CHRISTIAN HUEMER
MARION SCHOLZ

Objektorientierte Modellierung mit UML
Teil IV - Aktivitätsdiagramm

Aktivitätsdiagramm

Das Aktivitätsdiagramm

Christian Huemer und Marion Scholz

Einführung

- Fokus auf **prozedurale Verarbeitungsaspekte**
- Spezifikation von **Kontroll-** und/oder **Datenfluss** zwischen Arbeitsschritten (Aktionen) zur Realisierung einer Aktivität
- **Aktivitätsdiagramm in UML2:**
  - ablauforientierte Sprachkonzepte
  - basierend u.a. auf Petri-Netzen und BPEL
- Sprachkonzepte und Notationsvarianten decken ein **breites Anwendungsgebiet** ab
  - Modellierung objektorientierter und nichtobjektorientierter Systeme
  - Neben vorgeschlagener grafischer Notation sind auch beliebige andere Notationen (z.B. Pseudocode) erlaubt

Aktivitätsdiagramm

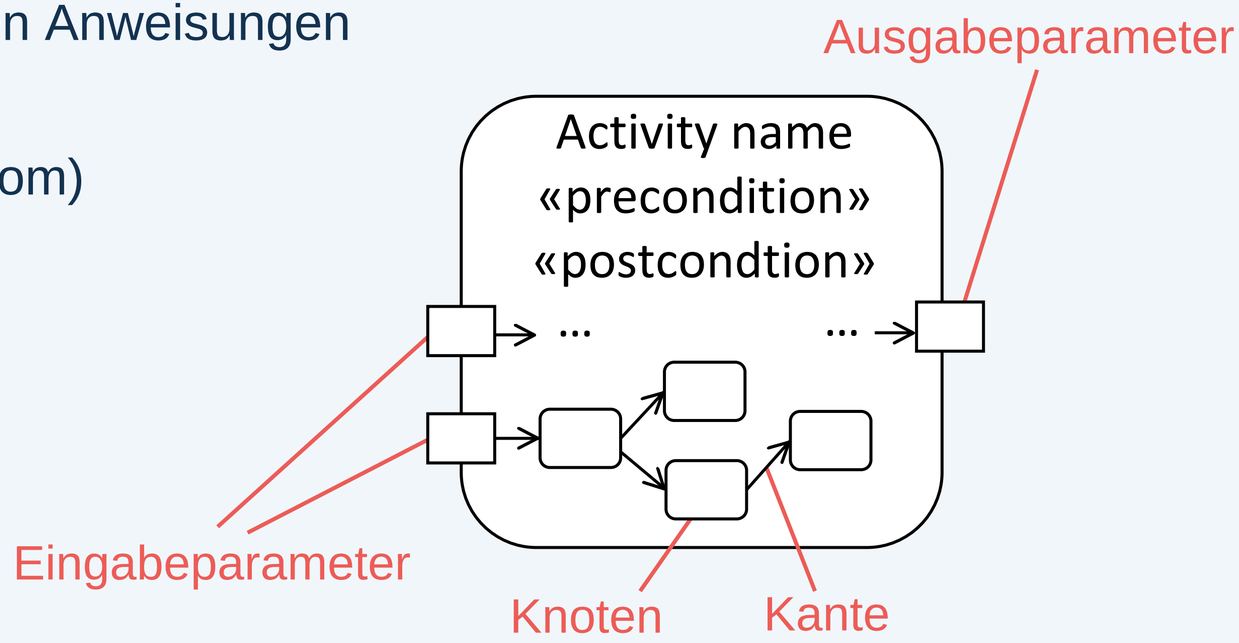
Die Aktivitäten, die Aktionen und deren Übergänge

Christian Huemer und Marion Scholz

Aktivität

Activity A

- Eine Aktivität ist ein gerichteter Graph
  - Knoten: Aktionen (bzw. Aktivitäten) und Objekte
  - Kanten: Kontroll- und Datenflüsse
- Kontroll- und Datenflüsse legen potentielle »Abläufe« fest
- Spezifikation von benutzerdefiniertem Verhalten auf unterschiedlichen Granularitätsebenen Beispiele:
  - Definition einer Operation in Form von einzelnen Anweisungen
  - Ablauf eines Anwendungsfalls
  - Spezifikation eines Geschäftsprozesses (autonom)
- optional:
  - Parameter (z.B. wie bei Operationen)
  - Vor- und Nachbedingungen, die bei Beginn bzw. bei Beendigung der Aktivität gelten müssen

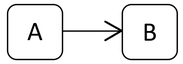


Aktionen

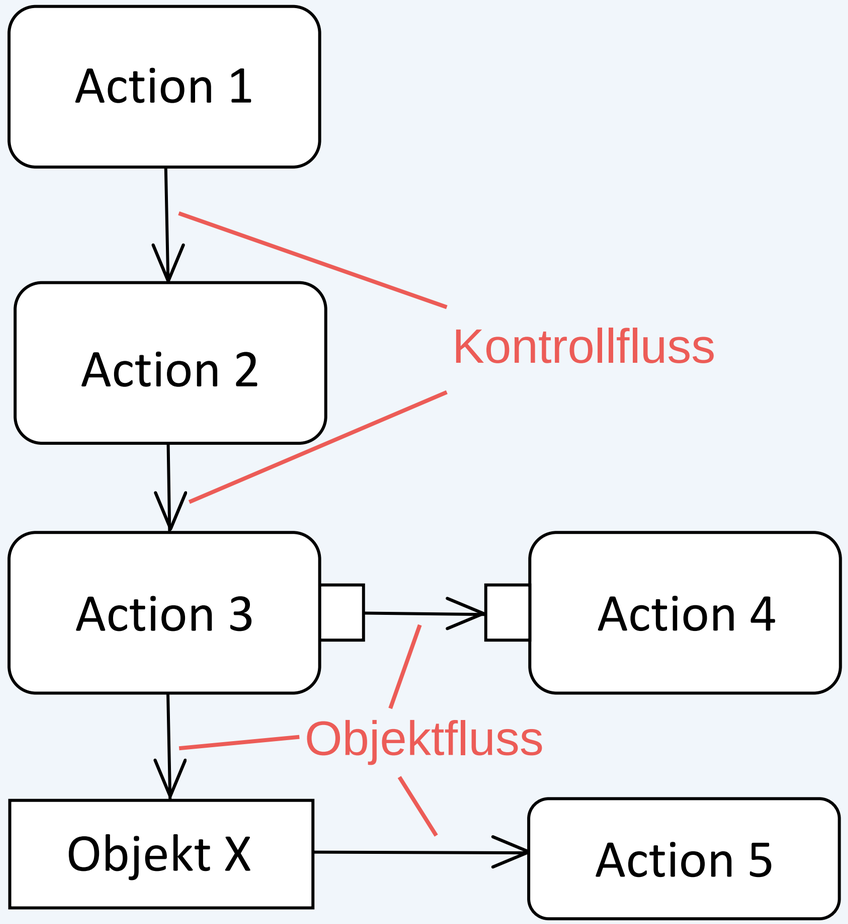
Action A

- **Elementare Bausteine**
- **Atomar**, können aber abgebrochen werden
- **Sprachunabhängig**, allerdings Definition in beliebiger Programmiersprache möglich
- Aktionen können Eingabewerte zu Ausgabewerten verarbeiten
- Spezielle **Notation** für 44 verschiedene Aktionsarten
- **Kategorisierung** der vordefinierten Aktionen:
  - Kommunikationsbezogene Aktionen (z.B. Signale und Ereignisse)
  - Objektbezogene Aktionen (z.B. Erzeugen und Löschen von Objekten)
  - Strukturmerkmals- und variablenbezogene Aktionen (z.B. Setzen und Löschen einzelner Werte von Variablen)
  - Linkbezogene Aktionen (z.B. Erzeugen und Löschen von Links zwischen Objekten sowie Navigation)

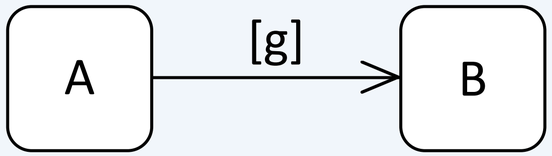
Kanten



- Kanten verbinden Knoten und legen **mögliche Abläufe** einer Aktivität fest
  - Kontrollflusskanten
    - Drücken eine **reine Kontrollabhängigkeit** zwischen Vorgänger- und Nachfolgerknoten aus
  - Objektflusskanten
    - Transportieren zusätzlich Daten und drücken dadurch auch eine **Datenabhängigkeit** zwischen Vorgänger- und Nachfolgerknoten aus



- Überwachungsbedingung (guard)
  - Bestimmt, ob Kontroll- und Objektfluss weiterläuft oder nicht



Aktivitätsdiagramm

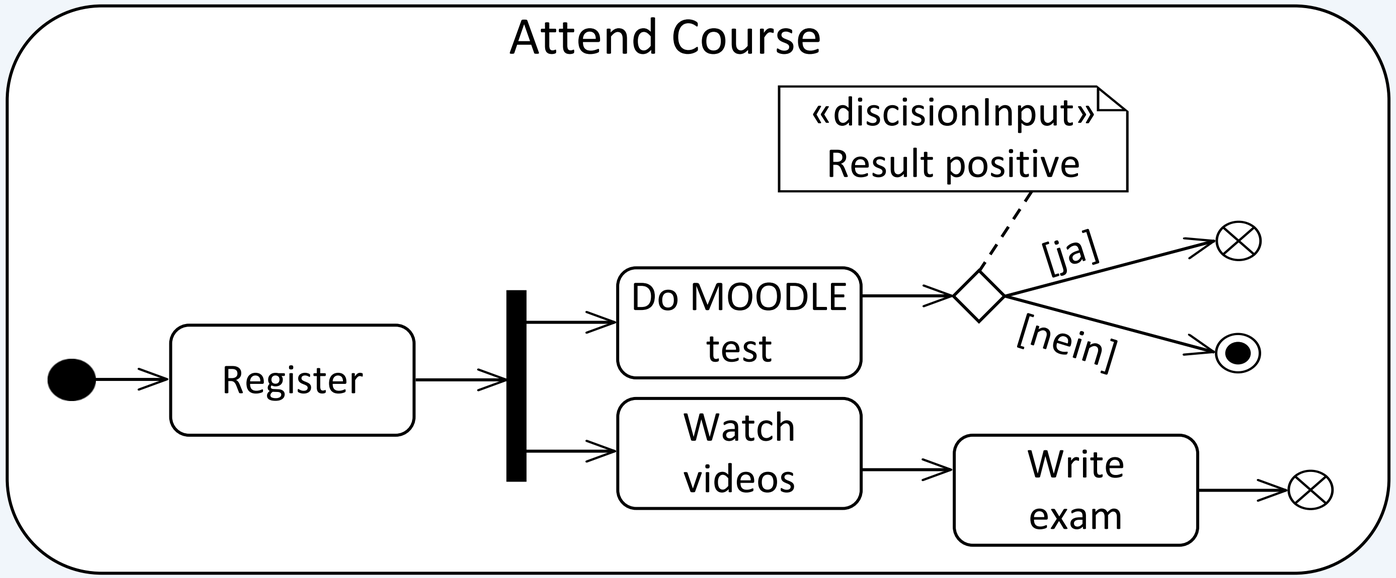
Der Start und das Ende von Abläufen

Christian Huemer und Marion Scholz

Start und Ende von Aktivitäten und Abläufen

- Initialknoten
  - Beginn eines Aktivitätsablaufs
  - Versorgt alle ausgehenden Kanten mit Kontrolltoken
  - Aufbewahrung von Token erlaubt, aber Überwachungsbedingungen blockieren ev. Weitergabe
  - Pro Aktivität keine oder mehrere Initialknoten erlaubt
- Aktivitätsendknoten
  - Beendet alle Abläufe einer Aktivität sowie den Lebenszyklus eines Objekts
  - Der erste Token, der zu einem Endknoten gelangt, beendet die Aktivität
  - Keine Ausführung weiterer Aktionen
  - Kontrolltoken werden gelöscht, Datentoken an Ausgabepins der Aktivität dagegen nicht
  - Pro Aktivität mehrere Aktivitätsendknoten erlaubt
- Ablaufendknoten
  - Beendet einen Ablauf einer Aktivität

Bsp.: Absolvieren einer LVA



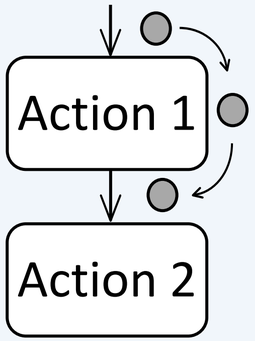
Aktivitätsdiagramm

Der Token und alternative Abläufe

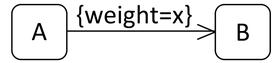
Christian Huemer und Marion Scholz

Token

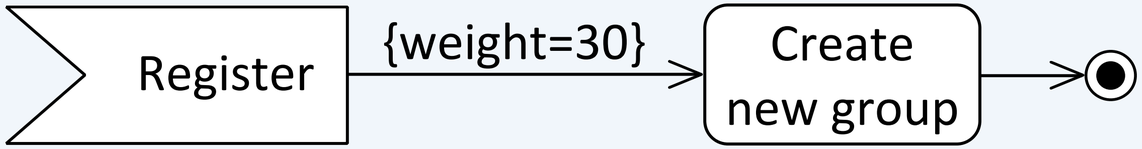
- »**Virtueller Koordinationsmechanismus**«
  zur Beschreibung von Aktivitätsabläufen
- Token beschreibt möglichen Ablauf einer Aktivität
- Token fließen entlang der Kanten von Vorgänger- zu Nachfolgerknoten
- Aktion startet wenn an allen eingehenden Kanten ein Token liegt
- Nach Durchführung der Aktion wird an alle ausgehenden Kanten ein Token weitergegeben
- Überwachungsbedingung kann Weitergabe von Token verhindern
- Unterscheidung in Kontroll- und Datentoken
  - **Kontrolltoken**:
    "Ausführungserlaubnis" für den Nachfolgeknoten
  - **Datentoken**:
    Transport von Datenwert oder Referenz auf Objekt



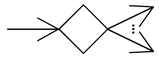
Das Gewicht einer Kante



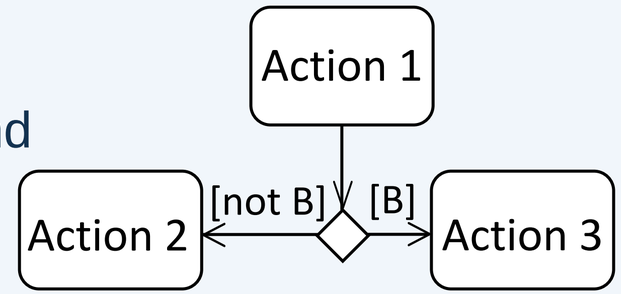
- Minimale Anzahl an Token, die anliegen müssen, damit eine Aktion ausgeführt wird
- Default: **1**

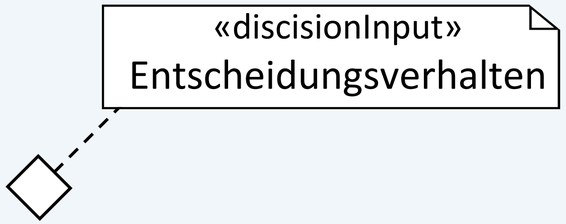


Alternative Abläufe - Entscheidungsknoten

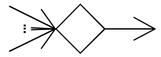


- Definiert alternative Zweige und repräsentiert eine »Weiche« für den Tokenfluss
  - Verwendung auch zur Modellierung von Schleifen
- Überwachungsbedingungen
  - Wählen den Zweig aus
  - Wechselseitig ausschließend
  - **`[else]`** ist vordefiniert
- Entscheidungsverhalten
  - Ermöglicht detailliertere Spezifikation der Auswahlentscheidung an zentraler Stelle
  - Ankunft von Token startet das Entscheidungsverhalten – Datentoken fungieren als Parameter



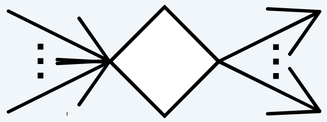


Alternative Abläufe - Vereinigungsknoten

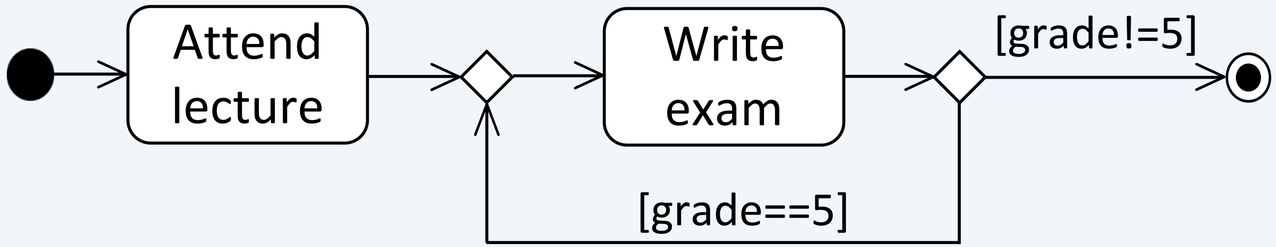


- Ein Vereinigungsknoten führt alternative Abläufe wieder zusammen
- Token werden, sobald möglich, an den Nachfolgerknoten weitergereicht

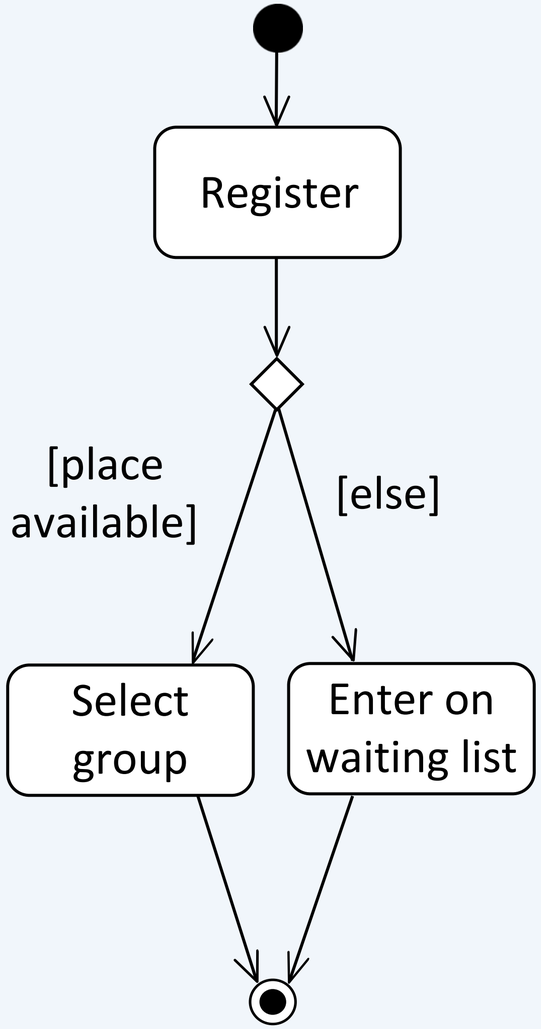
- Kombinierter Entscheidungs- und Vereinigungsknoten



- Bsp.:



Alternative Abläufe – Bsp.



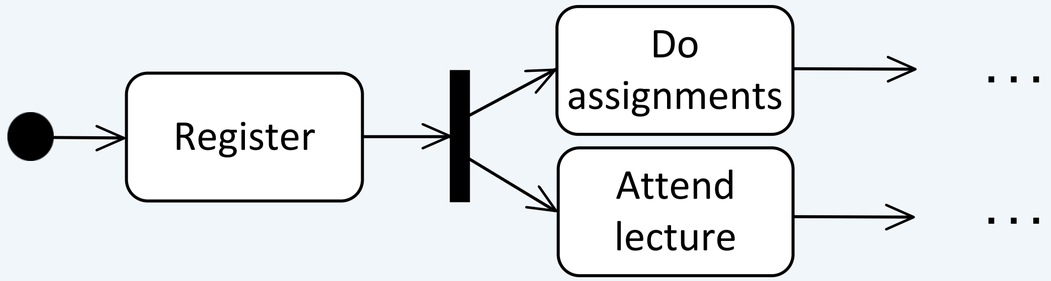
Aktivitätsdiagramm

Der Token und nebenläufige Abläufe

Christian Huemer und Marion Scholz

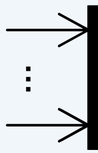
Nebenläufige Abläufe - Parallelisierungsknoten

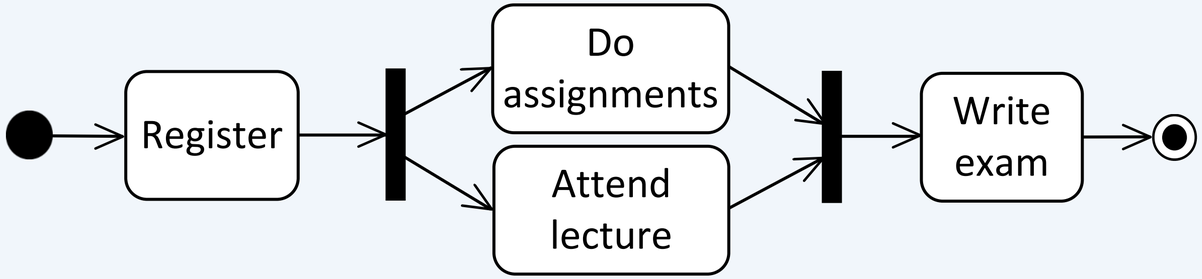
- Zur Modellierung der Aufspaltung von Abläufen
- Eingehende Token werden für alle ausgehenden Kanten dupliziert, sobald zumindest eine Überwachungsbedingung diese akzeptiert
- Nichtakzeptierte Token werden aufbewahrt
- Bsp.:



Nebenläufige Abläufe – Synchronisierungsknoten

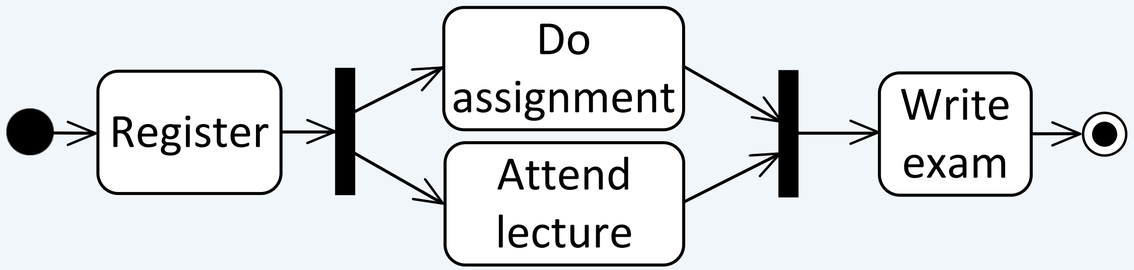
- Führt nebenläufige Abläufe zusammen
- Tokenverarbeitung
  - Vereinigung der Token, sobald an allen Kanten vorhanden
  - Kontrolltoken verschiedener Kanten werden vereinigt und nur ein einzelnes Token weitergereicht
  - Datentoken werden alle weitergereicht
  - Bei Kontroll- und Datentoken werden nur Datentoken weitergereicht
- Kombinierter Parallelisierungs- und Synchronisierungsknoten:

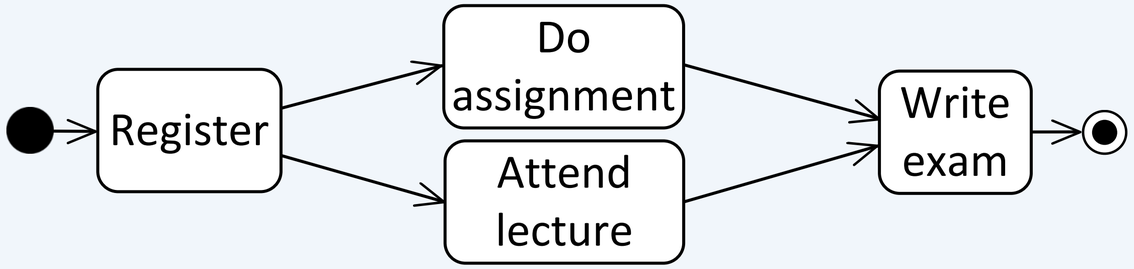




Bsp.: Alternative Modellierungen

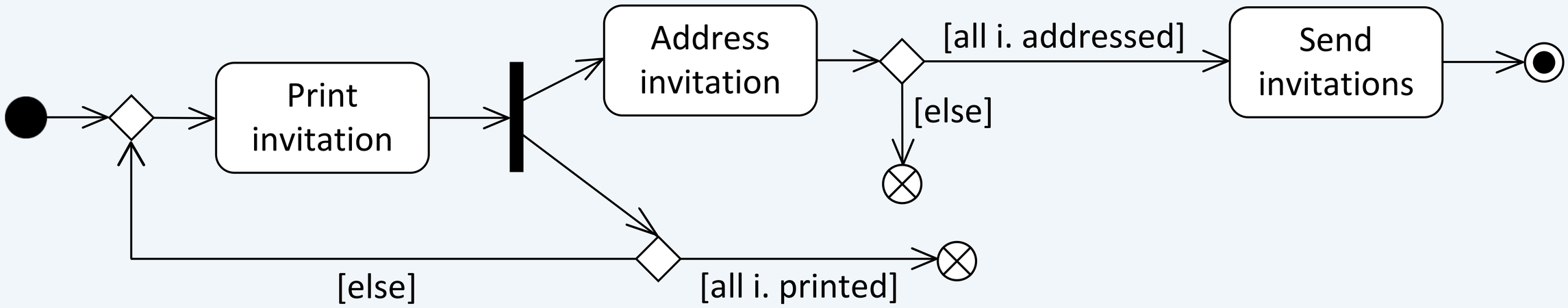
- Äquivalenter Kontrollfluss



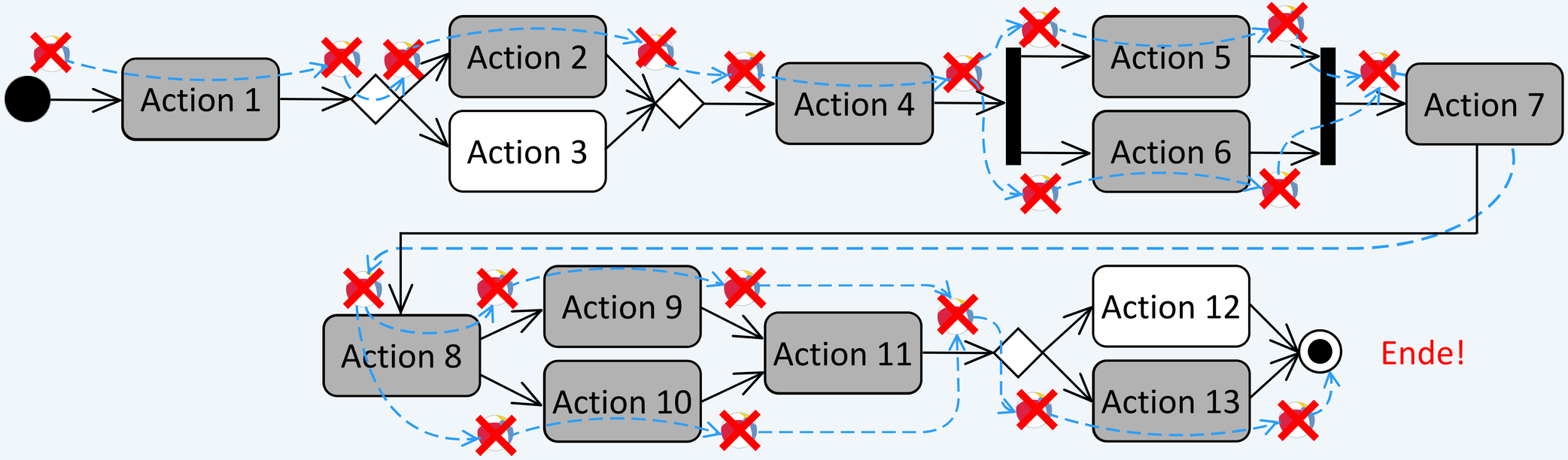


Bsp.: Erstellen und Versenden von Einladungen zu einem Treffen

- Während neue Einladungen gedruckt werden, werden bereits gedruckte Einladungen adressiert.
- Sobald alle Einladungen adressiert sind, werden sie verschickt.



Token – Beispiel (Kontrollfluss)



Aktivitätsdiagramm

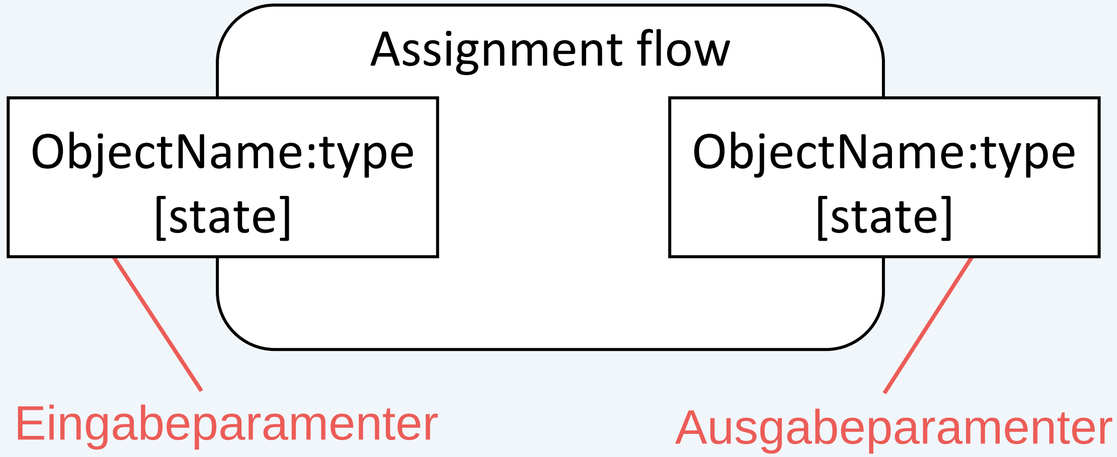
Der Objektknoten

Christian Huemer und Marion Scholz

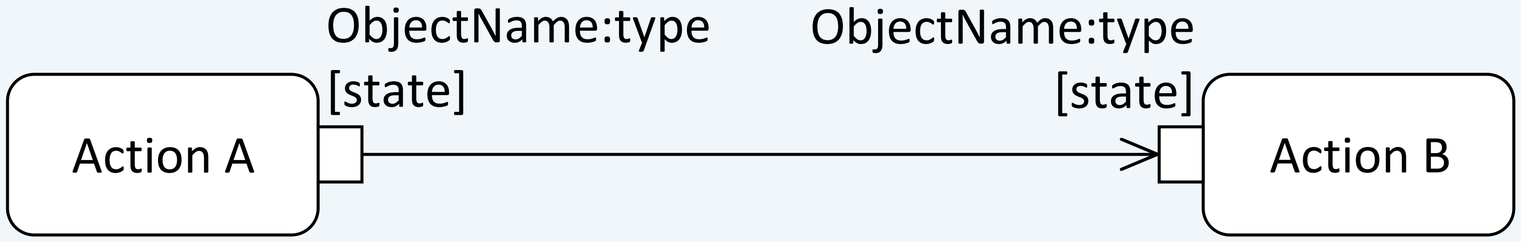
Objektknoten

Object

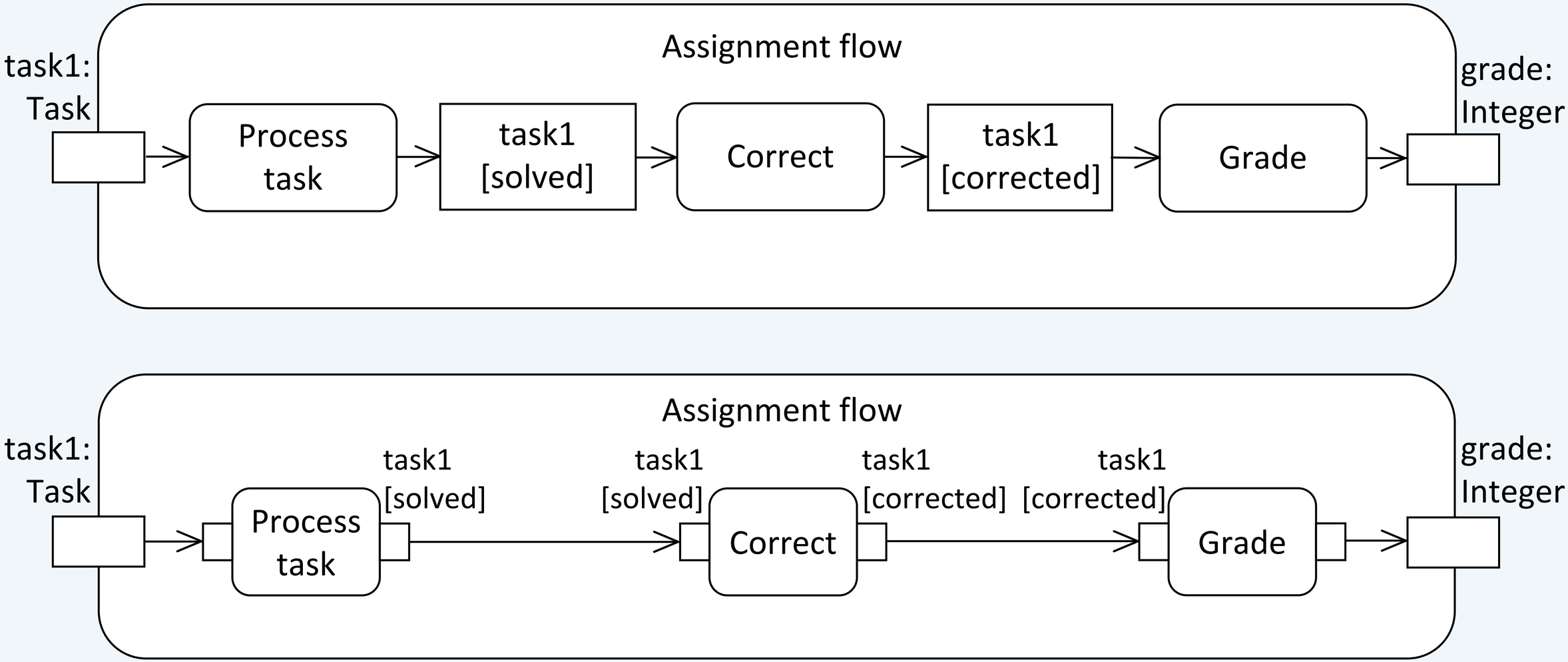
- Inhalt: **Datentoken**
- Objektknoten verbinden Aktionen über Objektflüsse
- Inhalt ist **Ergebnis einer Aktion** und **Eingabe** für eine weitere Aktion
- Typangabe und Zustandseinschränkung sind optional
- Objektknoten als **Ein-/Ausgabeparameter**
  - für Aktivitäten (activity parameter node)



  - für Aktionen (pins)



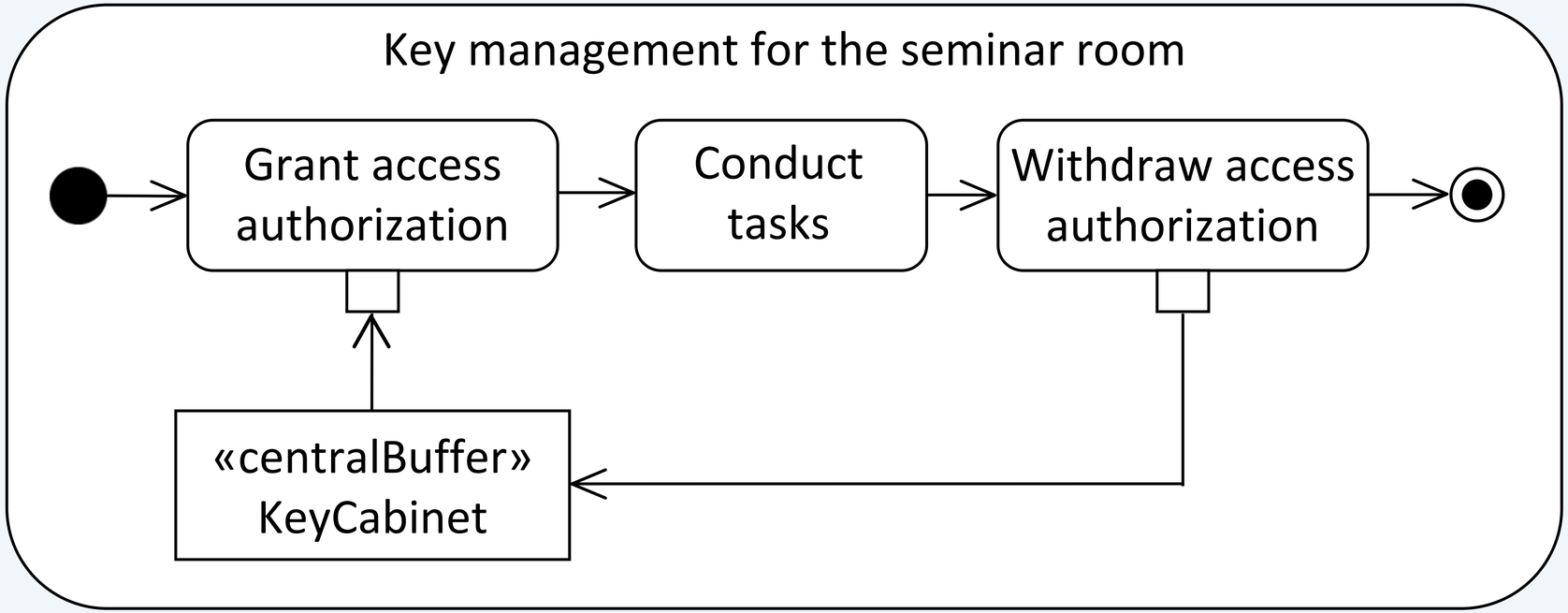
Objektknoten bei Aktionen: 2 Notationsvarianten



Central Buffer

«centralBuffer»
CB

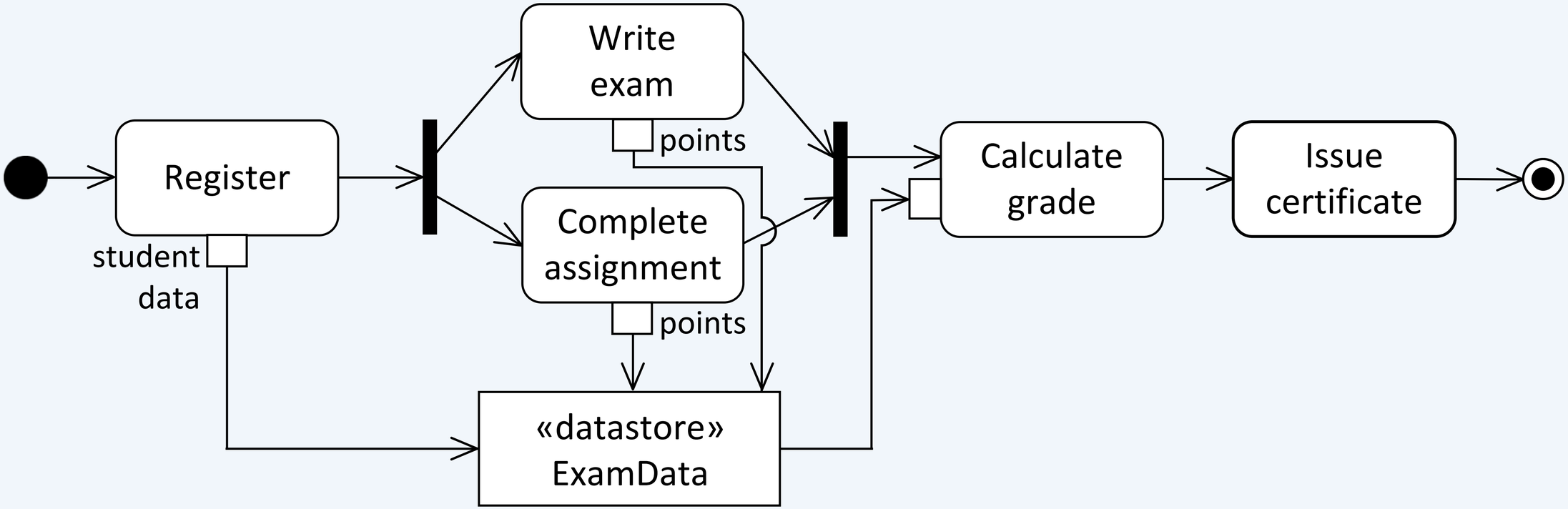
- Zentrale Pufferung von Datentoken
- Zur Speicherung und Weitergabe von Objekt-Token
- Akzeptiert Objekt-Token von Objektknoten und gibt sie an andere Objektknoten weiter
- Transienter Pufferknoten
  - Löscht Datentoken, sobald er sie weitergegeben hat



Data Store

«datastore»
DS

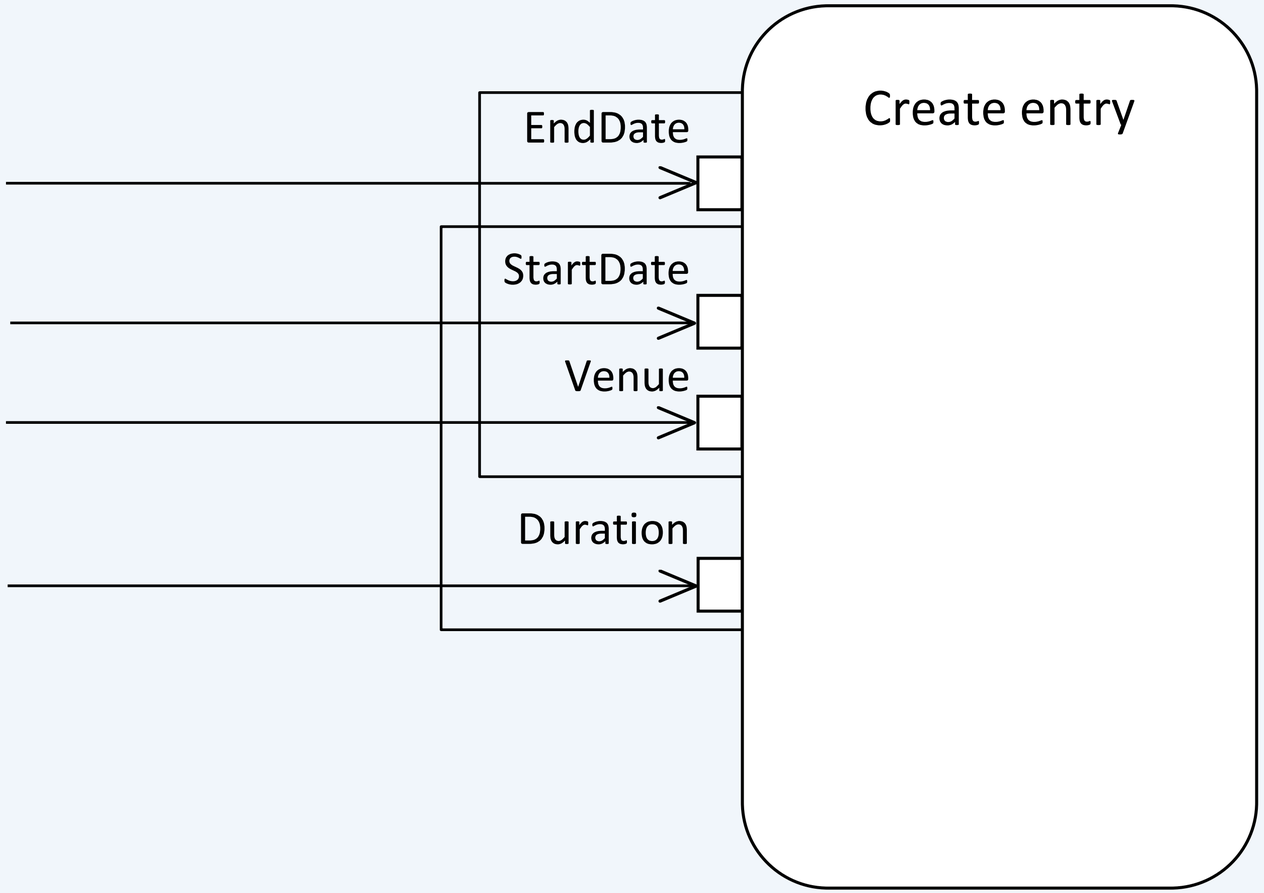
- Zur Speicherung und Weitergabe von Objekt-Token
- Permanenter Speicher
  - Bewahrt Datentoken auf und gibt Duplikate weiter
- Keine Mehrfachspeicherung identer Objekte
- Explizites »Abholen« der Datentoken möglich



Parametersatz

- Gruppierung von Parametern
- Alternative Gruppen von Ein- bzw. Ausgabewerten

- Bsp.: 2 Arten von Terminen



Aktivitätsdiagramm

Der Objektfluss und die Partitionen und Signale und Ereignisse

Christian Huemer und Marion Scholz

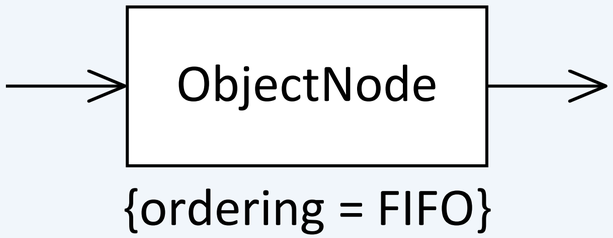
Objektfluss (1/4)

- **Transport-** und **Kontrollfunktion**
- **Verknüpft** Aktionen nicht direkt, sondern **über Objektknoten**
- Objektknoten bestimmen den Typ der zu transportierenden Objekte

- **Steuerungsmöglichkeiten** der Weitergabe von Datentoken:
  - Reihenfolge
  - Kapazitätsobergrenze und Gewicht
  - Selektionsverhalten
  - Transformationsverhalten

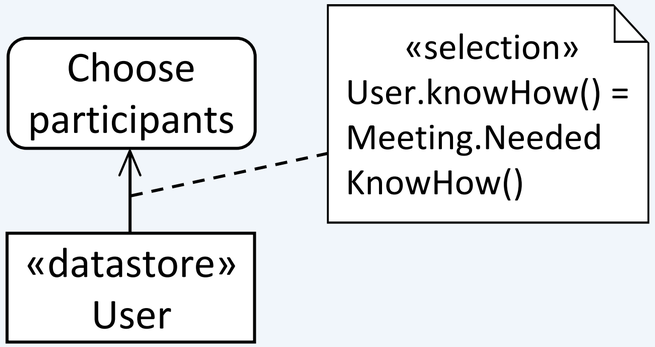
Objektfluss (2/4) – Reihenfolge der Tokenweitergabe

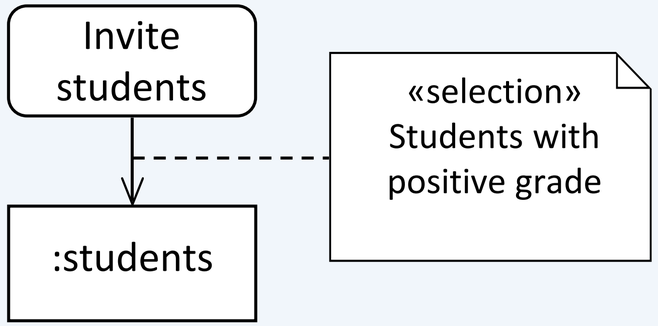
- Explizites Festlegen der Reihenfolge, in der ein Datentoken weitergegeben wird
  - **FIFO** (first in, first out) - {ordering = FIFO}
  - **LIFO** (last in, first out) - {ordering = LIFO}
  - **Geordnet** - {ordering = ordered}
    - benutzerdefinierte Reihenfolge (Angabe von Selektionsverhalten)
  - **Ungeordnet** - {ordering = unordered}
    - Reihenfolge in der die Token eingehen, hat keinen Einfluss auf die Reihenfolge, in der sie weitergereicht werden



Objektfluss (3/4) - Selektionsverhalten

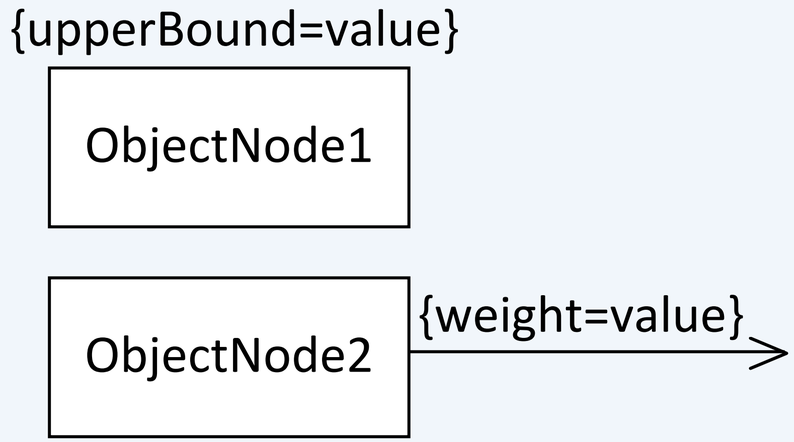
- Wählt bestimmte Token zur Weitergabe aus
- Objektknoten und Objektflusskanten können Selektionsverhalten aufweisen
- Beispiele:



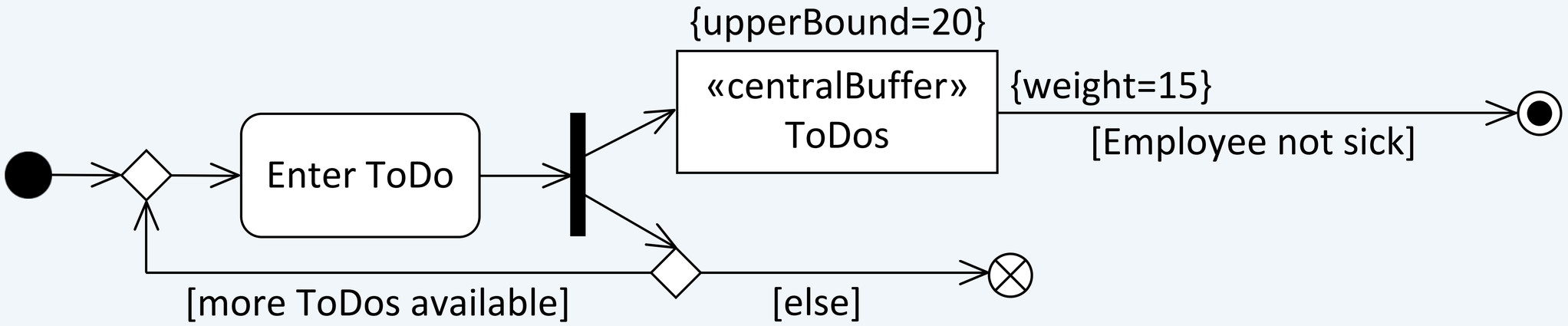


Objektfluss (4/4)

- **Kapazitätsobergrenze** eines Objektknotens
  - max. Anzahl von Token, die sich zu einem Zeitpunkt in diesem Knoten befinden dürfen
- **Gewicht** einer Objektflusskante:
  - Anzahl der Token die anliegen müssen, bevor sie an Nachfolgeknoten weitergegeben werden

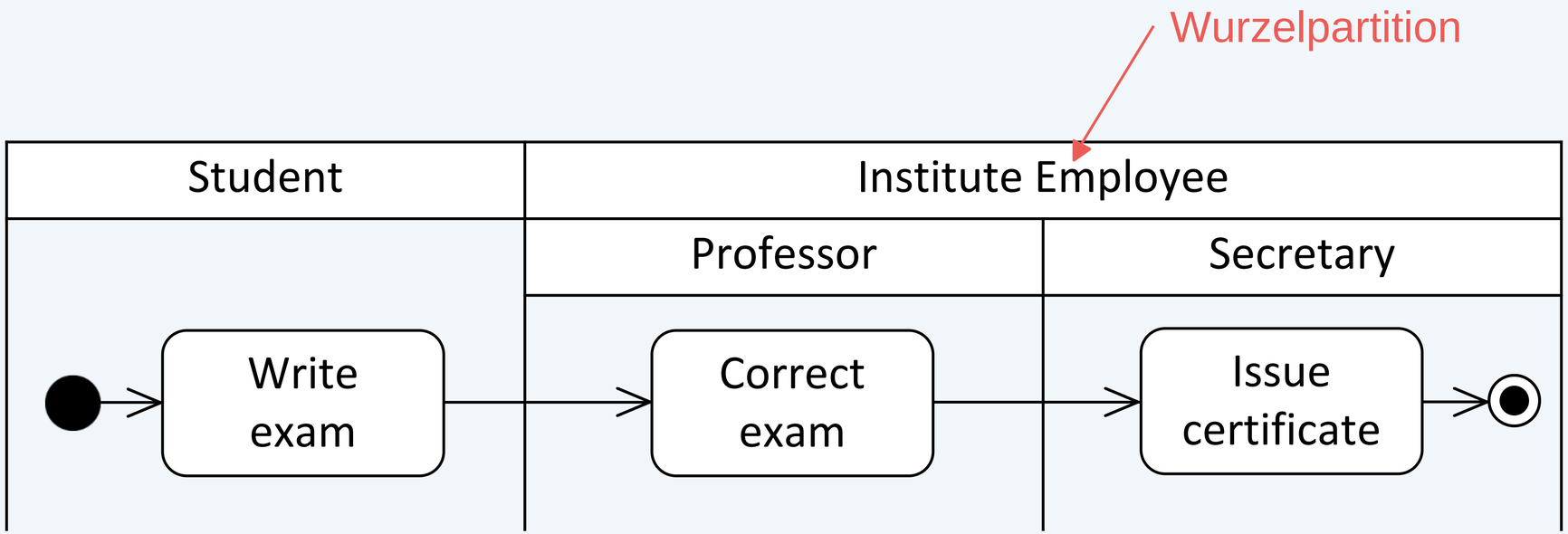


- **Beispiel**: Pufferknoten kann max. 20 ToDos aufnehmen.

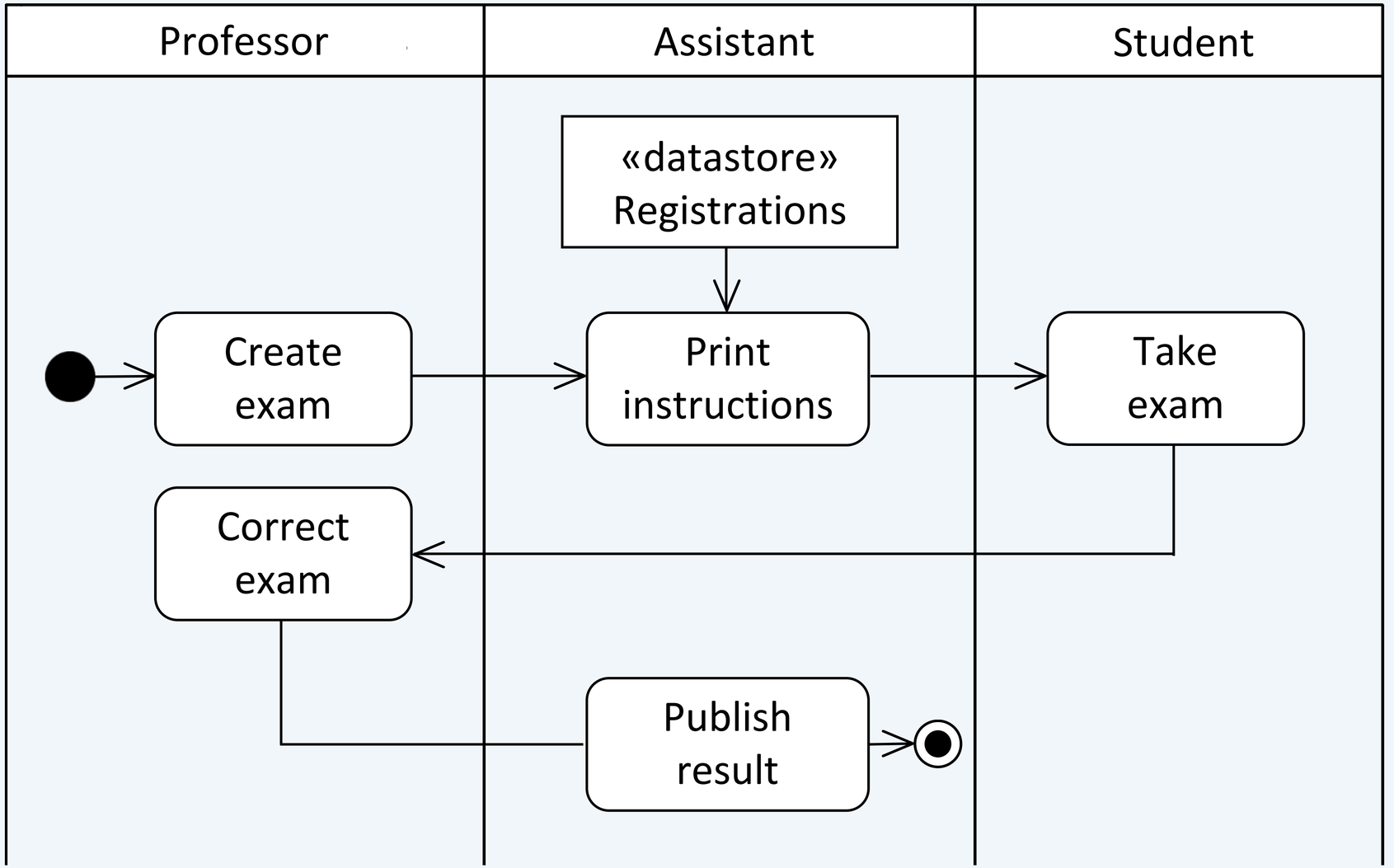


Partitionen

- Erlauben die **Gruppierung von Knoten und Kanten** einer Aktivität nach bestimmten Kriterien
- **Logische Sicht auf eine Aktivität** zur Erhöhung der Übersichtlichkeit und Semantik des Modells
- Hierarchische Partitionen
  - Zur Schachtelung auf verschiedenen Hierarchieebenen

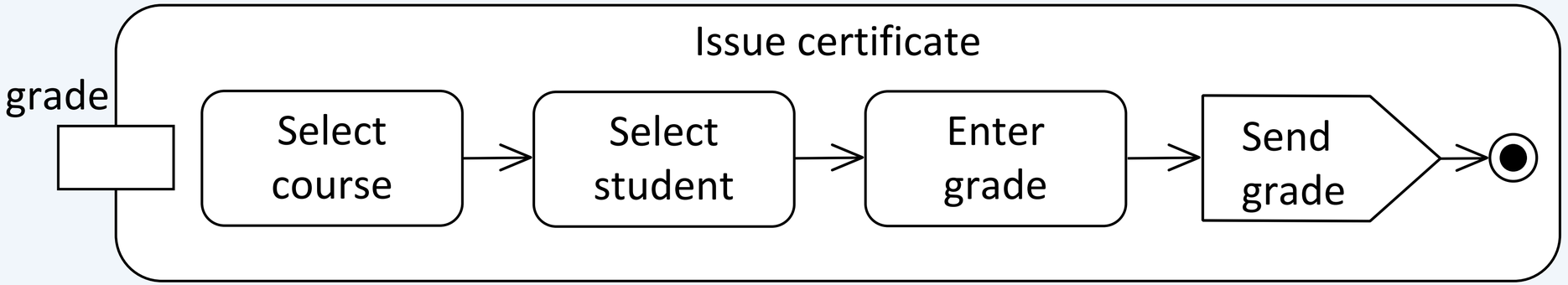


Partitionen – Bsp.:

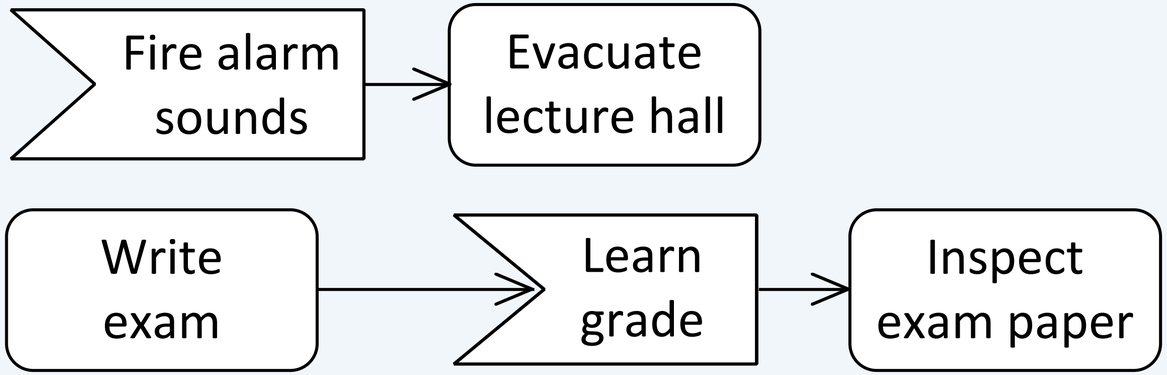


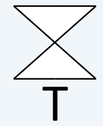
Ereignisbasierte Aktionen

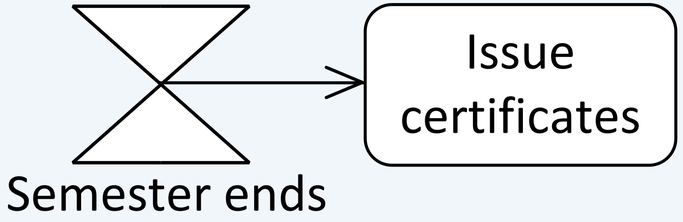
- Senden von Signalen 



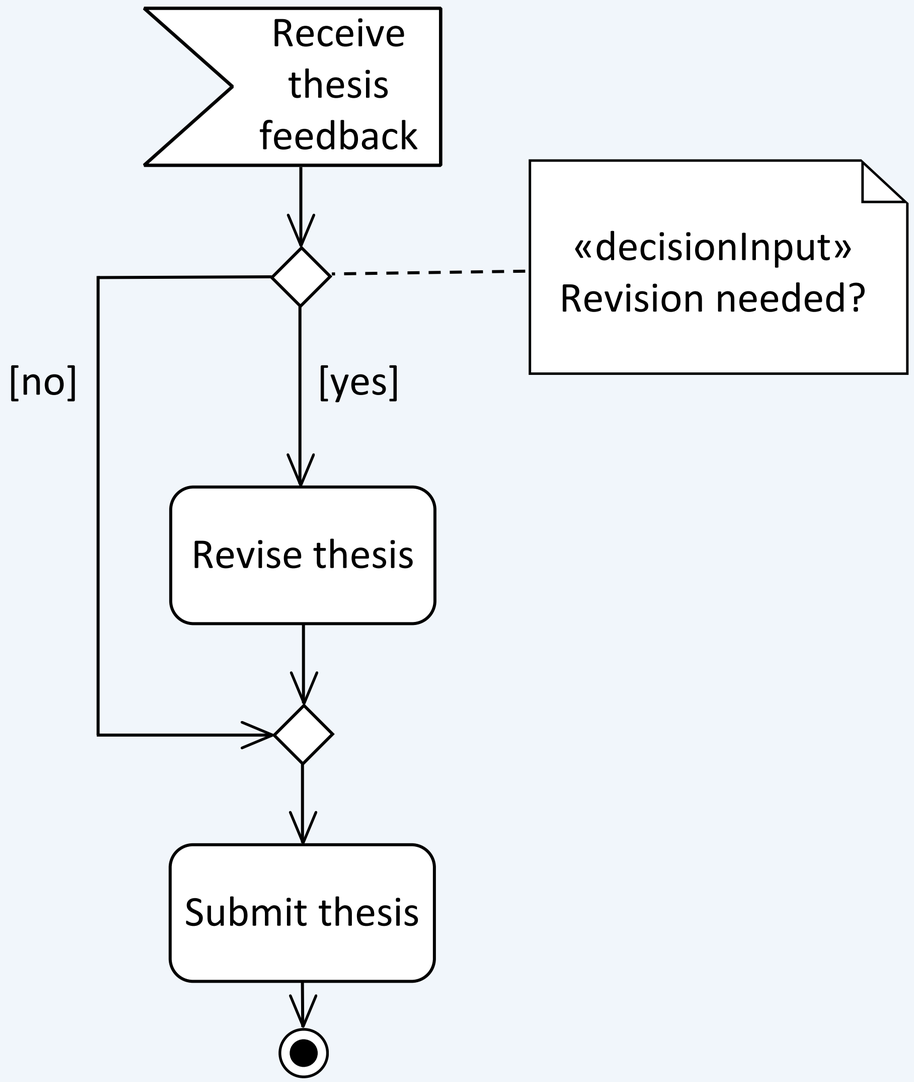
- Empfangen von Ereignissen
  - Asynchrones Ereignis 



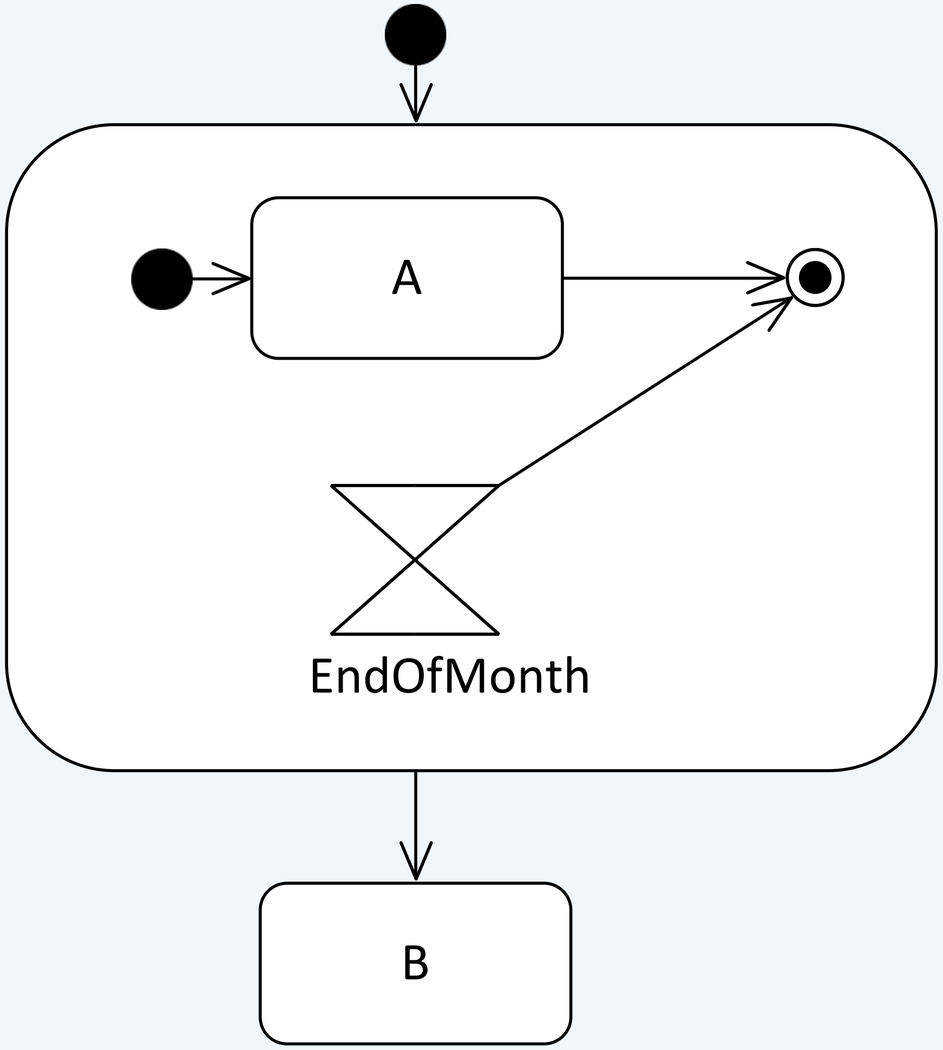
  - Asynchrones Zeitereignis 



Bsp.: Asynchrones Ereignis



Bsp.: Asynchrones Zeitereignis

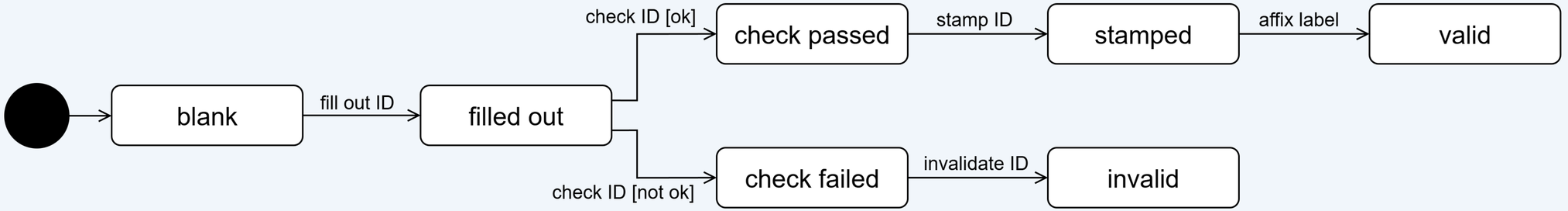


Aktivitätsdiagramm
Ein Beispiel

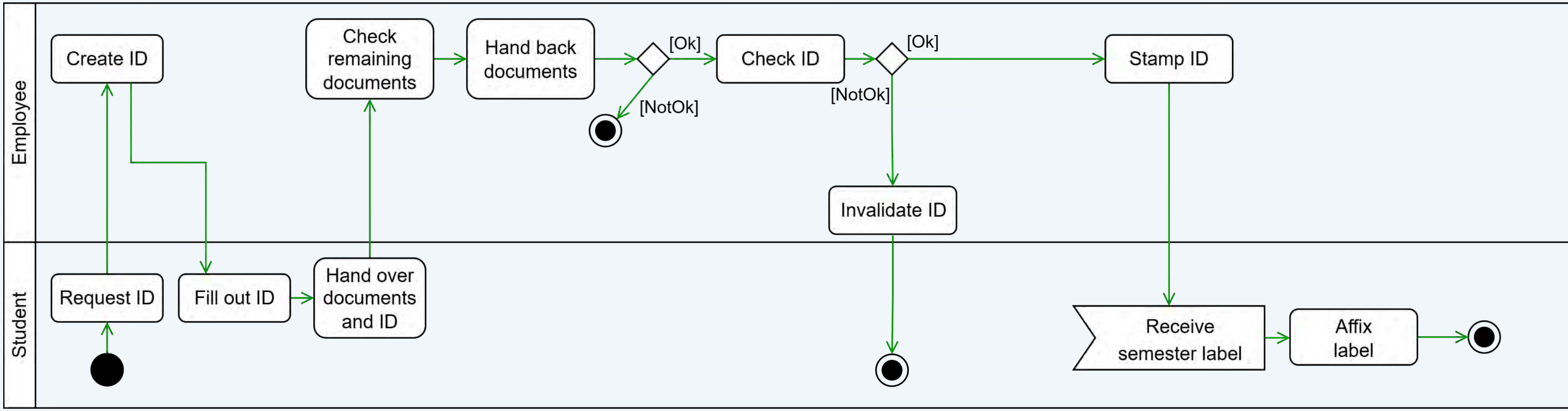
Christian Huemer und Marion Scholz

Studierendenausweis

- Zustandsdiagramm:

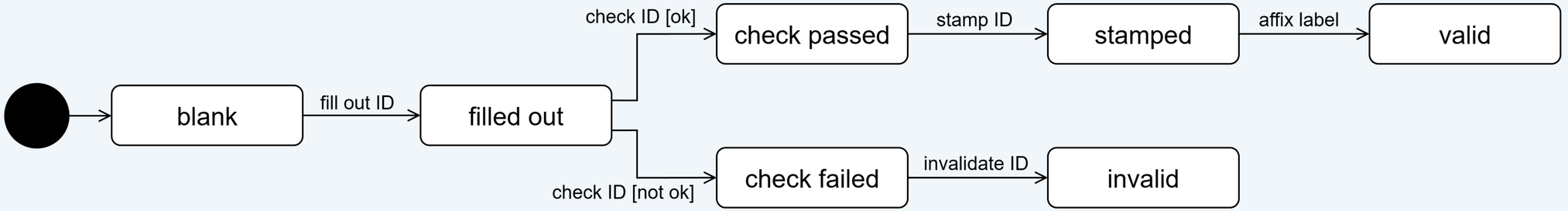


- Aktivitätsdiagramm:

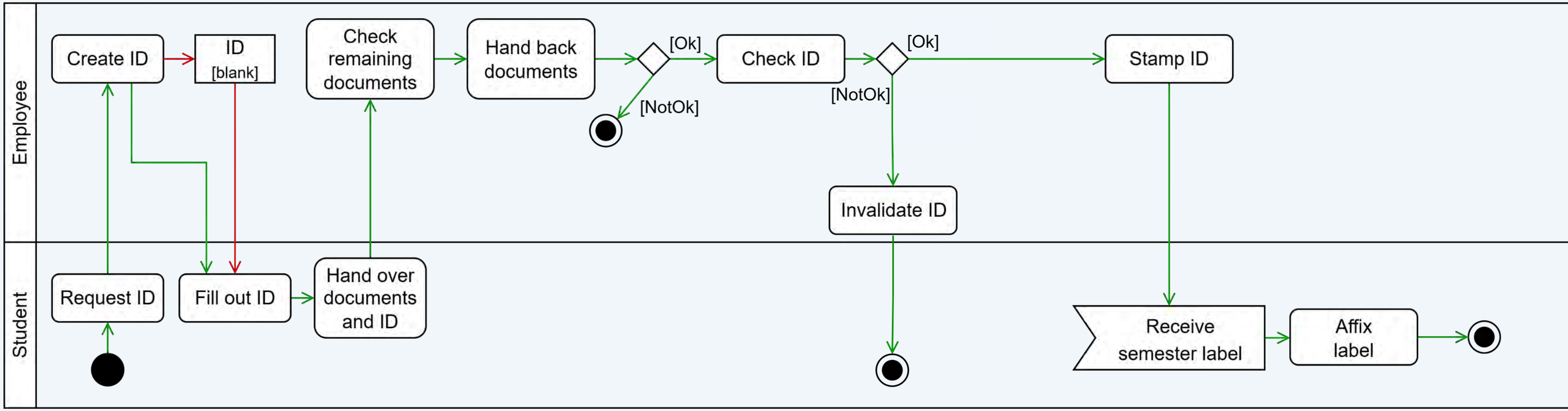


Studierendenausweis

- Zustandsdiagramm:

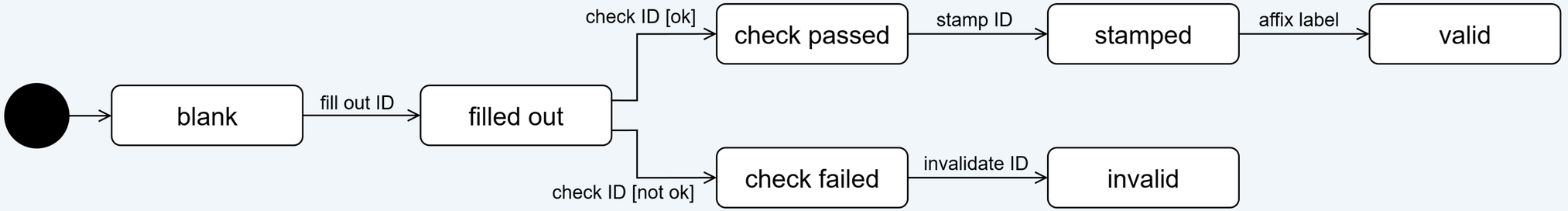


- Aktivitätsdiagramm:

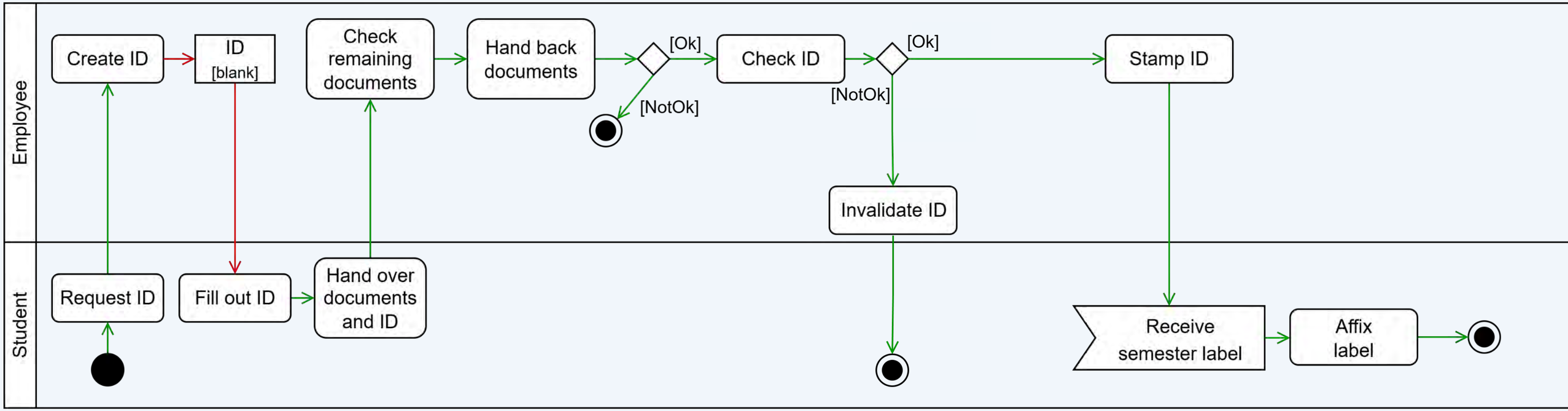


Studierendenausweis

- Zustandsdiagramm:

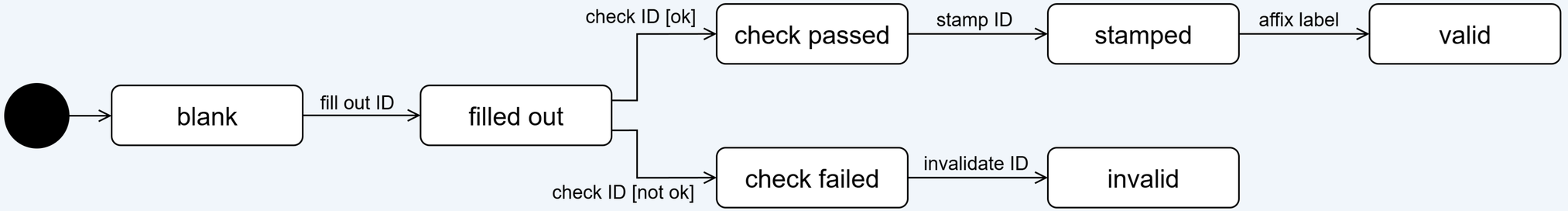


- Aktivitätsdiagramm:

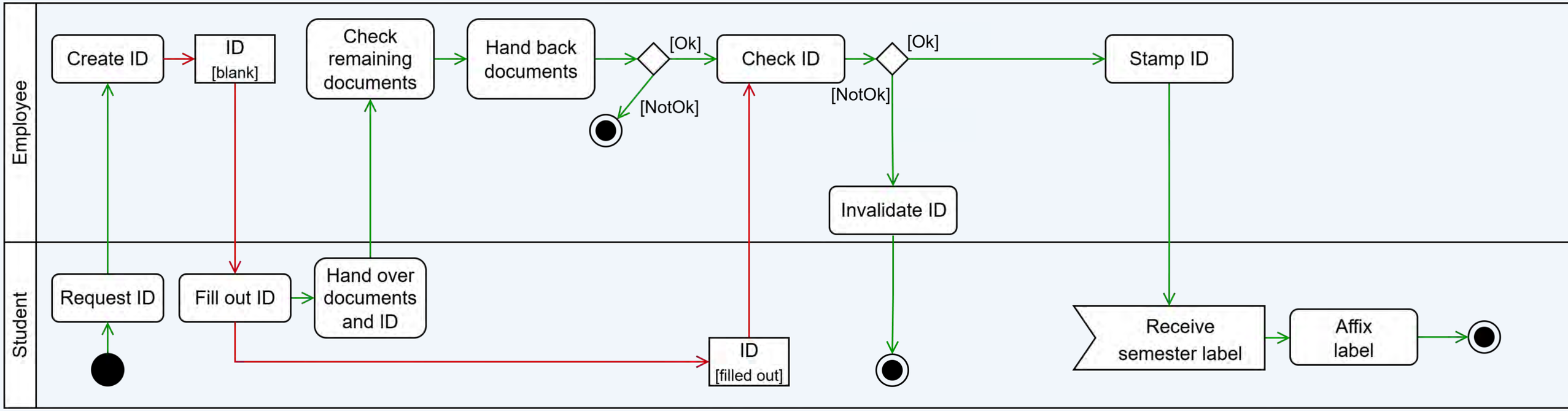


Studierendenausweis

- Zustandsdiagramm:

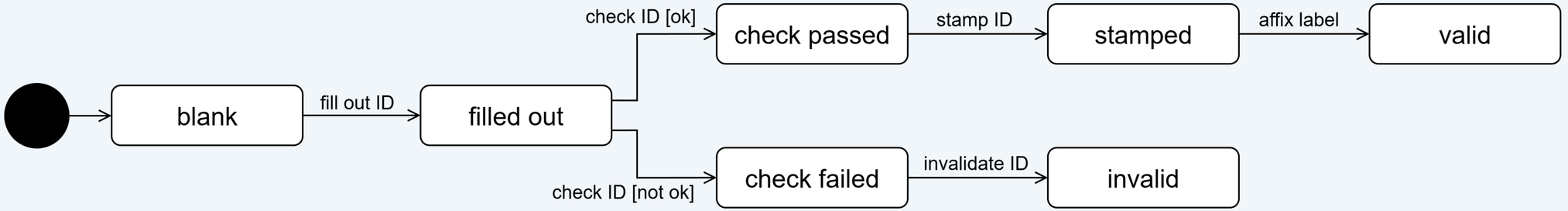


- Aktivitätsdiagramm:

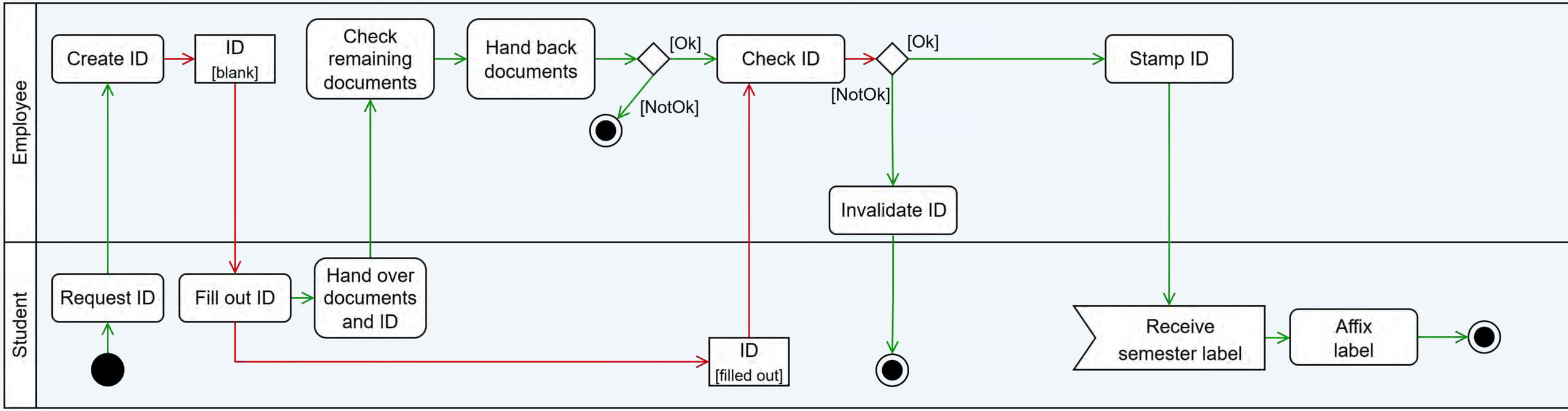


Studierendenausweis

- Zustandsdiagramm:

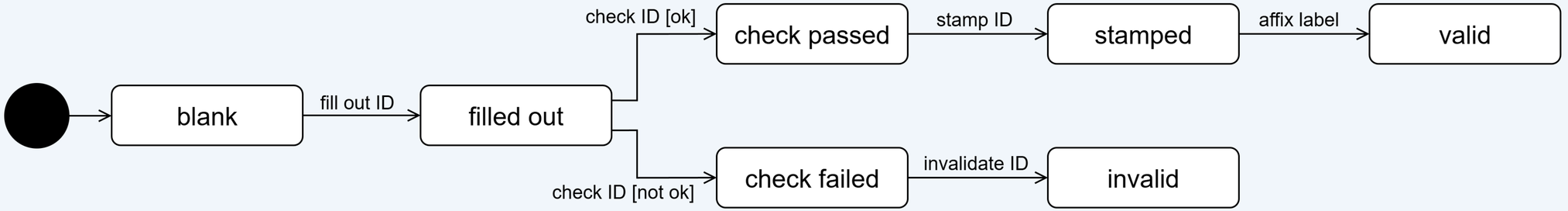


- Aktivitätsdiagramm:

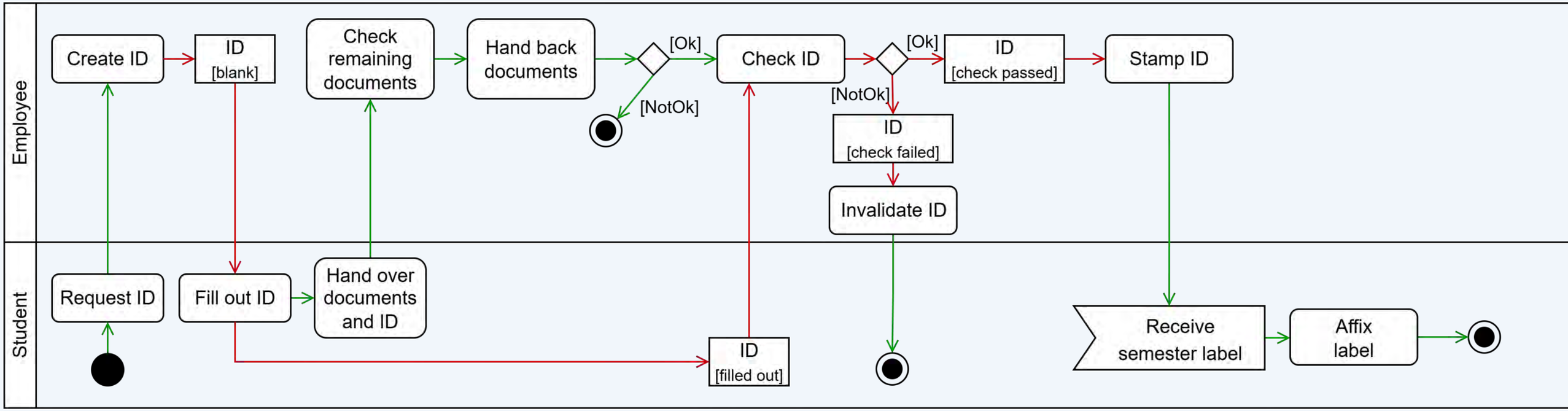


Studierendenausweis

- Zustandsdiagramm:

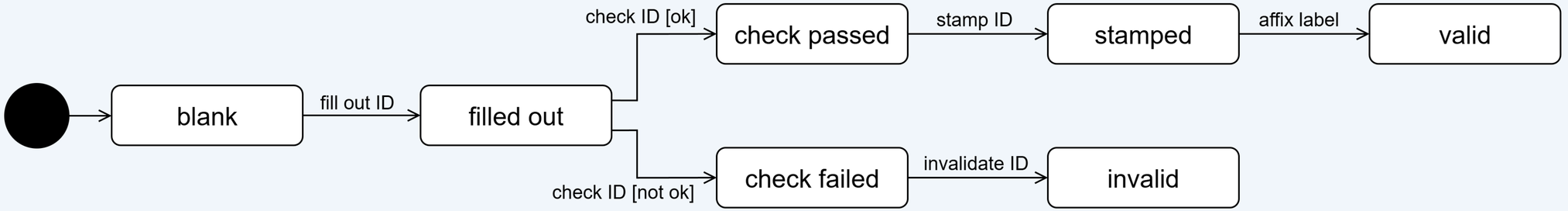


- Aktivitätsdiagramm:

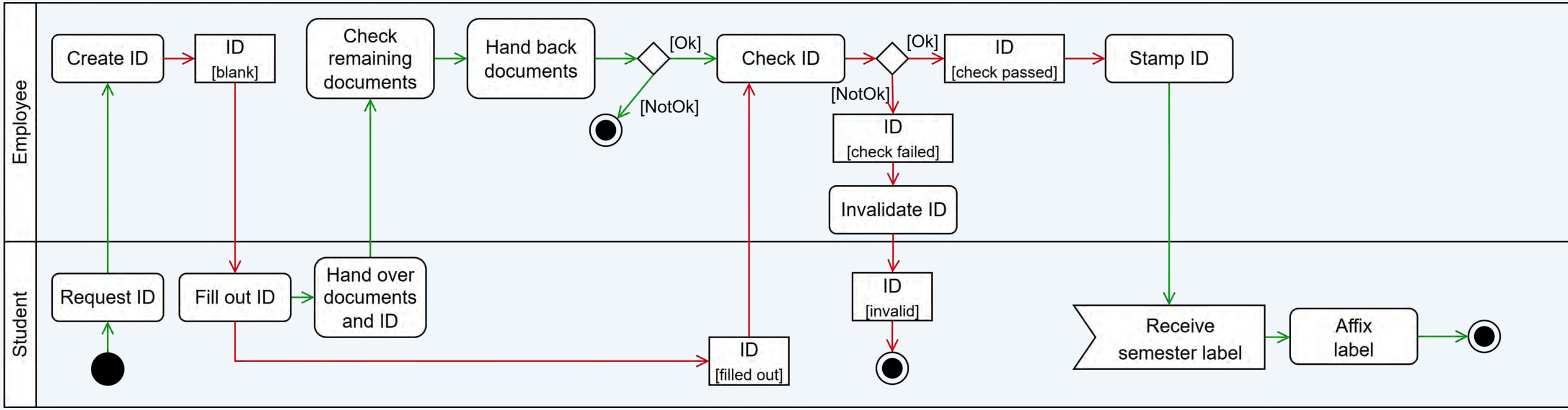


Studierendenausweis

- Zustandsdiagramm:

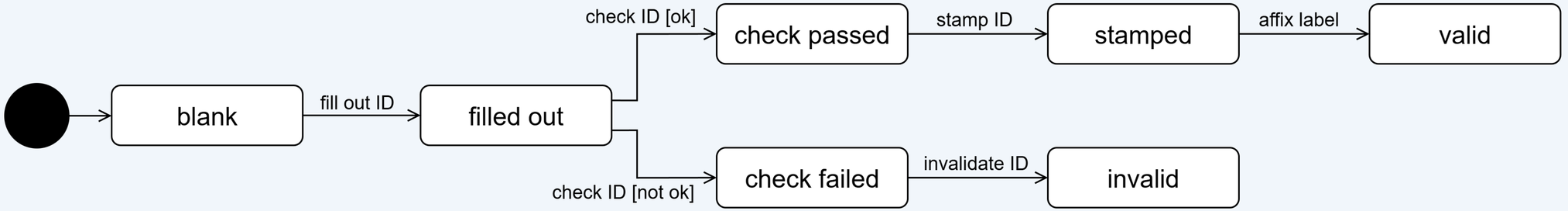


- Aktivitätsdiagramm:

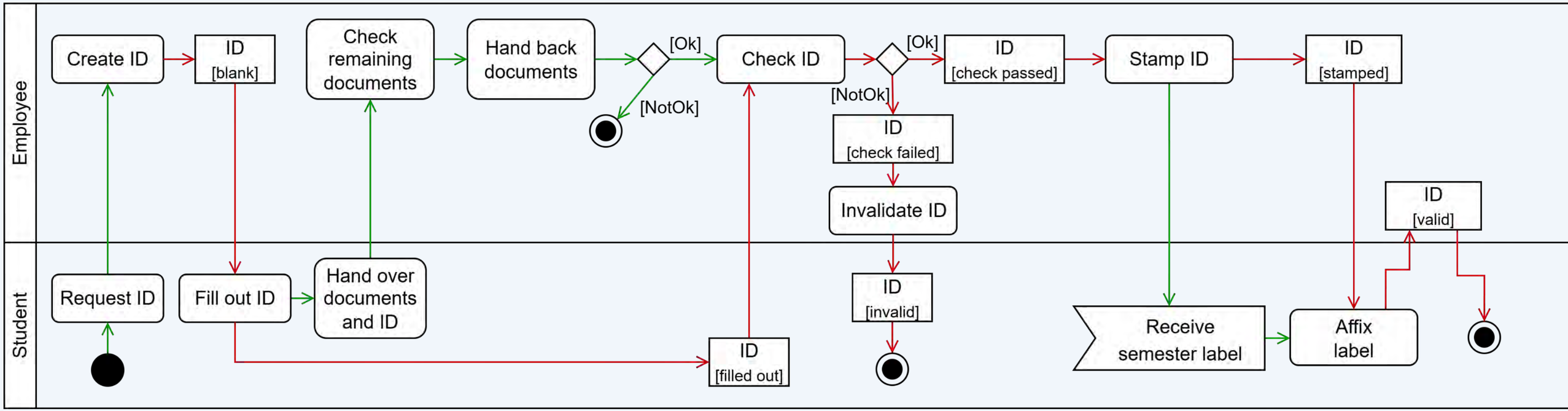


Studierendenausweis

- Zustandsdiagramm:



- Aktivitätsdiagramm:



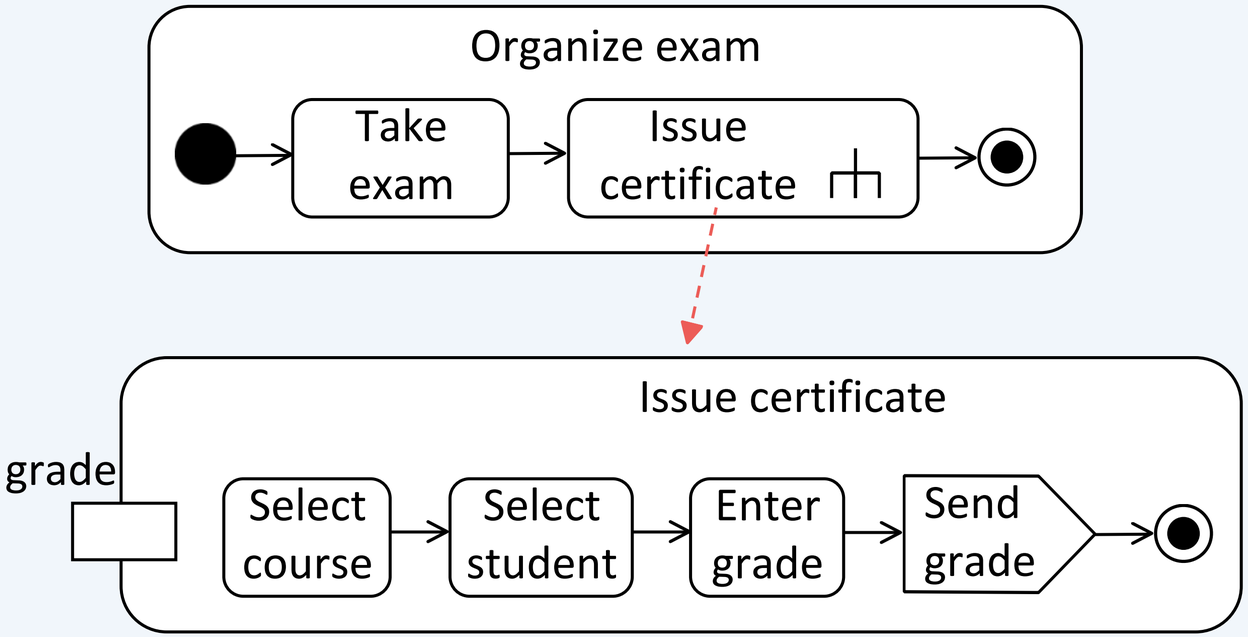
Aktivitätsdiagramm

Die Schachtelung von Aktivitäten und die Behandlung von Ausnahmen

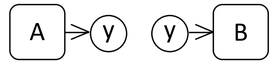
Christian Huemer und Marion Scholz

Aktivitätsaufruf

- Aktivitäten können wiederum Aktivitäten aufrufen
- Details werden an eine tiefere Ebene ausgelagert
- Vorteile:
  - Bessere Lesbarkeit
  - Wiederverwendung
- Notation:
  - In einer Aktion wird eine Aktivität aufgerufen



Konnektor

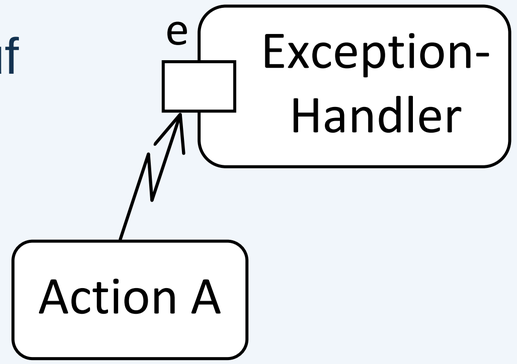


- Sinnvoll, wenn zwei verbundene Aktionen, im Diagramm weit auseinander liegen
- Ohne Konnektor:

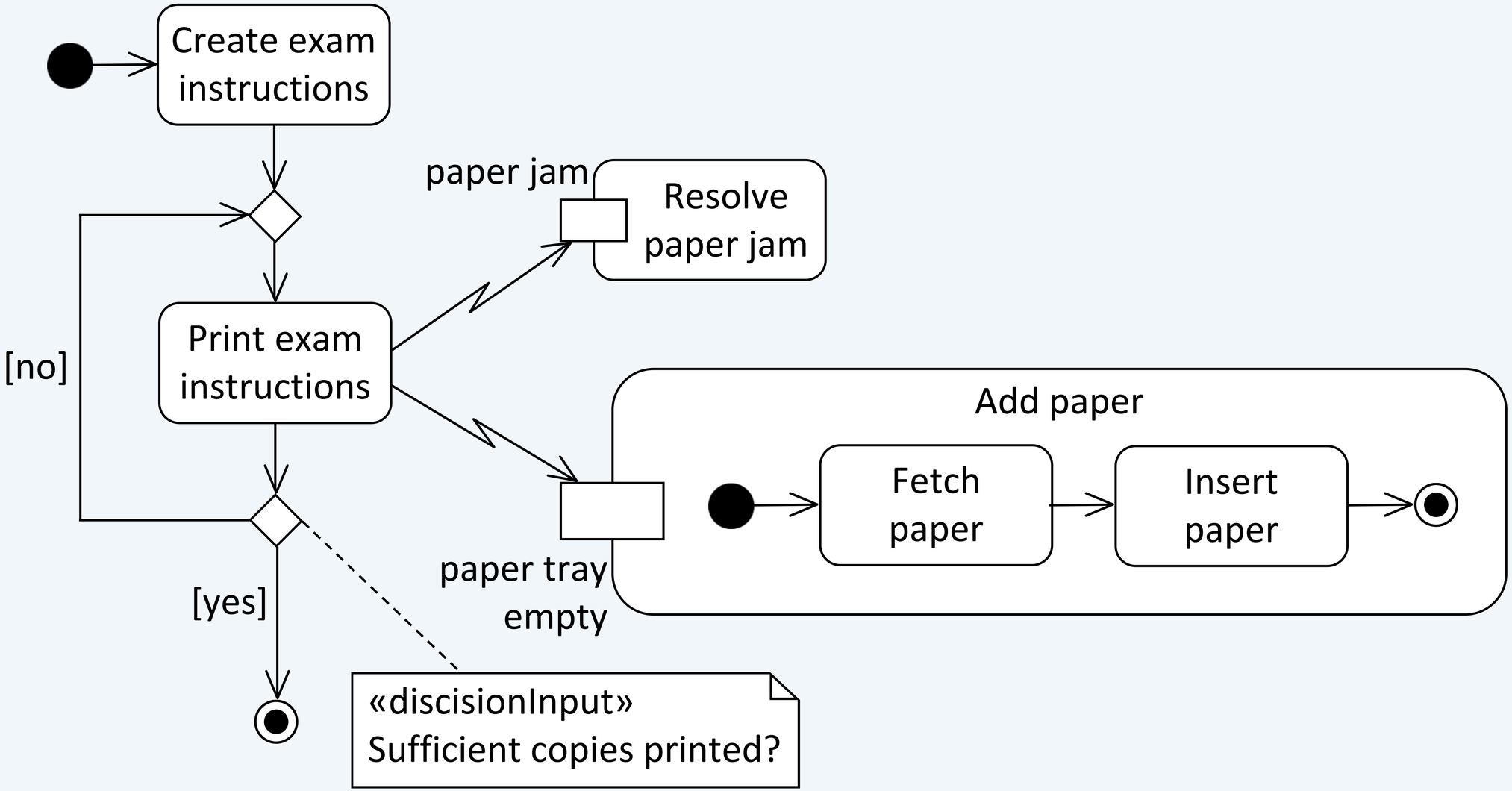
- Mit Konnektor:

Ausnahmebehandlung – Exception Handler (1/3)

- **Vordefinierte Ausnahmen** (z.B. Division durch 0)
- Definiert, wie das System auf einen bestimmten Fehler reagieren soll
- Der Ausnahmebehandlungsknoten **substituiert** den „geschützten“ Knoten und hat keine ausgehenden Kontroll- oder Objektflüsse
- Wenn Fehler **`e`** auftritt…
  - Alle Token in **`Action A`** werden gelöscht
  - Der **`Exception-Handler`** wird aktiviert
  - Der **`Exception-Handler`** wird statt **`Action A`** ausgeführt
  - Danach geht der Ablauf regulär weiter



Ausnahmebehandlung – Exception Handler (2/3) Bsp.

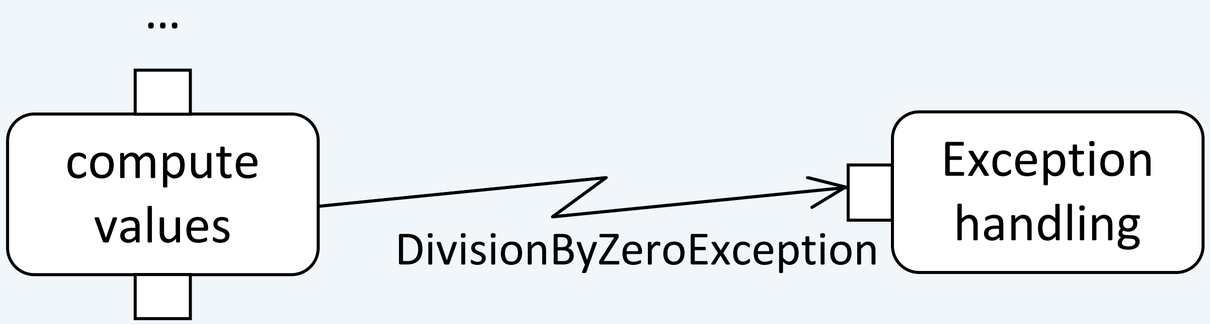


Ausnahmebehandlung – Exception Handler (3/3)

- Existiert für einen Ausnahmetyp keine Ausnahmebehandlung, wird die betroffene Aktion beendet und die Ausnahme nach außen propagiert

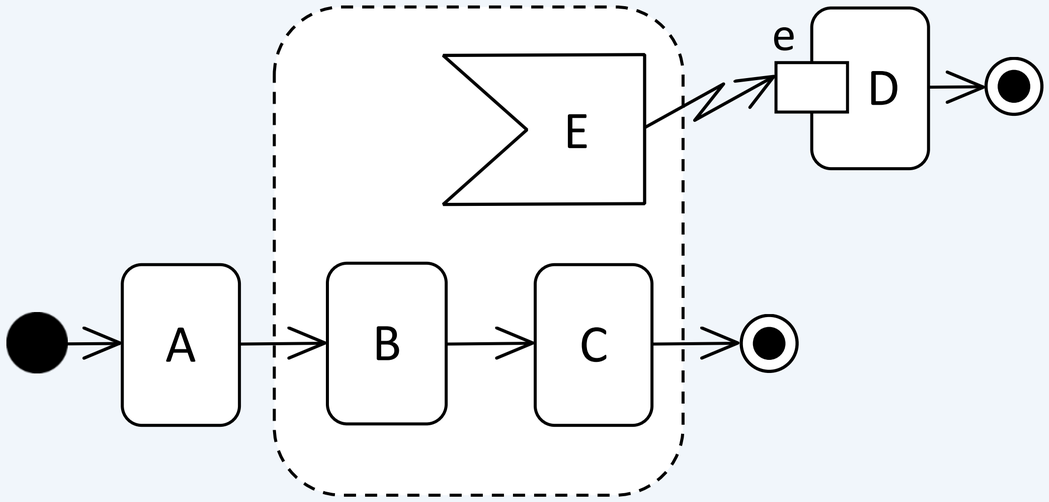
- Bsp.:

```
try {

    // compute values

} catch (DivisionByZeroException) {

    // exception handling

}
```



Ausnahmebehandlung – Unterbrechungsbereich (1/2)

- Umschließt 1-n Aktionen, deren Ausführung unmittelbar beendet wird, falls ein bestimmtes Ereignis eintritt
- Falls während der Ausführung von **B** oder **C** das Ereignis **E** eintritt
  - Ausnahmebehandlung wird aktiviert
  - Alle Kontrolltoken innerhalb des Unterbrechungs-bereichs (also in **B** und **C**) werden gelöscht
  - **D** wird aktiviert und ausgeführt



Ausnahmebehandlung – Unterbrechungsbereich (2/2)