  mehrere Aktivitätsendknoten erlaubt, Aktivität endet, sobald einer Aktivitätsendknoten erreicht wird

- für Kontrollflüsse: löscht nur den dort ankommenden Token, beendet zug. Ablaufstrang

  Darstellung: Kreis mit stilisiertem X innen

  generell: keine ausgehenden Kanten erlaubt

6.3 Verzweigungs- und Verbindungsknoten

Graphische Darstellung:

Verzweigungsknoten Verbindungsknoten

Verzweigungsknoten: hat ähnliche Wirkung wie switch- bzw. case-Anweisung

- Bedingungen stehen an den weiterführenden Kanten

- müssen logisch disjunkt sein
- müssen alle Fälle abdecken
- max. 1 Kante mit Bedingung [else] erlaubt; deckt nicht explizit genannte Fälle ab

ankommendes Token wird auf die zutreffende Kante weitergeleitet

Verbindungsknoten: leitet an eingehenden Kanten liegende Token weiter

wenn Token an mehreren Eingängen liegen: zufällige Auswahl eines der Knoten

6.4 Parallelisierungs- und Synchronisationsknoten

entsprechen weitgehend Transitionen in (erweiterten) Petri-Netzen

Graphische Darstellung:

Parallelisierungsknoten Synchronisationsknoten

Parallelisierungsknoten: ankommende Kontroll- bzw. Datentoken werden auf alle Ausgänge dupliziert

Bedingungen an Eingangs- und Ausgangskanten zulässig

Synchronisationsknoten: schaltet, wenn an allen Eingangskanten Token anliegen

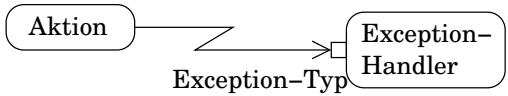
- überall Kontrolltoken → Weitergabe eines Kontrolltokens
- wenigstens 1 Datentoken liegt an irgendeiner Eingangskante an → Löschung aller Kontrolltoken, Weitergabe aller (!) Datentoken

Auch anderes Synchronisationsverhalten angebbar mit Merkmalsangabe {joinspec = ... }

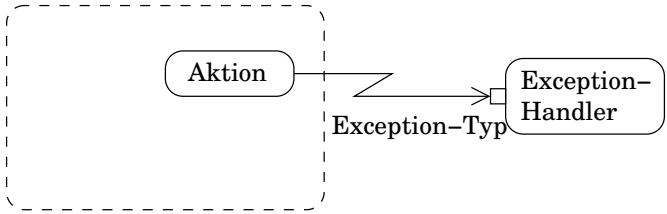
6.5 Exception-Handler

bei einer Ausnahme wird die normale Abarbeitung einer Aktion abgebrochen, sofern vorhanden, übernimmt ein Exception-Handler die weitere Bearbeitung (auch mit den Daten der abgebrochenen Aktion); er bestimmt auch die Rückgabewerte

Graphische Darstellung: gezackter Pfeil

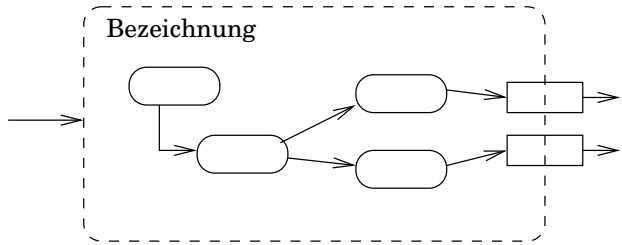


Unterbrechungsbereiche: wenn der Bereich über eine Unterbrechungskante verlassen wird, werden *alle* Aktionen bzw. Abläufe in diesem Bereich abgebrochen



7 Strukturierte Knoten

Beispiel:



stehen zwischen Aktionen und Aktivitäten:

- können Datenpins haben, Ausführung der inneren Aktionen beginnt erst, wenn an allen Datenpins Datentoken anliegen
- können wie eine Aktion in Kontrollflüsse eingebaut werden
- können aber nicht aufgerufen werden
- können hierarchisch geschachtelt werden

Graphische Darstellung: ähnlich wie Aktivität, aber gestrichelte Umrandung; Umrandung kann Ziel eines Kontrollflusses sein

Untertypen:

- Mengenverarbeitungsbereiche
- Schleifenknoten
- Entscheidungsknoten

7.1 Schleifenknoten

Schema:

- sind spezielle strukturierte Knoten
- sind in drei Bereiche eingeteilt, Schlüsselwörter for, while, do

for-Bereich: dient zur Forbereitung (??!?) der Schleifenausführung

- wird nur einmal zu Beginn ausgeführt
- ist optional
- hat nichts mit for-Konstrukten in vielen Programmiersprachen zu tun!

do-Bereich: = Schleifenrumpf

while-Bereich:

- kann *vor* oder *nach* do-Bereich stehen
- erzeugt letztlich einen Booleschen Wert als Ausgabe einer Aktion; der Ausgabeparameter dieser Aktion ist mit einer kleinen Raute markiert
- übliche Bedeutung: Schleifenknoten wird verlassen, wenn negativer Test
- kann auch Berechnungen durchführen, sollte aber keine Aktionen enthalten, die besser in den Schleifenrumpf (do-Bereich) gehören
- ist optional; while-Bereich fehlt = while (true) .... [Abbruch in diesem Fall mit Unterbrechungskanten]

Ablaufsteuerung innerhalb der Bereiche:

Front-end-Knoten: Knoten ohne eingehende Kante

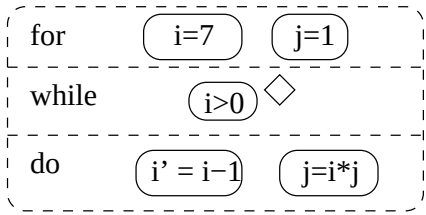
alle Front-end-Knoten erhalten ein Token bei Aktivierung eines Bereichs → parallele Ausführbarkeit

Back-end-Knoten: Knoten ohne ausgehende Kante

Terminierung des Bereichs, wenn alle Back-end-Knoten durchlaufen worden sind (∼ impliziter Synchronisationsknoten hinter allen Back-end-Knoten)

Variablen ... werden sonst in Aktivitätsdiagrammen nicht benötigt (sind ersetzt durch Datenflußkanten)

in Schleifen: Wiedereinführung von Variablen; Beispiel:



Problem mit parallelen Zweigen und Wertzuweisungen an Variablen in den Zweigen

Notation: i' = i-1 : neuer Wert wird erst nach Beendigung des Bereichs in i lesbar

Syntax in Anlehnung an Programmiersprachen (Pseudocode) → Korrektheitsüberprüfung schwierig bis unmöglich

7.2 Entscheidungsknoten

ist ein spezieller strukturierter Knoten

Schema:

if-Bereich: erzeugt Booleschen Wert, Markierung mit Raute, analog wie while-Bereich

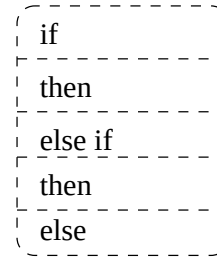
mehrere if-Bereiche in einem Entscheidungsknoten sind erlaubt:

- werden *parallel* (!!) durchlaufen
  (obwohl sie hintereinander stehen - unintuitiv)
- sobald ein Test positiv endet, Abbruch aller noch laufenden Tests (auch solcher, die positiv enden würden)
  → nichtdeterministisches Verhalten
- bei disjunkten Bedingungen: ersetzt case- oder switch-Anweisung aus Programmiersprachen

then-Bereich: steht direkt hinter if-Bereich
führt Berechnungen durch, analog zu do-Bereich in Schleifenknoten

else if-Bereich: ist nur hinter einem if-Bereich (oder else if-Bereich) und dem folgenden then-Bereich erlaubt, dahinter folgt der zug. then-Bereich

wird durchlaufen, wenn der davorstehende if-Test (oder else if-Test) negativ verlief



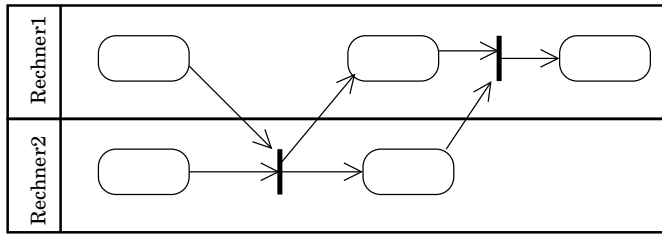
else-Bereich:

- nur 1* als letzter Bereich in einem Entscheidungsknoten erlaubt
- ist optional
- wird durchlaufen, wenn keine der if-Tests und else if-Tests positiv verlief

8 Aktivitätsbereiche

Aktivitätsbereiche (Partitionen) dienen dazu, Aktionen bestimmten Rollen, Verantwortlichen, Subsystemen ö.ä. zuzuordnen und optisch zu gruppieren

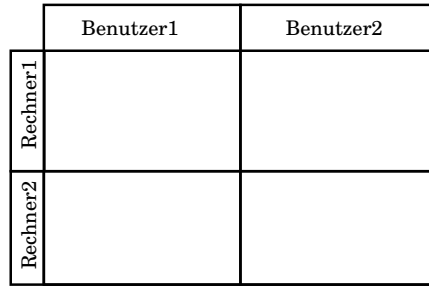
Beispiel:



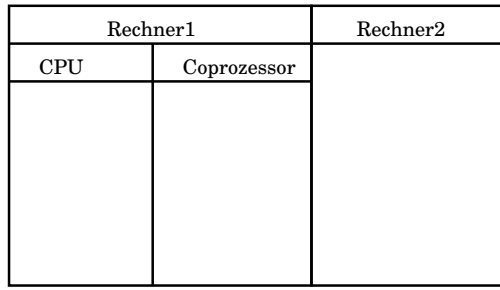
Alternativ: Rolle in Klammern oben im Aktionsknoten notieren

Strukturierung:

a) mehrere (max. 2?)) Dimensionen



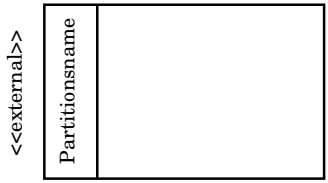
b) hierarchisch geschachtelt

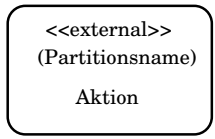


Externe Aktionen ... gehören eigentlich nicht zum modellierten System, werden aber zum besseren Verständnis dargestellt

Stereotyp: ≪`external`≫

Darstellungen:





Literatur

[PN] Kelter, U.: Lehrmodul “Petri-Netze”; 2003