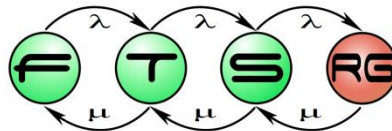


# Prozessmodellierung

**Budapesti Műszaki és Gazdaságtudományi Egyetem**  
**Hibatűrő Rendszerek Kutatócsoport**



# Inhalt

Wiederholung

Ziel der Prozessmodellierung

Prozessmodelle

Kontrollfluss

Verwirklichung

**Wiederholung**

**Ziel der  
Prozessmodellierung**

**Prozess-  
modelle**

**Kontroll-  
fluss**

**Verwirklichung**

# WIEDERHOLUNG

# Strukturelle und Verhaltensmodellierung

- Die Struktur (*structural*)
  - statisch
  - Teil und Ganzheit, Bestandteile
  - Verhältnisse, Verbindungen

Hauptteile des Roboterstaubsaugers sind das Steuerwerk, das Laufwerk und der Staubsauger.

- Verhalten (*behavioral*)

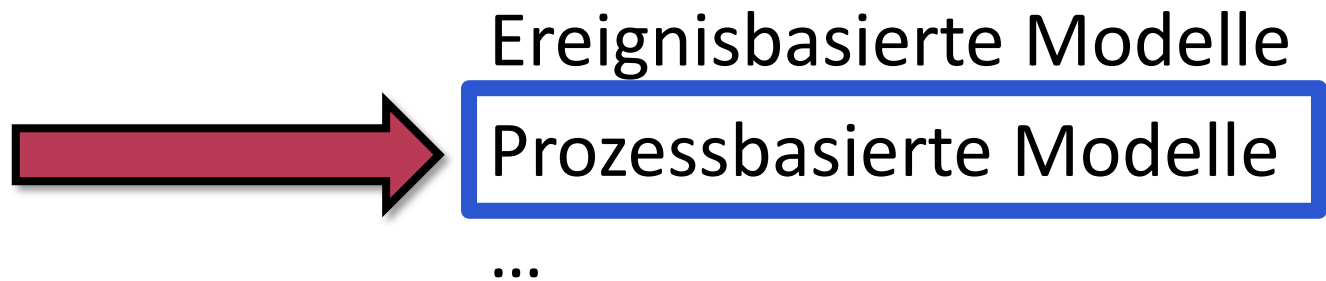
Auf dem Befehl „rechts“ wechselt das Laufwerk seine Betriebsart auf „Abbiegen“.

- dynamisch
- zeitlicher Verlauf
- Zustände, Prozesse
- Reaktionen auf die Außenwelt



# Hauptfragen der Verhaltensmodelle

- Was „macht“ das System?



- „Wie“ ist das System aktuell und wie verändert es sich?



# Hauptfragen der Verhaltensmodelle

- Zustandsbasierterer Ansatz
  - das System ändert sich
  - Eigenschaften des Systems werden geändert
  - als Reaktionen auf (externen) Ereignissen
  - Ein-/Ausgabekanäle
- Prozessbasierterer Ansatz
  - das System ändert das Arbeitsobjekt
  - Eigenschaften des Arbeitsobjekts werden geändert
  - als Reihe von Arbeitsschritten
  - Datenfluss

# Definition: Prozess

**Prozess:** Folge von Schritten (Aktivitäten), die in der gegebenen Reihenfolge ausgeführt zu irgendeinem Ziel führen.

Wiederholung

**Ziel der  
Prozessmodellierung**

Prozess-  
modelle

Kontroll-  
fluss

Verwirklichung

# ZIEL DER PROZESSMODELLIERUNG



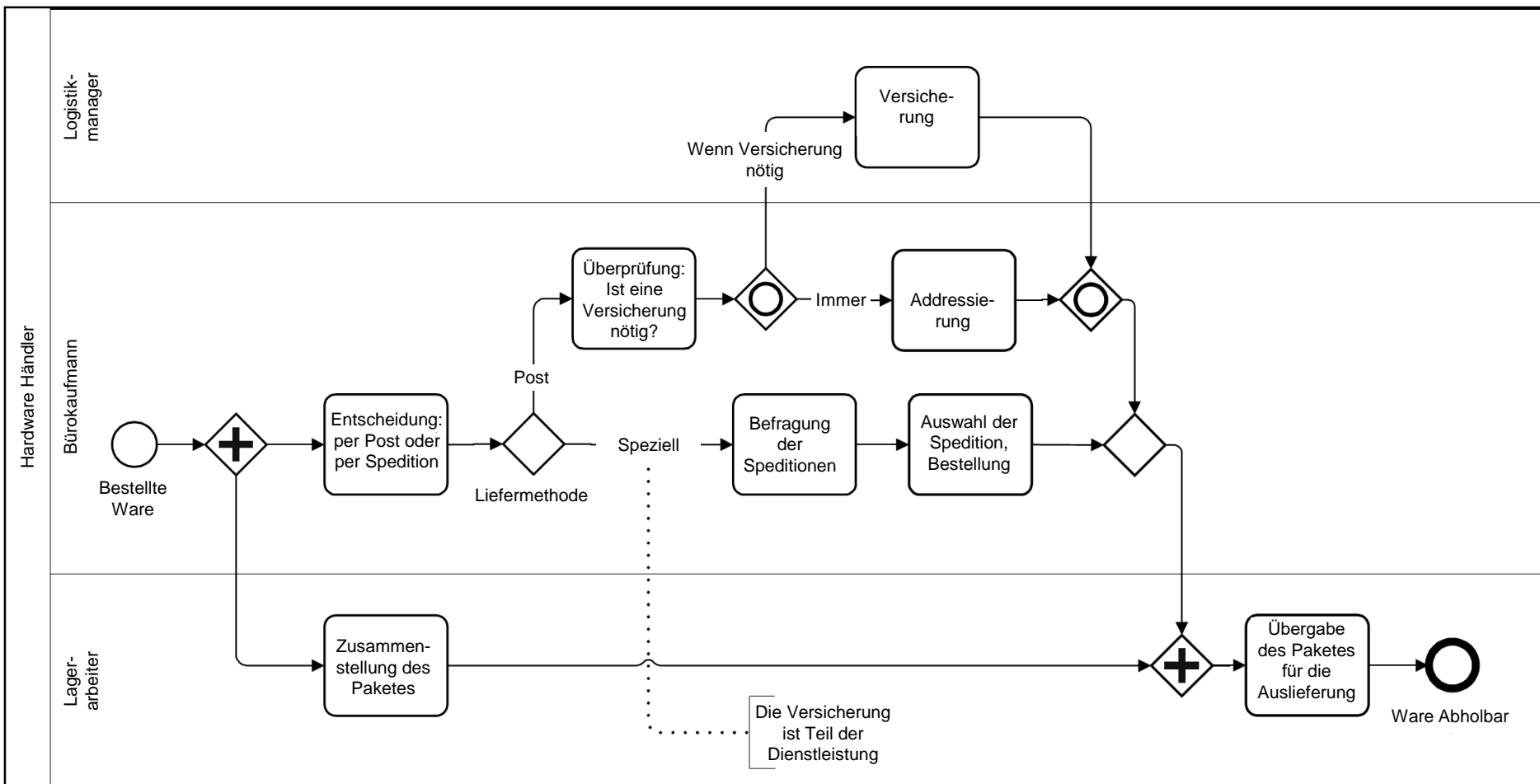
# Ziel der Prozessmodellierung

- Spezifikation
- Entwurf
- Implementation
  - Ausführbare Modelle
  - Codegenerierung
- Überprüfung auf Modellebene (Verifikation)
  - Simulation
  - Beobachtung (monitoring)
  - Automatisierte Modellprüfung
- Dokumentation

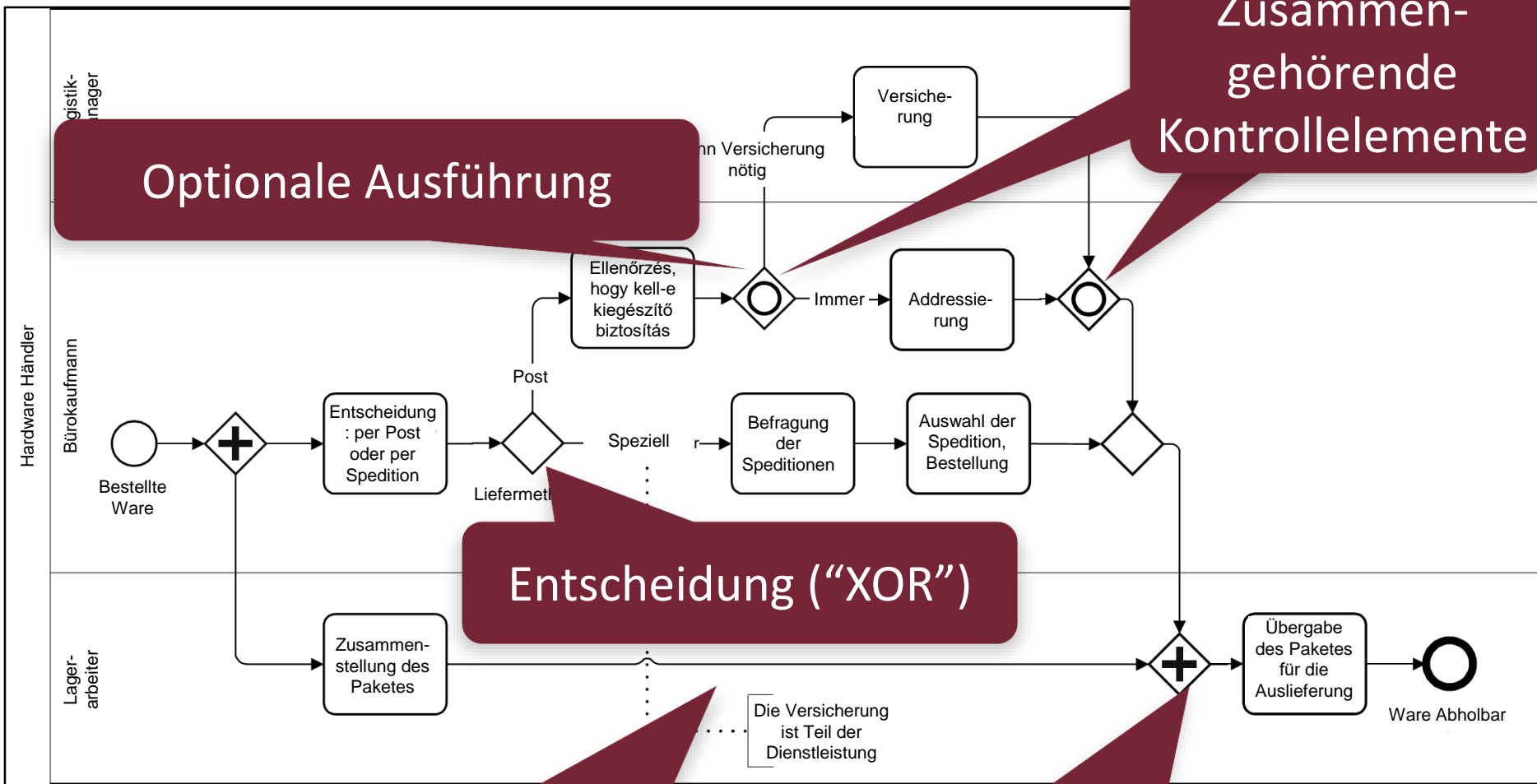
# Beispiel: Wie kommen die Waren an?

- Bestellung, Bezahlung
- Zusammenstellung des Paketes
  - Ware am Lager
  - Ware bei Vorlieferant
  - Ware wird erst gefertigt
- Benachrichtigung
- Auslieferung
  - Selbstabholung
  - Versand per Post/Paketdienst
  - Versand per Spedition

# Beispiel: Wie kommen die Waren an?



# Beispiel: Wie kommen die Waren an?



Optionale Ausführung

Zusammengehörende Kontrollelemente

Entscheidung ("XOR")

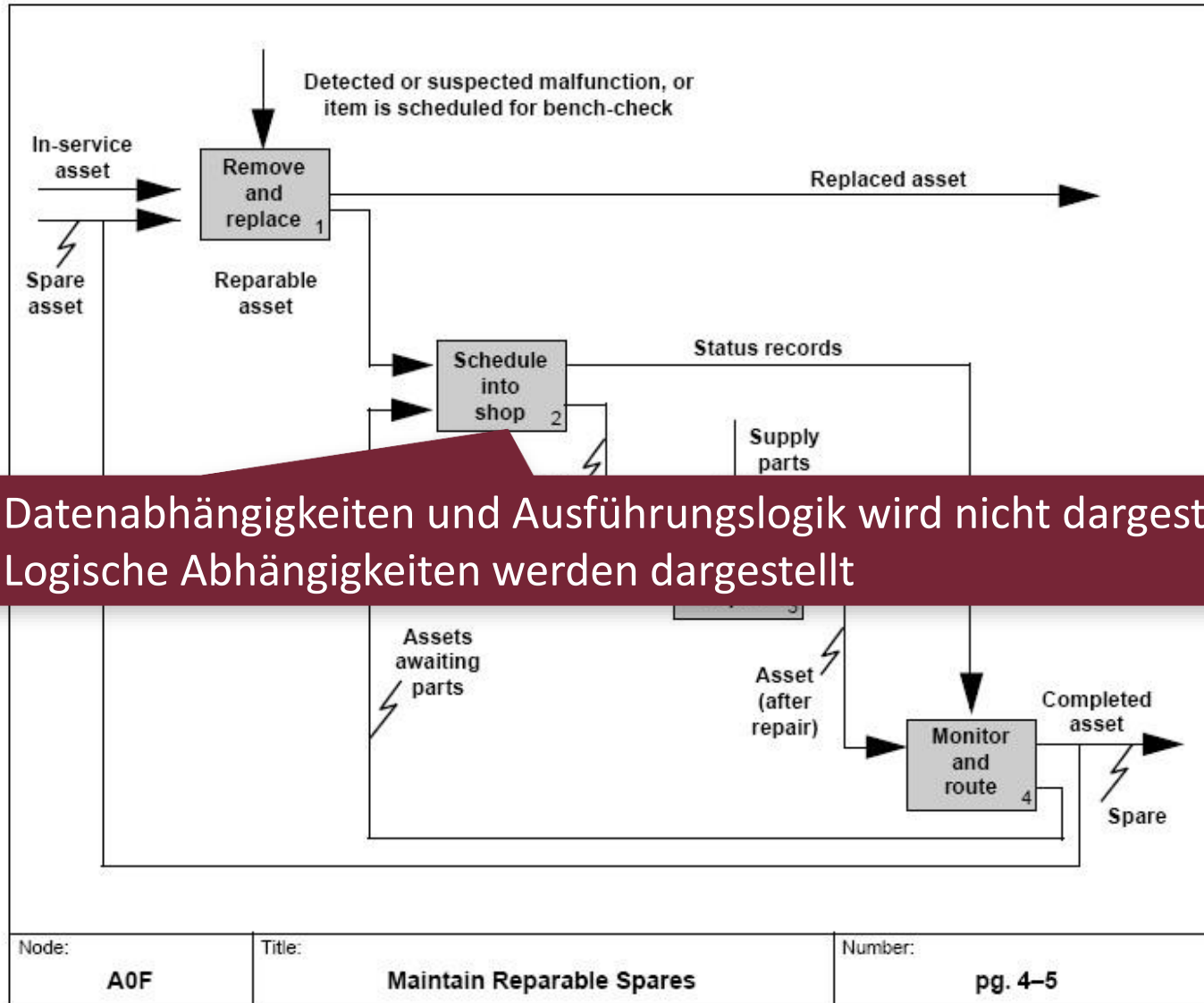
Reihenfolge der Schritte

"Parallele" (unabhängige) Ausführung ("AND")

# Grundsteine

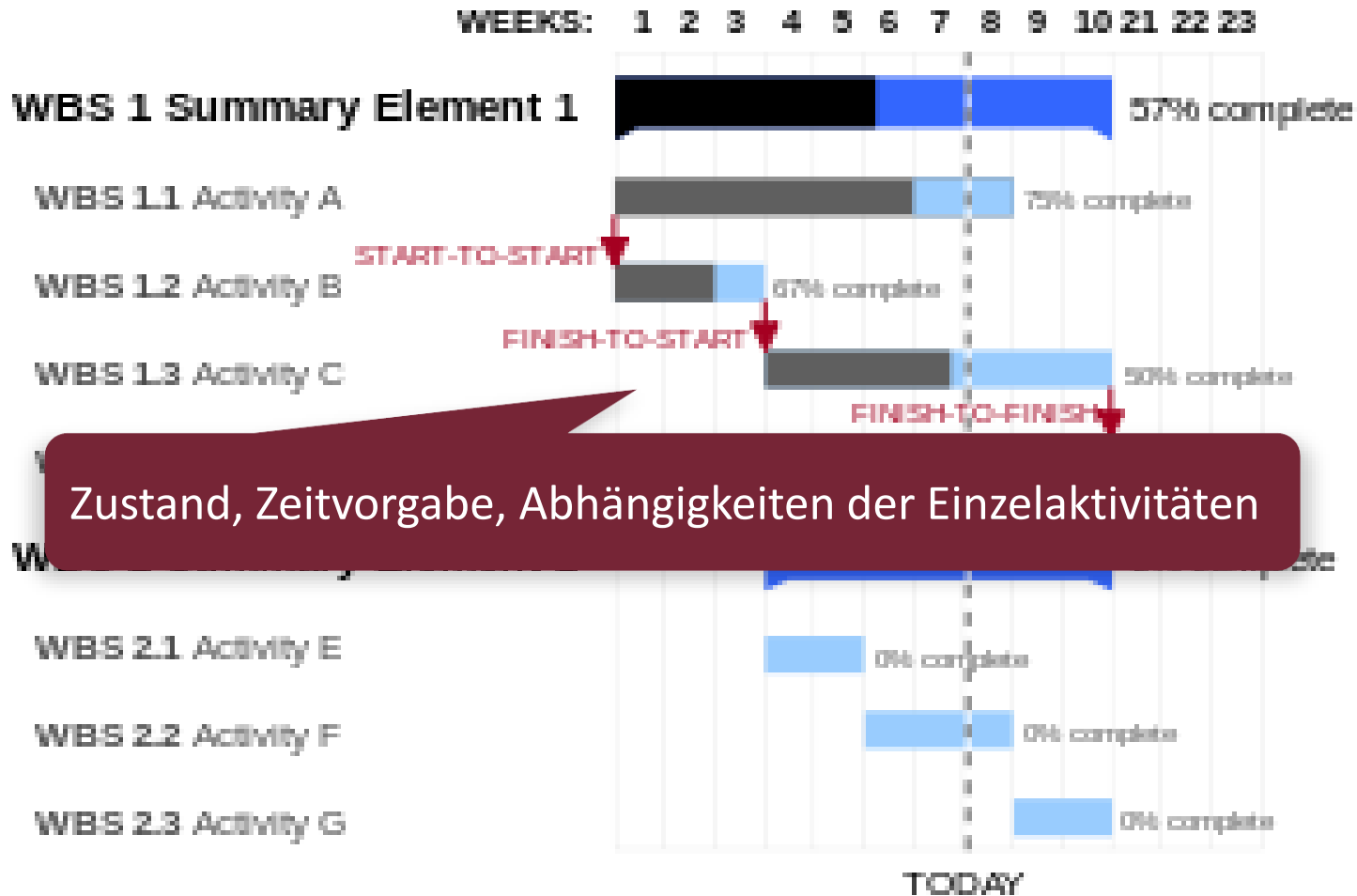
- Vorgängertechniken
  - Kontrollstrukturen der Programme
  - Zeitplanerstellung/Zeitablaufsteuerung (z.B. GANTT Diagramme)
  - Modellierung von Produktions-/Büroprozessen
  - IDEF-0: 1980-er Jahre, US AirForce
  - Beschreibung logistischer Prozesse
  - Betrieb komplexer Systeme: “runbook”
- Gemeinsame Elemente
  - Es gibt elementare Aktivitäten (Schritte)
  - Abhängigkeiten (Zeit? Daten? Reihenfolge?)
  - Entscheidungspunkte
  - → Universelle Prozessmodellierungssprachen (z.B. BPMN)

# Beispiel: IDEF-0



Datenabhängigkeiten und Ausführungslogik wird nicht dargestellt  
Logische Abhängigkeiten werden dargestellt

# Beispiel: GANTT



Zustand, Zeitvorgabe, Abhängigkeiten der Einzelaktivitäten

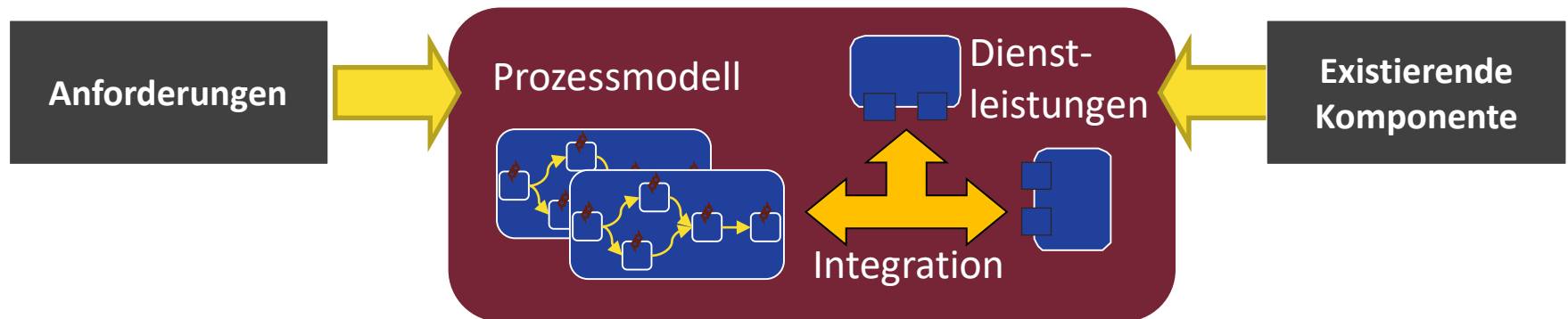
# Wiederverwendung

- Idee aus dem System-/Softwareentwurf:
  - Wiederverwendung bereits existierender Elemente
  - Beschreibung der Zusammensetzung des Systems
- Vielfalt der Bauelemente
  - Validierung von Webformulare, Email-Versand, Datenbankoperationen, Anruf von Web Services, menschliche Interaktion, SMS-Versand, Darstellung von Diagrammen, usw.



# Verwendung der Beschriebenen Kontrolllogik

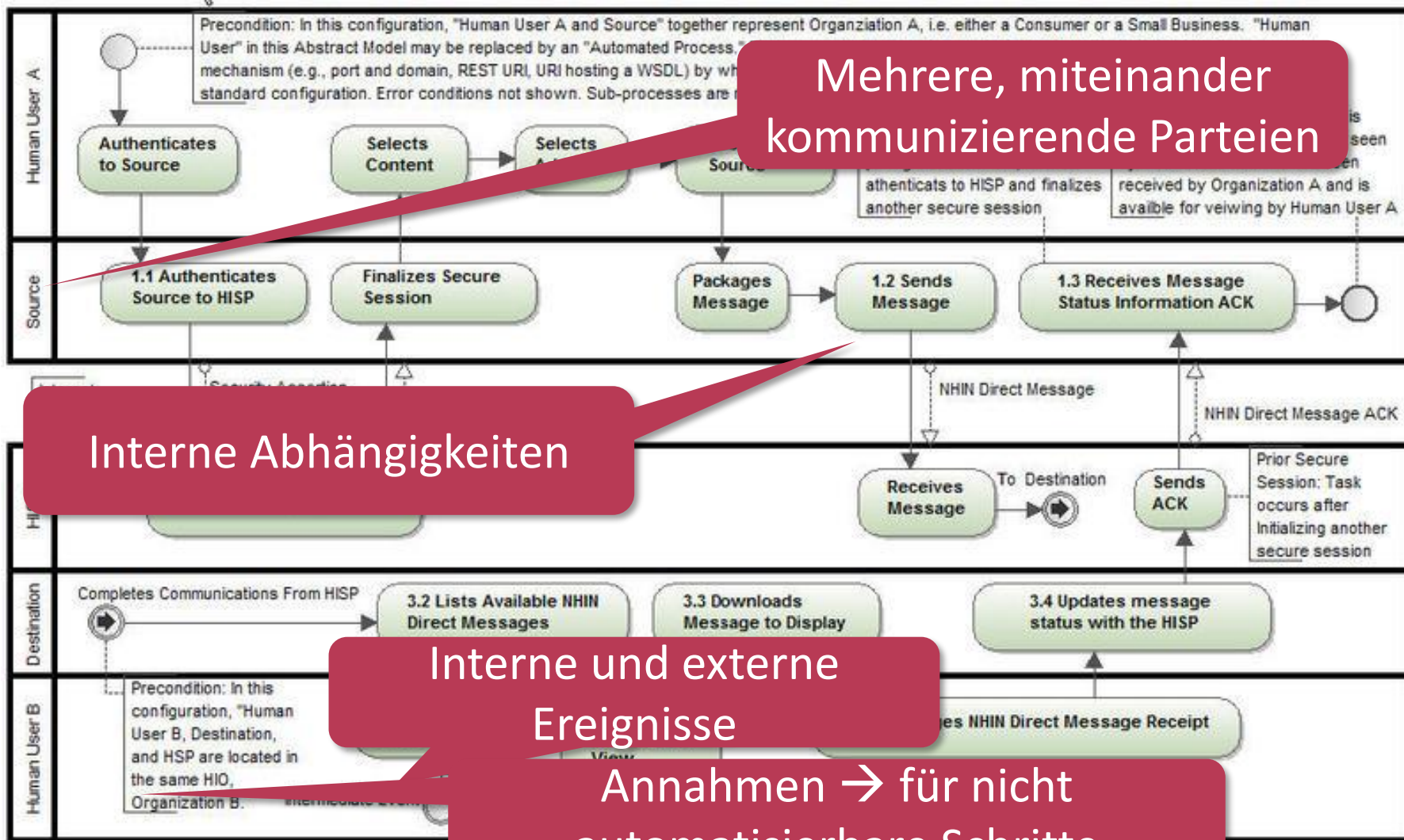
- Kodegenerierung (C/C++, C#, Java, ...)
- Eingabe für Ausführungsumgebungen
  - “Mach mir solche Prozesse”



# Wo werden Prozessmodelle verwendet?

- Betrieb der Informationssysteme
  - ITIL, COBIT
- Spezifikation der Protokolle
  - Zusammenarbeit innerhalb komplexer Systeme
  - Rollen der einzelnen Komponente
- Entwurf ausführbarer Prozesse
  - Auswertung von Bestellungen, Vorbereitung der Beurteilung von Kreditanforderungen
- Prozesse für Datenbearbeitung/-Analyse

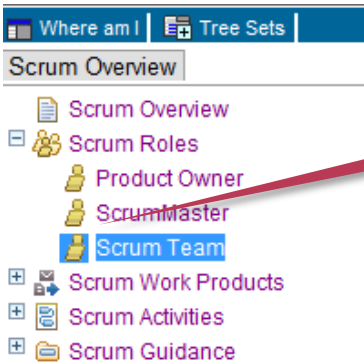
# Beispiel: Bearbeitung Gesundheitsdaten



<http://wiki.directproject.org/Abstract+Model+Examples>

# Beispiel: agile SW-Entwicklung als Prozess

Rollen, Produkte



Scrum Roles > Scrum Team

Role: Scrum Team



The Scrum Team builds the product that the customer is going to consume: the software or website, for example. The team in Scrum is "cross-functional" - it includes all the expertise necessary to deliver the potentially shippable product each Sprint - and it is "self-organizing", with a very high degree of autonomy and accountability.

Role Sets: Scrum Roles

Schritte der Team-Arbeit

Relationships



<http://www.eclipse.org/epf/>

# Weitere Beispiele

- Modellierung Bankprozesse
  - Welche Aktivitäten müssen bei Tages- und Wochenschluss ausgeführt werden?
  - Könnte die Bank täglich mehrmals Überweisungen ausführen?
- Modellierung Fertigungsprozesse
  - Optimale Produktionsablaufplanung: Wechseln oder Weiterproduzieren?
  - Wie wird produziert? (Technologie)
  - (siehe Vorlesung über Simulation)
- Modellierung Geschäftsprozesse
  - Gibt es sich wiederholende Kommunikationsmuster?
  - Modellbasierte Datenbearbeitung

# Beispiel: Datenbearbeitungsprozess

Schritte: Einlesen, Filtern, Diagrammgenerierung, ...



Visualisierung des Zustandes der Schritte für die einzelnen Abläufe: ist das Ergebnis schon produziert?

Werkzeug: z.B. KNIME

# Grundbegriffe der Prozessmodellierung

- Prozessmodellierungssprache
  - BPMN, jPDL, XPDL, BPEL, UML AD, ...
  - Kontrollfluss, Datenfluss
  - Datenstrukturen für Eingabe, Ausgabe und intern
  - Definition der zu ausführenden Schritte
  - Zeitplanung, Ressourcen, Rohstoffe
- Prozessvorlage (template)
  - Z.B. “Flugticketbestellung”-Prozess
  - Versionen, ...
- Prozessablauf (instance)
  - „Prof. Pataricza bestellt ein Flugticket nach Lissabon”

Wiederholung

Ziel der  
Prozessmodellierung

Prozess-  
modelle

Kontroll-  
fluss

Verwirklichung

# PROZESSMODELLE



# Elementare Aktivitäten

Compile

Anfang der Ausführung

Ende der Ausführung

*Compile*

t

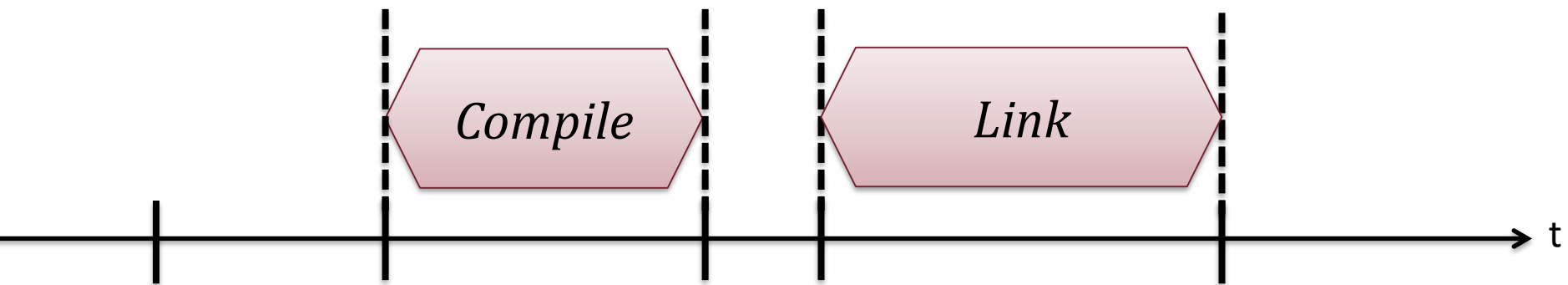
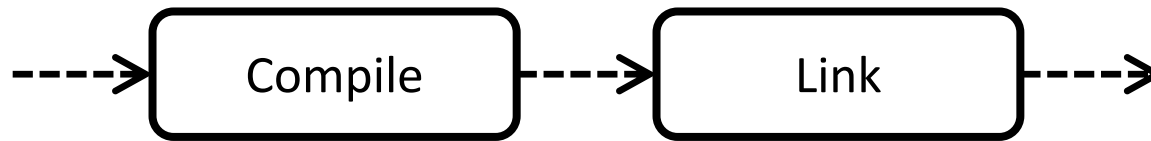
# Definition: Elementare Aktivität

Eine **elementare Aktivität** ist

- eine Tätigkeit, die eine zeitliche Ausdehnung hat,
- und die über ihren Anfang und Ende nicht mehr detailliert modelliert wird.

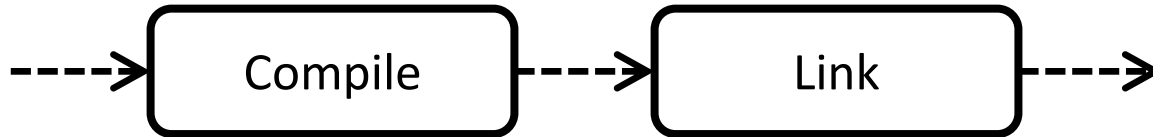
Compile

# Sequenz, Kontrollfluss

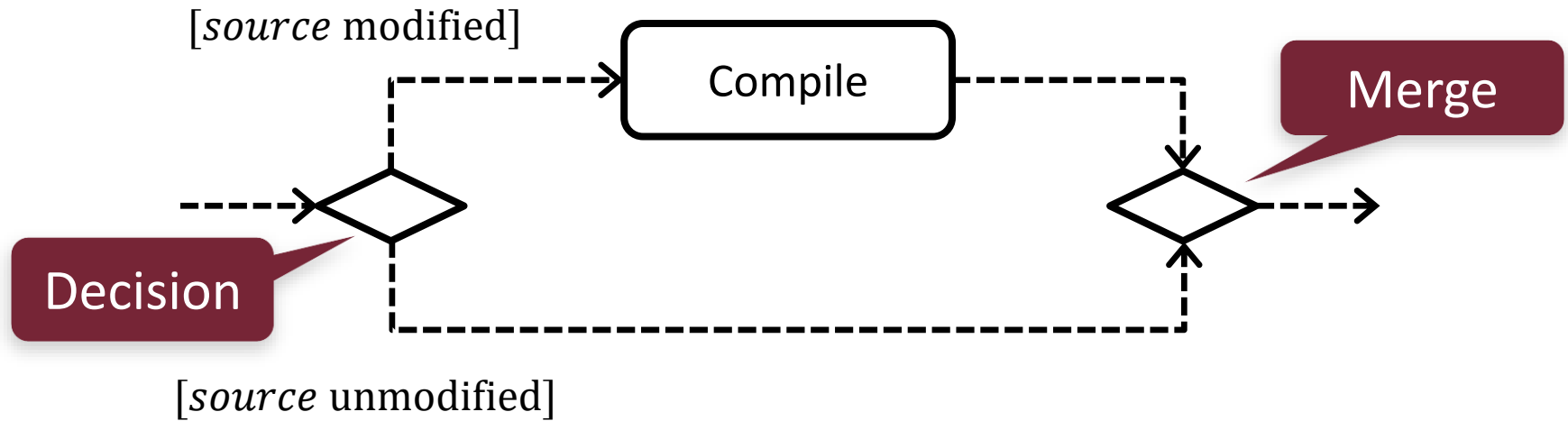


# Definition: Sequenz

Eine **Sequenz** definiert die Ausführungsreihenfolge der Tätigkeiten/Aktivitäten/Schritte.



# Wächterkonditionen, Verzweigung

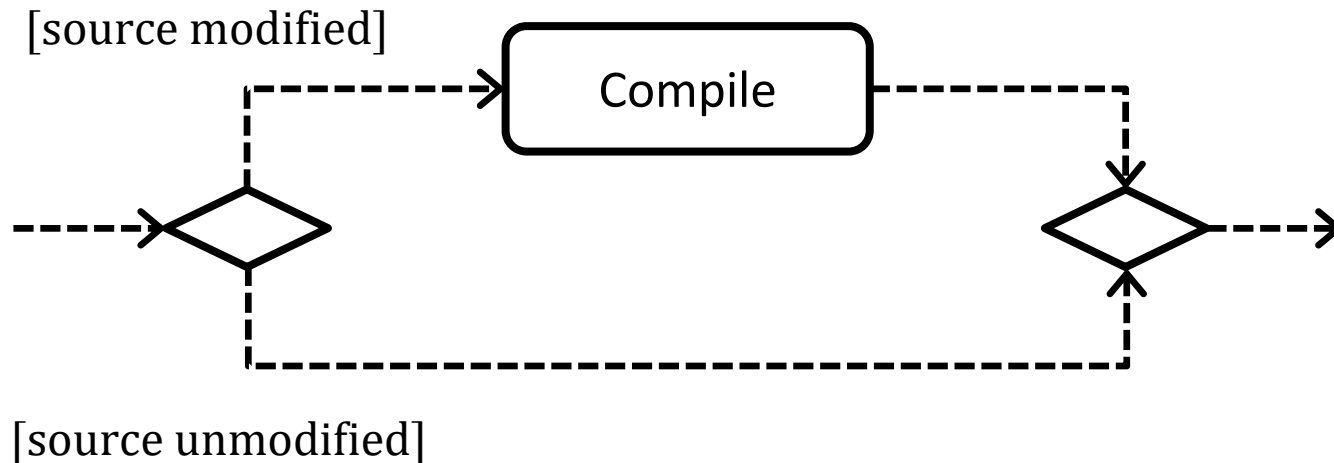


## ■ Semantik

- Nur der eine Zweig wird ausgeführt
- Nichtdeterminismus ist möglich
  - sich überlappende Wächterkriterien
  - ohne Wächterkriterien (nicht bekannt oder vernachlässigt)

# Definition: Kontrollelement

Ein **Kontrollelement** ist ein Knoten des Prozesses, der eine oder mehrere Aktivitäten des Prozessmodelles für Ausführung auswählt.



# Definition: Verzweigung

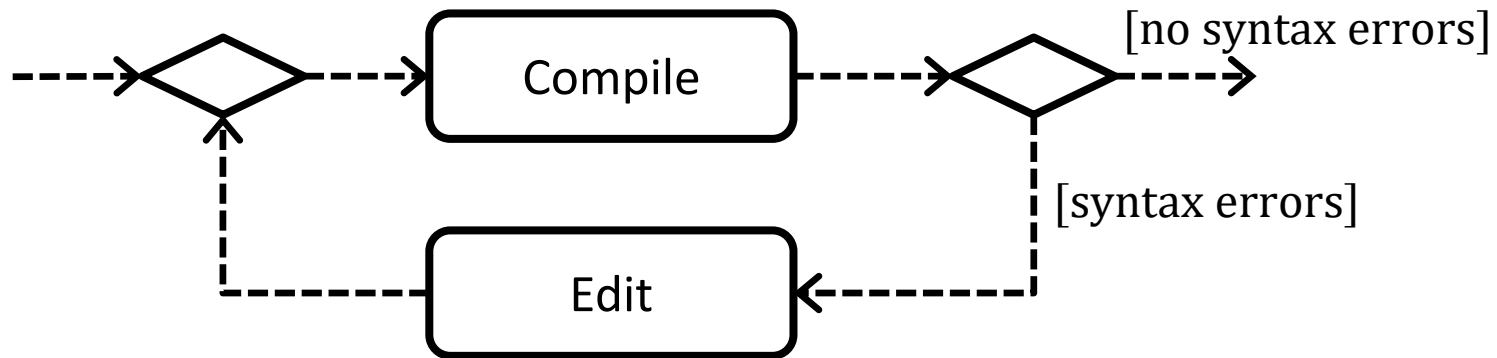
Eine **Verzweigung** ist eine Kontrollstruktur, die

- aus einem „**Entscheidung (Decision)**“ und einem „**Vereinigung (Merge)**“ Kontrollelement besteht, bei denen
- die Entscheidung mindestens zwei **Ausgänge** hat, von denen nach Auswertung der **Wächterkriterien** der Ausgänge gewählt wird, (das Kontrolltoken wird auf den entsprechenden Zweig gelegt),
- der ausgewählte Zweig (Ausgang) darf eine beliebige Anzahl von Elementen enthalten,
- alle Zweige führen zu derselben Vereinigung.

- Die Vorlesung benutzt ausschliessende Entscheidung (XOR), eine Auswertung darf immer nur einen Zweig auswählen.
- Eine Verzweigung kann binär oder mehrfach sein, die Vorlesung benutzt binäre Verzweigungen (mit genau zwei Ausgängen).

# Schleife

- Mehrfache Ausführung



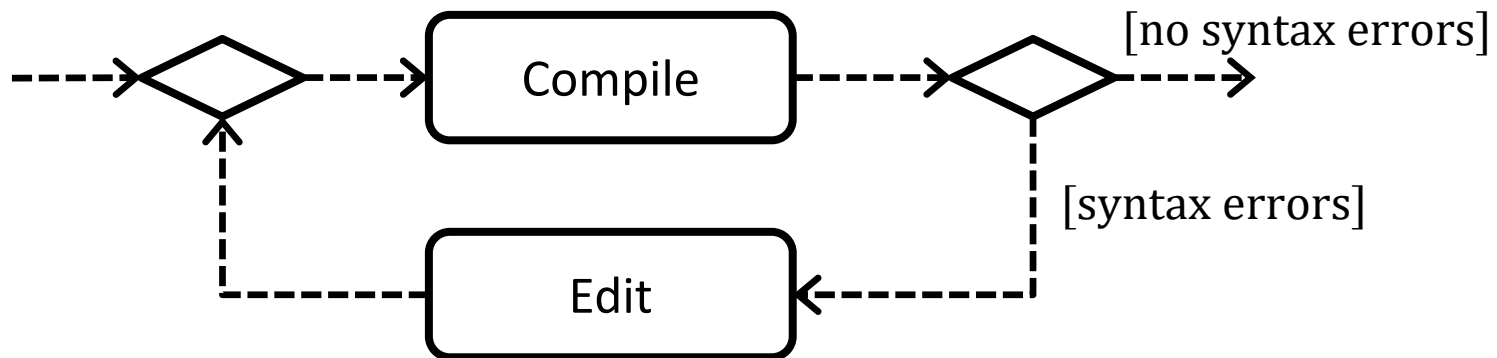


# Definition: Schleife

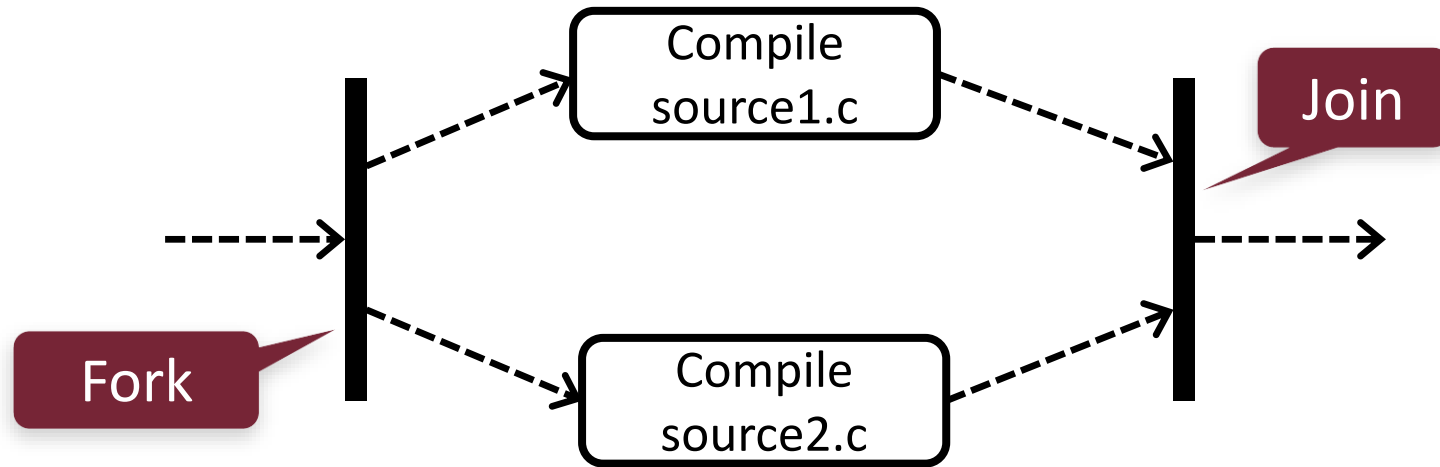
Eine **Schleife** ist eine Kontrollstruktur, die eine mehrfache Ausführung definiert. Die Schleife

- besteht aus einer **Entscheidung (Decision)** und einer **Vereinigung (Merge)**, wo
- einer der Zweige/Ausgänge der Entscheidung zurück zur Vereinigung führt.

- Anmerkung: dies entspricht einer fußgesteuerten Schleife.



# Fork / Join



- Semantik
  - Unbestimmte Ausführungsreihenfolge
  - Parallele oder sich überlappende Ausführung
- Siehe: LVA „Rechnerarchitekturen“

# Definition: Parallele Ausführung

## Die **parallele Ausführung**

- besteht aus einer **Gabelung (Fork)** und einer **Zusammenführung/Synchronisation (Join)** Kontrollelement,
- kann eine beliebige Anzahl von Zweigen/Ausgängen haben,
- die Zweige werden **nebenläufig** ausgeführt,
- alle Zweige führen zu derselben Zusammenführung,
- die parallele Ausführung ist beendet, wenn alle Zweige beendet sind.

Zwei Aktivitäten sind **nebenläufig**, wenn ihre Ausführungsreihenfolge nicht bestimmt ist. ( $\neq$  gleichzeitig!)

- Anmerkung: typischerweise arbeiten wir mit zwei parallelen Zweigen.
- **NICHT zu verwechseln mit der Entscheidung!**

# Flow begin / flow end

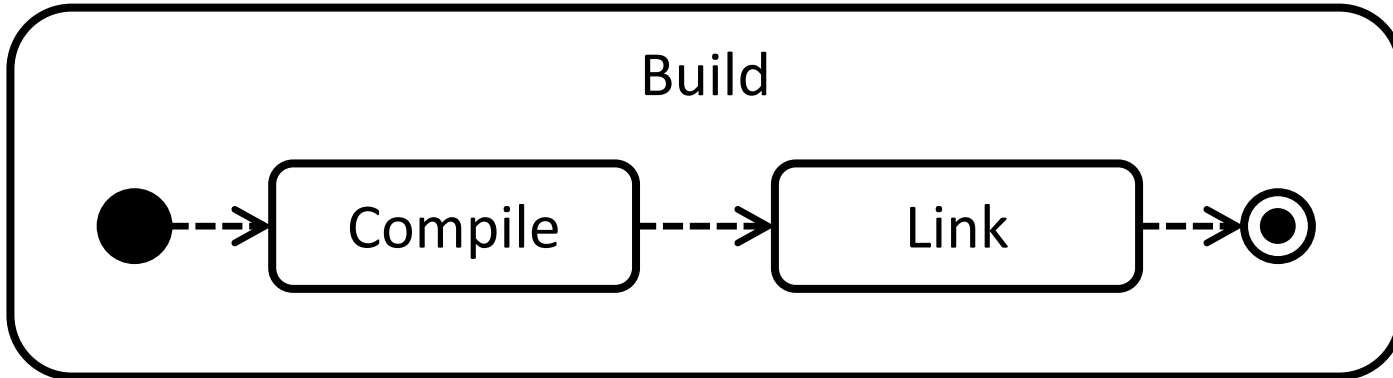


# Definition: Anfang/Ende des Prozesses

Jeder Prozess hat ein Anfangskontrollelement (**Flow Begin**) und ein Endekontrollelement (**Flow End**).

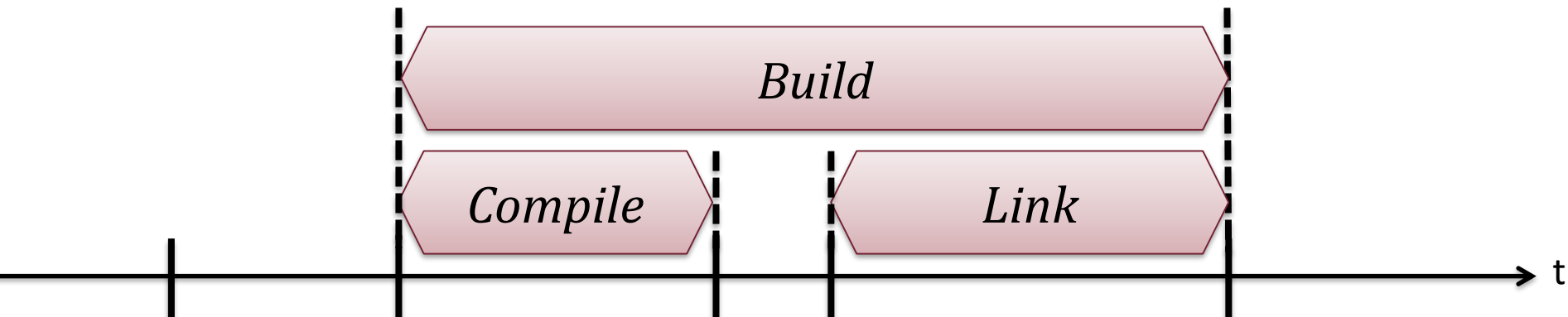
- Das Anfangskontrollelement ist das erste Element des (Unter-) Prozesses, das keinen Eingang und genau einen Ausgang hat.
  - Das Endekontrollelement ist das letzte Element des (Unter-) Prozesses, das genau einen Eingang und keinen Ausgang hat.
- 
- Anmerkung: Hier wird es nicht modelliert, was löst die Instanziierung des Prozesses aus, oder was stößt den Prozess an.

# Hierarchie



## ■ Zusammengesetzte Aktivitäten

- (Zusammengesetzte) Prozesse: *Build*
- Unterprozesse / Sub-Prozesse / Schritte: *Compile*, *Link*

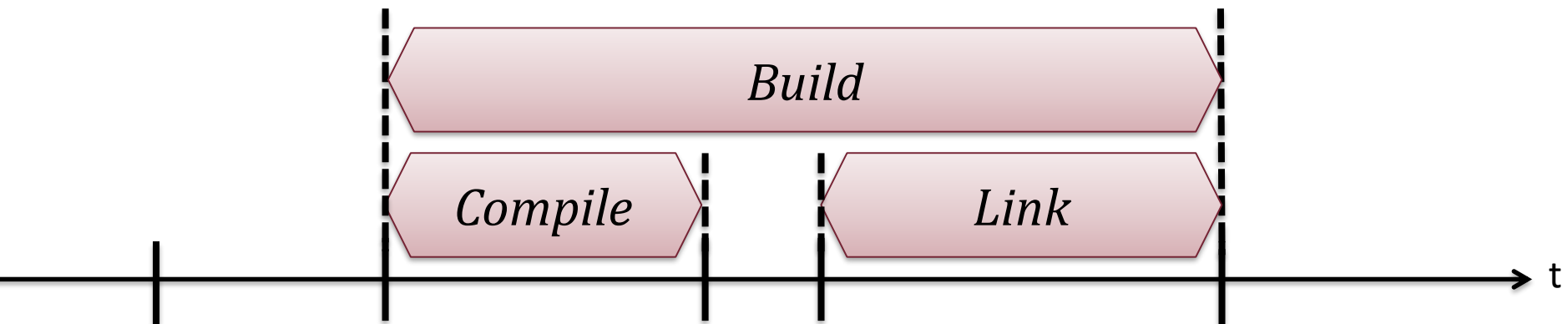
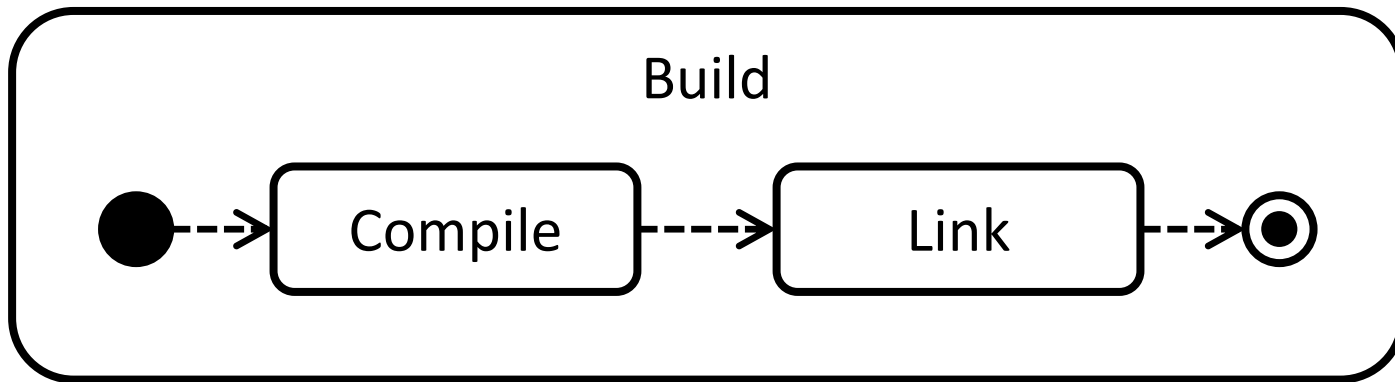


# Definition: Hierarchie

## Hierarchisches Prozessmodell:

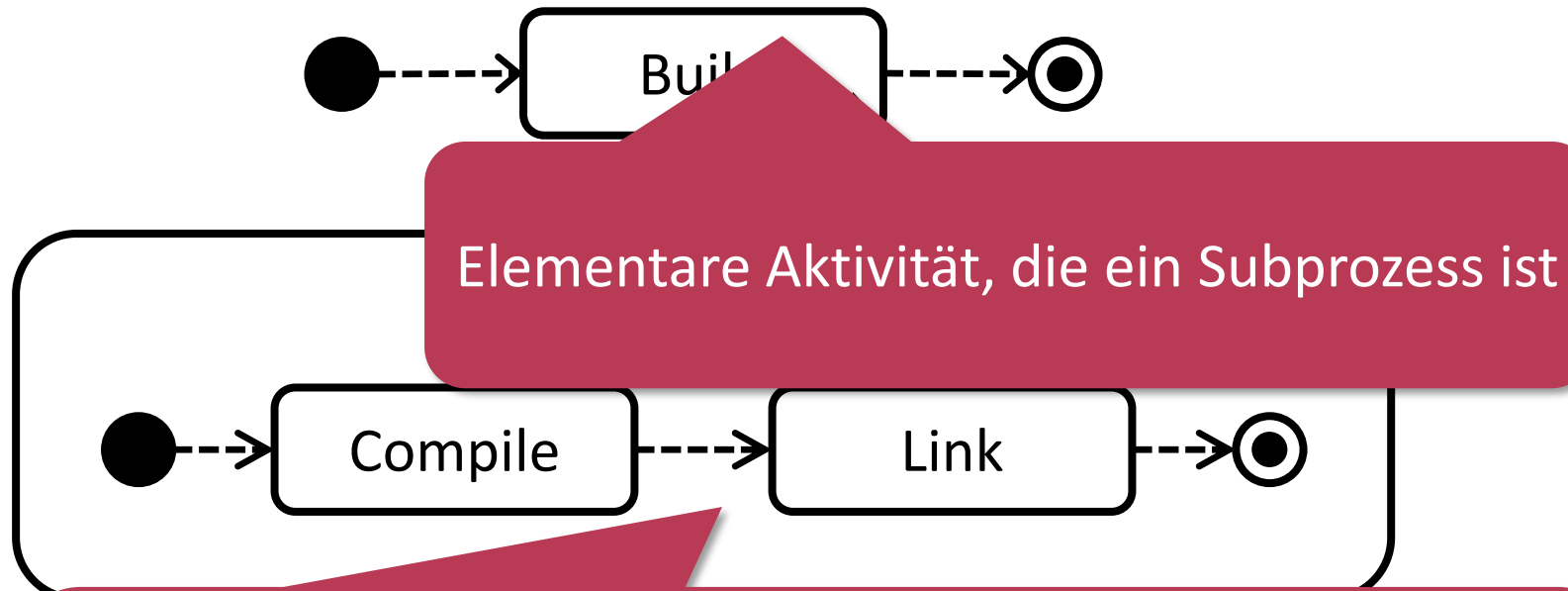
- Es kann anstatt manchen elementaren Aktivitäten auch mit einem Prozessmodell beschriebenen Teilmodelle enthalten. (hierarchische Verfeinerung)

# Referenz / Verweis / Anruf





# Referenz / Verweis / Anruf

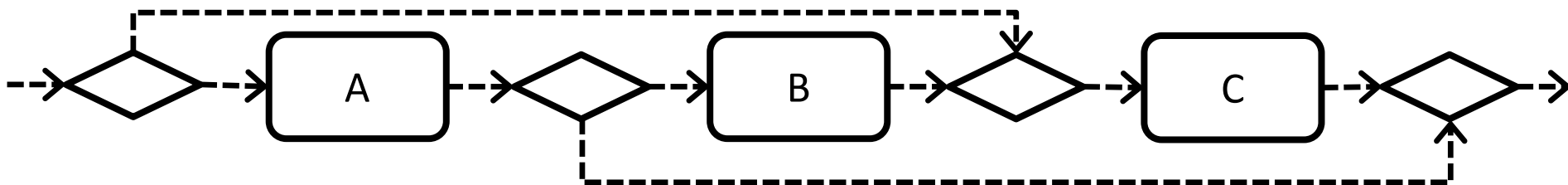


Kann in den „Hauptprozess“ eingebettet werden, wenn die Verfeinerung richtig ist, das ist

- die Schritte gemeinsam produzieren das selbe
- keine unbehandelte Fälle auf der aufrufenden seite
- Konsistenz der Eingaben/Ausgaben

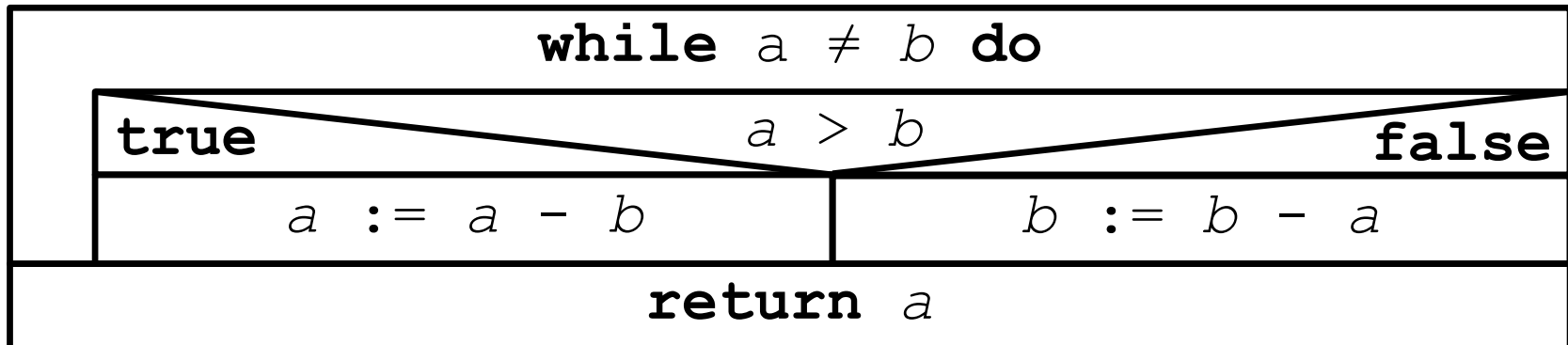
# Wohlstrukturierte Prozesse

- Von Kontrollblöcken aufgebaut
  - Ein Anfang, ein Ende, inzwischen ein wohlstrukturierter Block
  - Sequenz, Verzweigung, parallele Ausführung, Schleife
  - elementare Aktivität (leerer Block auch!)
- Analogie: strukturierte Programmierung (anstatt `goto` werden Kontrollstrukturen verwendet)
- Beispiel: nicht-wohlstrukturierter Prozess

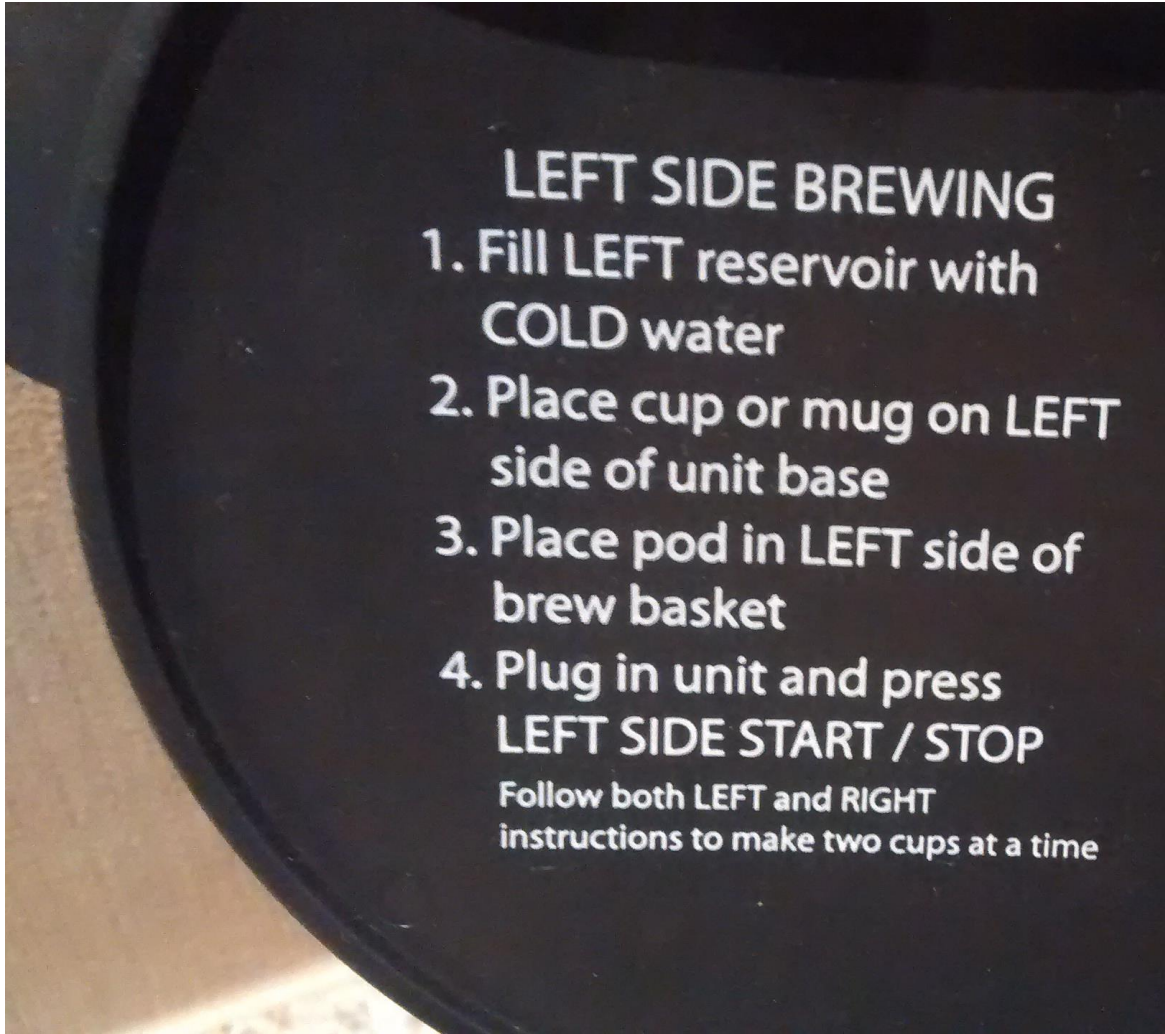


# Wohlstrukturierte Prozesse

- Erzwingt von bestimmten Formalismen
  - z.B. BPEL (Geschäftsprozess durch Web Services)
  - z.B. Struktogramm (Nassi-Shneiderman)



# Zwillingskaffeemaschine (Prozessmodell)

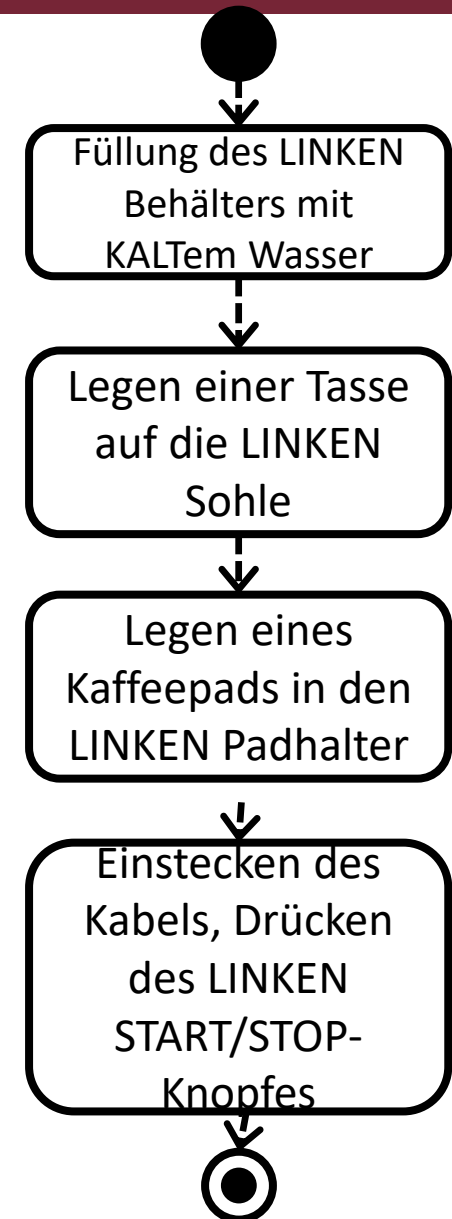
- 
- LEFT SIDE BREWING**
1. Fill LEFT reservoir with COLD water
  2. Place cup or mug on LEFT side of unit base
  3. Place pod in LEFT side of brew basket
  4. Plug in unit and press LEFT SIDE START / STOP
- Follow both LEFT and RIGHT instructions to make two cups at a time

1. Füllen Sie den LINKEN Behälter mit KALTem Wasser
2. Legen Sie eine Tasse auf die LINKE Sohle
3. Legen Sie ein Kaffeepad in den LINKEN Padhalter.
4. Stecken Sie den Kabel ein, und drücken Sie den LINKEN START/STOP-Knopf.

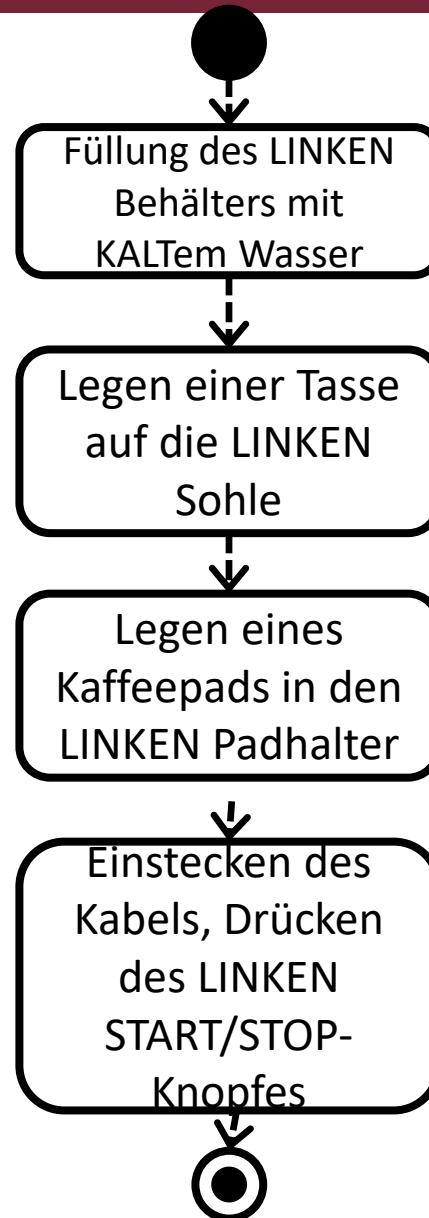
Folgen Sie die LINKE und RECHTE Anweisung, um zwei Tassen Kaffee gleichzeitig zu machen.

# Zwillingskaffeemaschine (Prozessmodell)

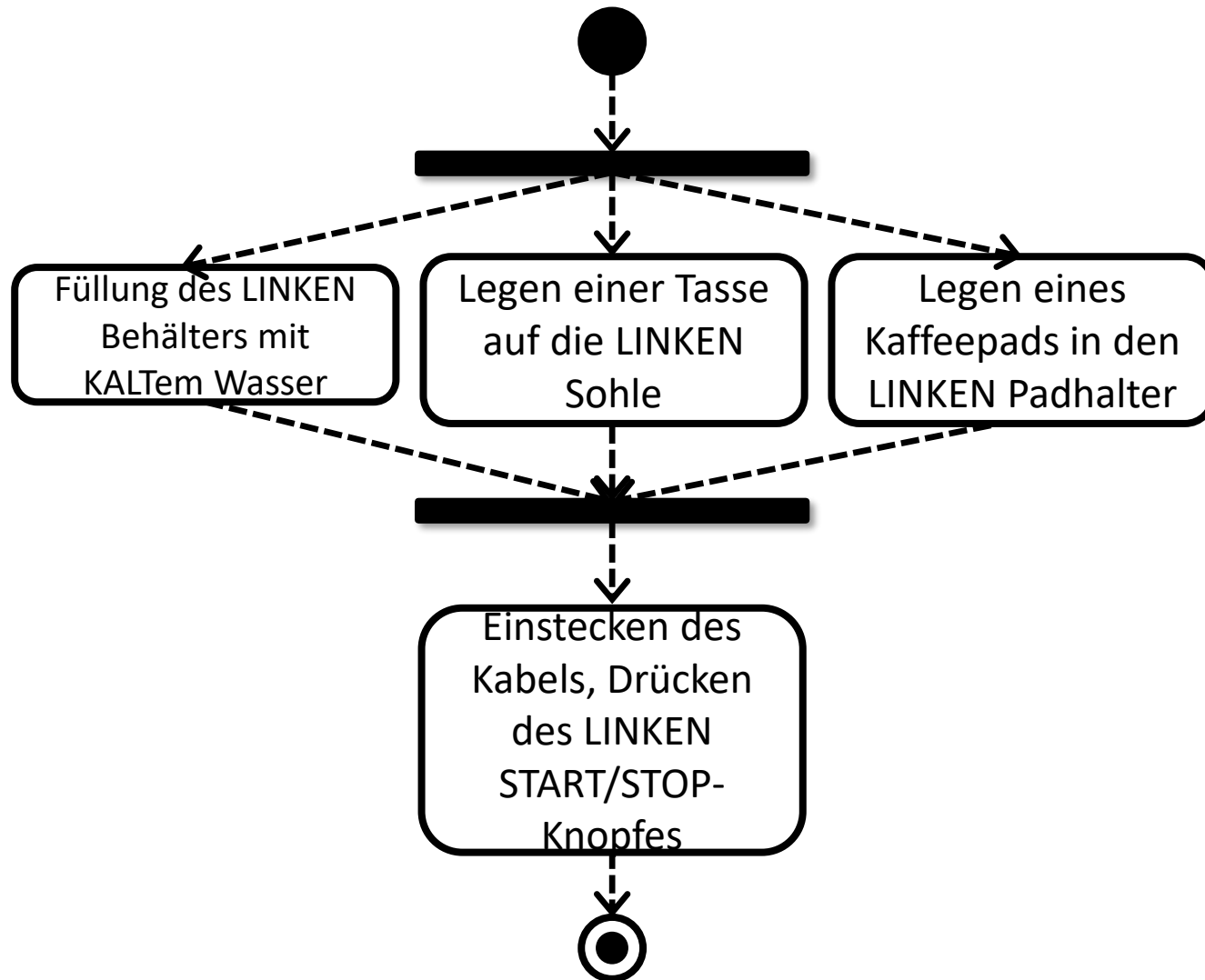
- LEFT SIDE BREWING**
1. Fill LEFT reservoir with COLD water
  2. Place cup or mug on LEFT side of unit base
  3. Place pod in LEFT side of brew basket
  4. Plug in unit and press LEFT SIDE START / STOP
- Follow both LEFT and RIGHT instructions to make two cups at a time



# Zwillingskaffeemaschine (Prozessmodell)



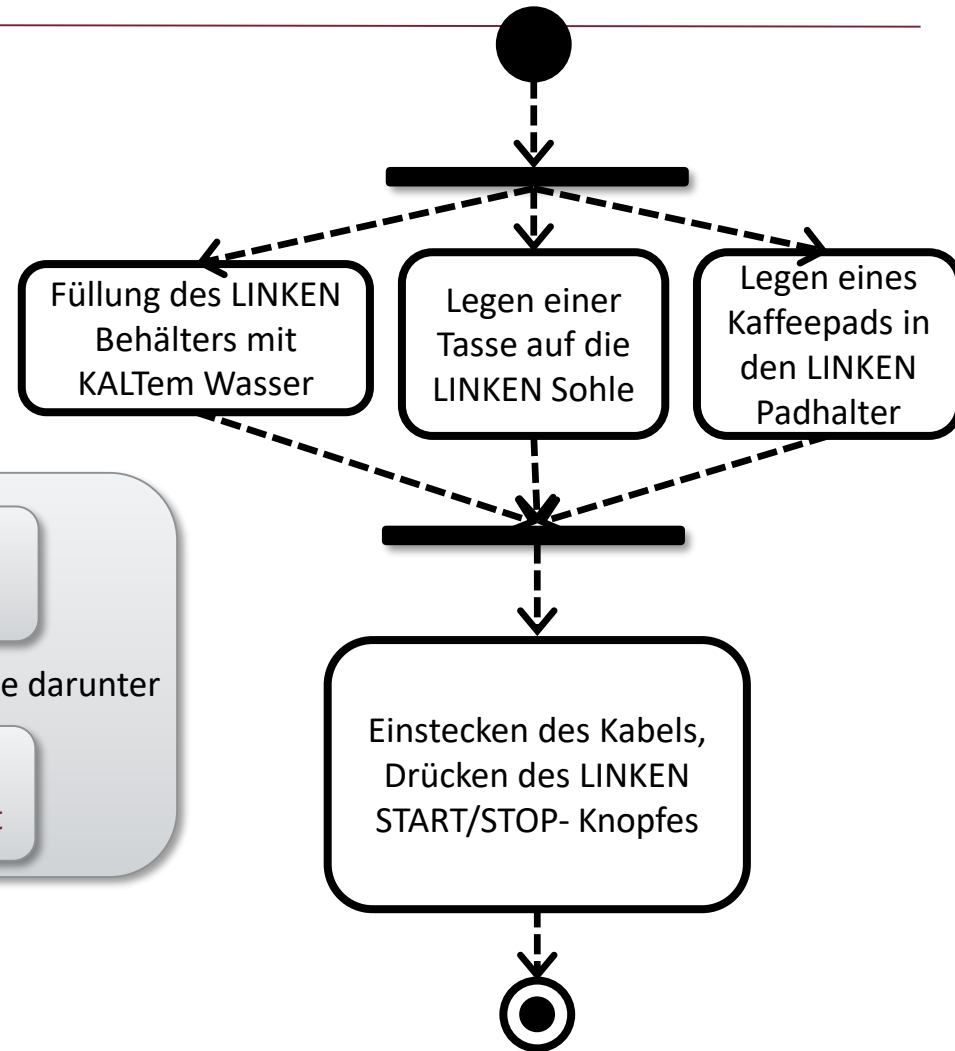
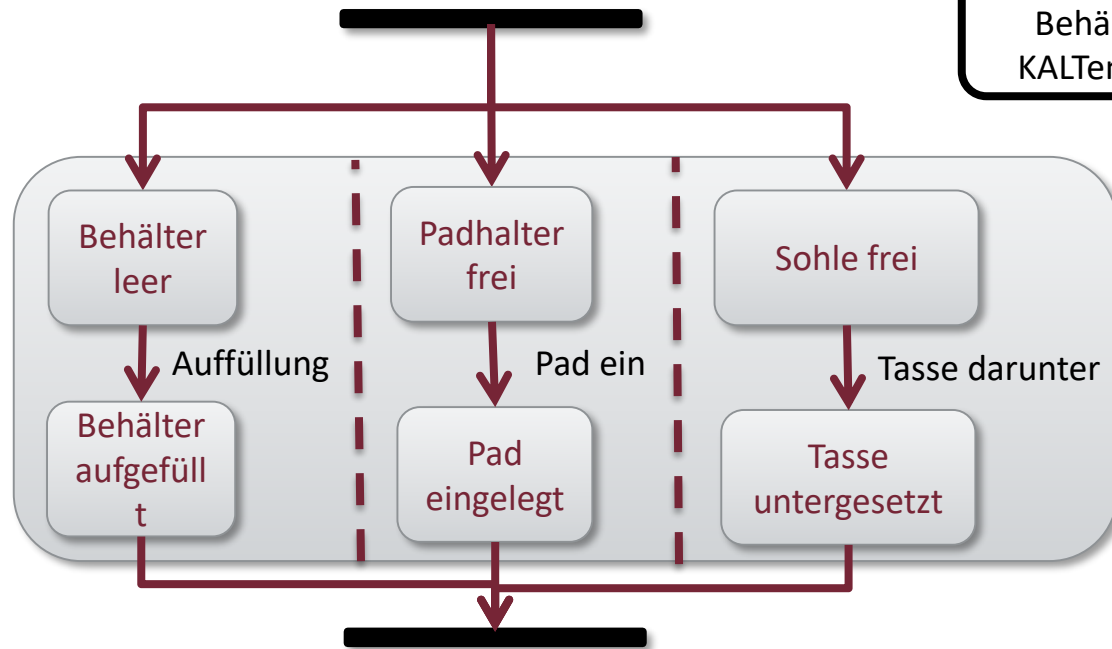
# Zwillingskaffeemaschine (Prozessmodell)



# Vergleich

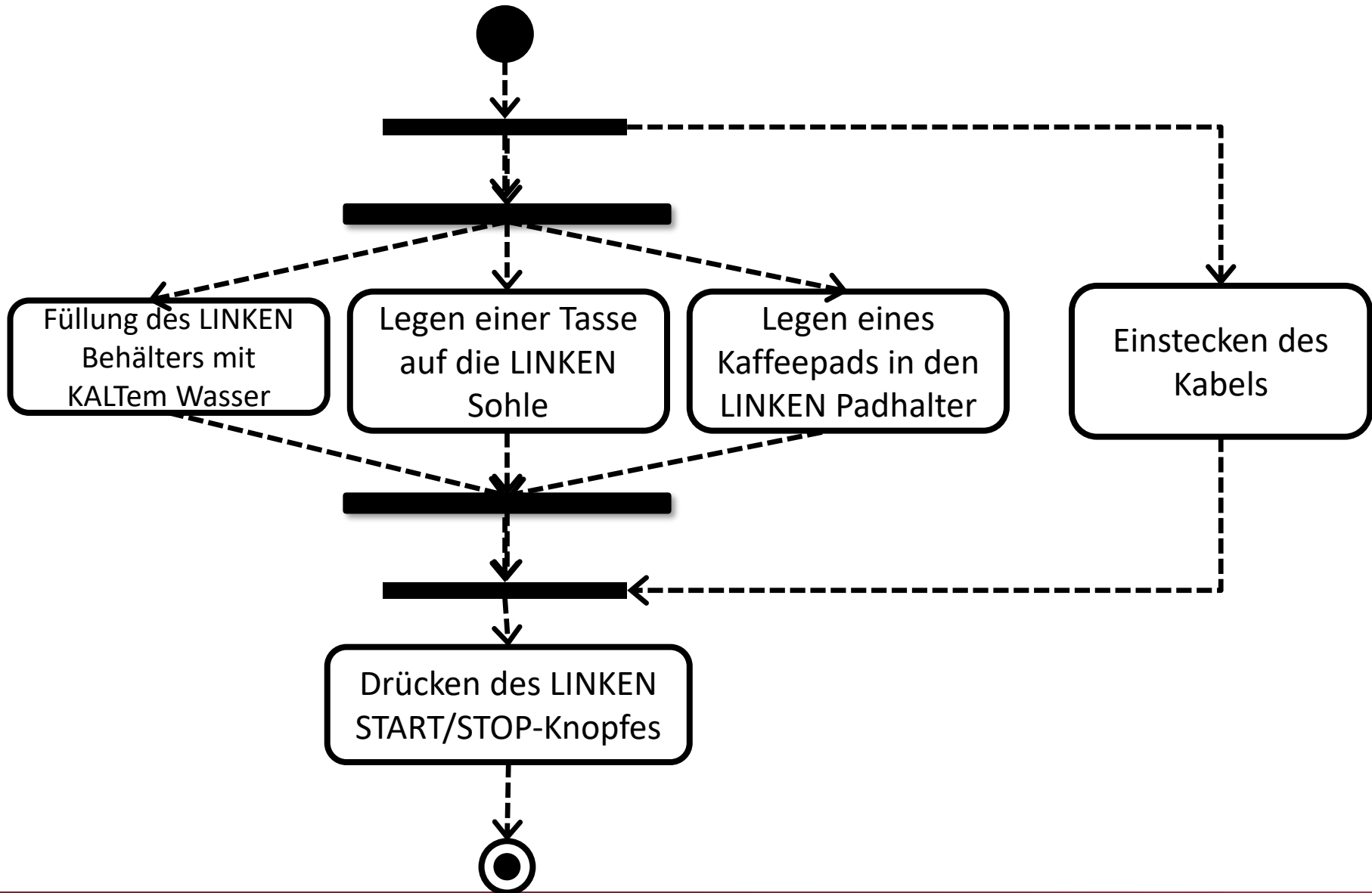
## ■ Zustandsmaschine

## ■ Prozess

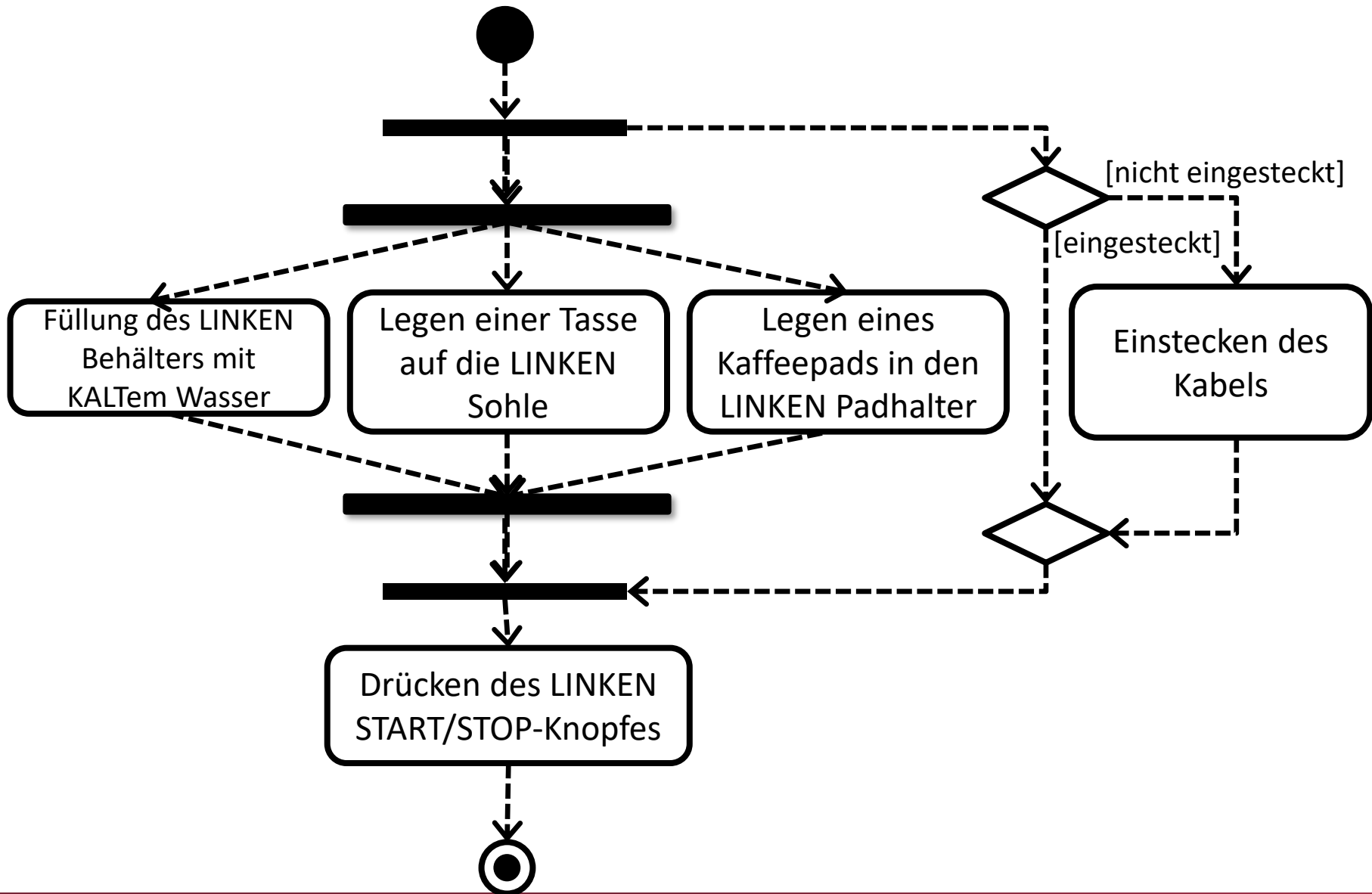




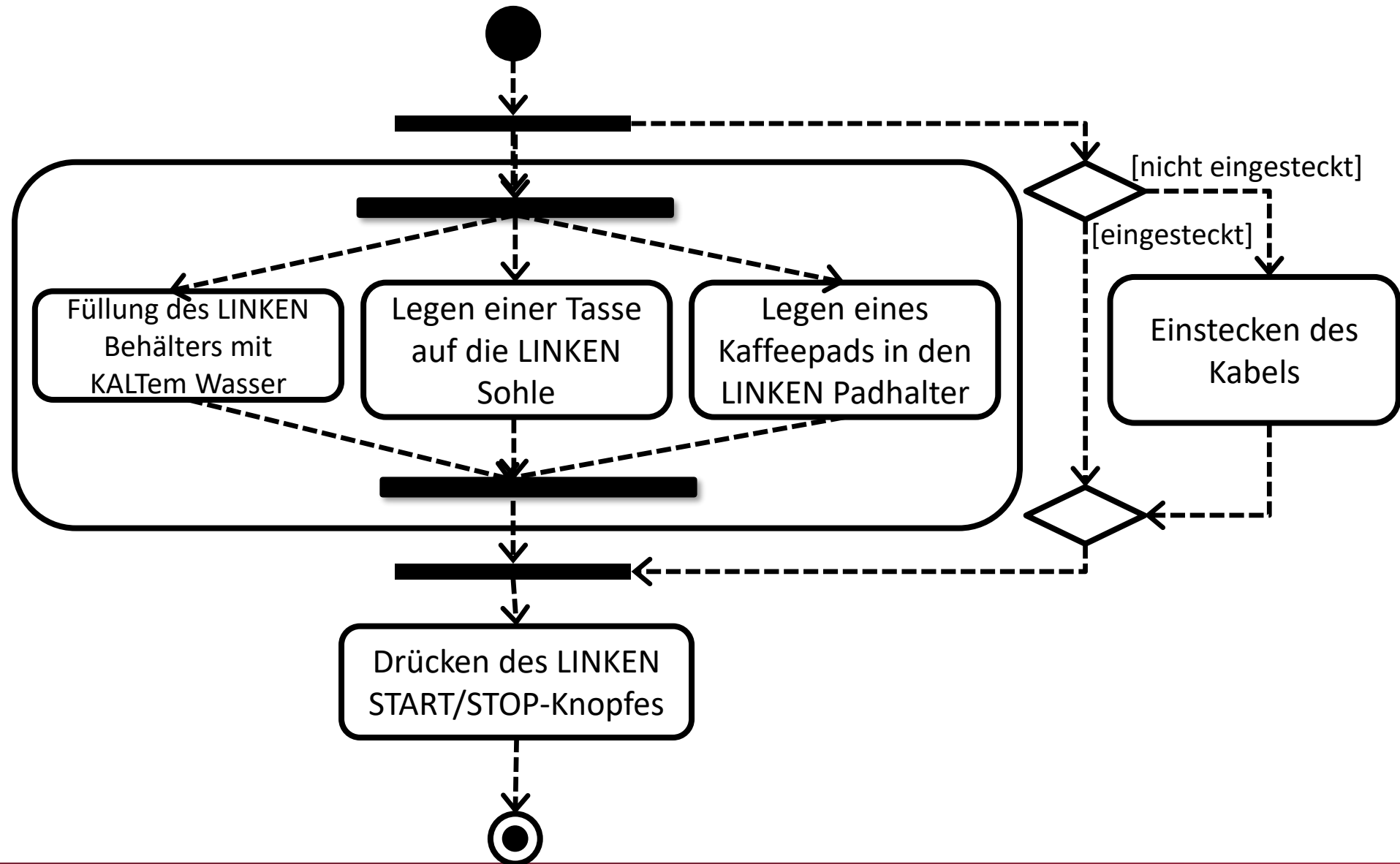
# Zwillingskaffeemaschine (Prozessmodell)



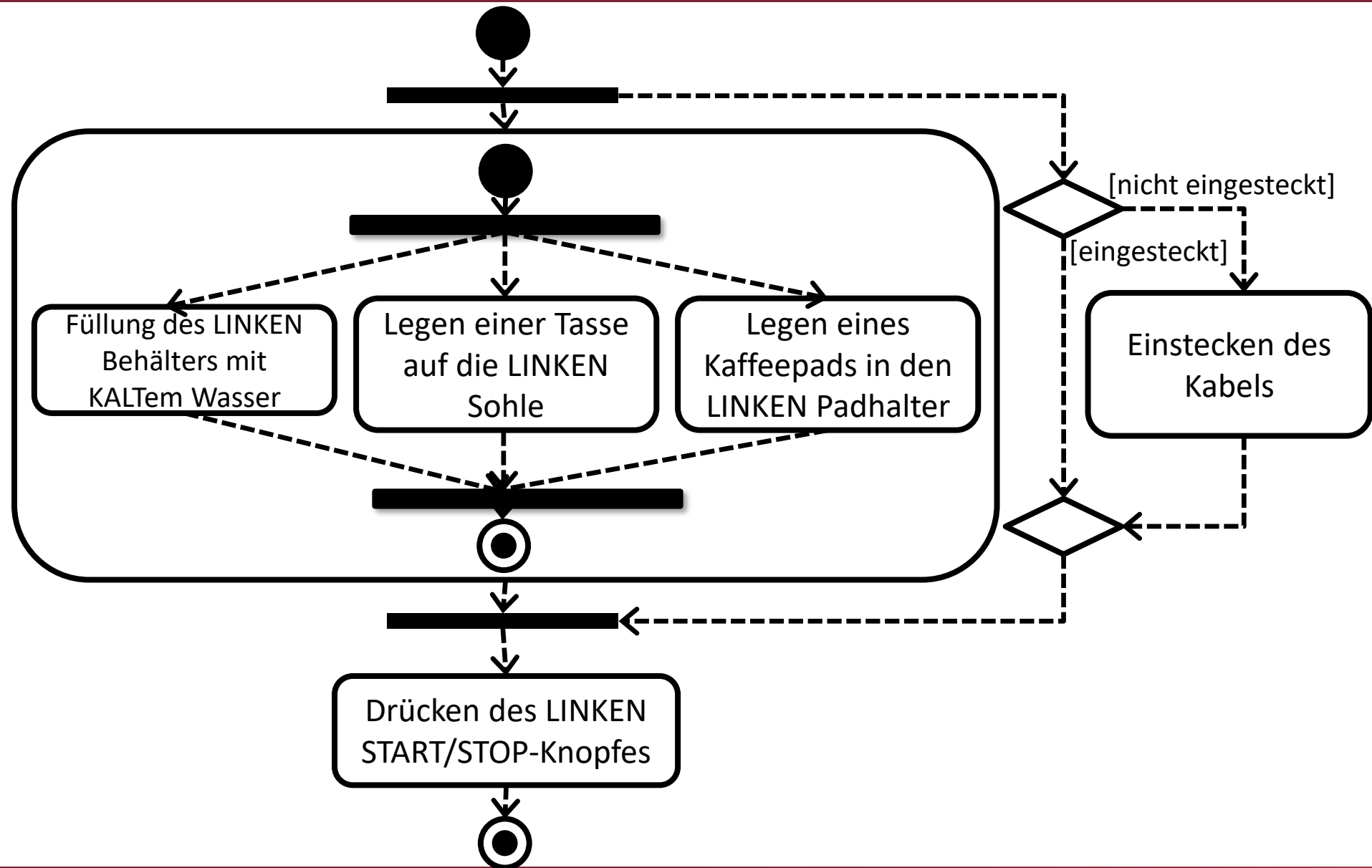
# Zwillingskaffeemaschine (Prozessmodell)



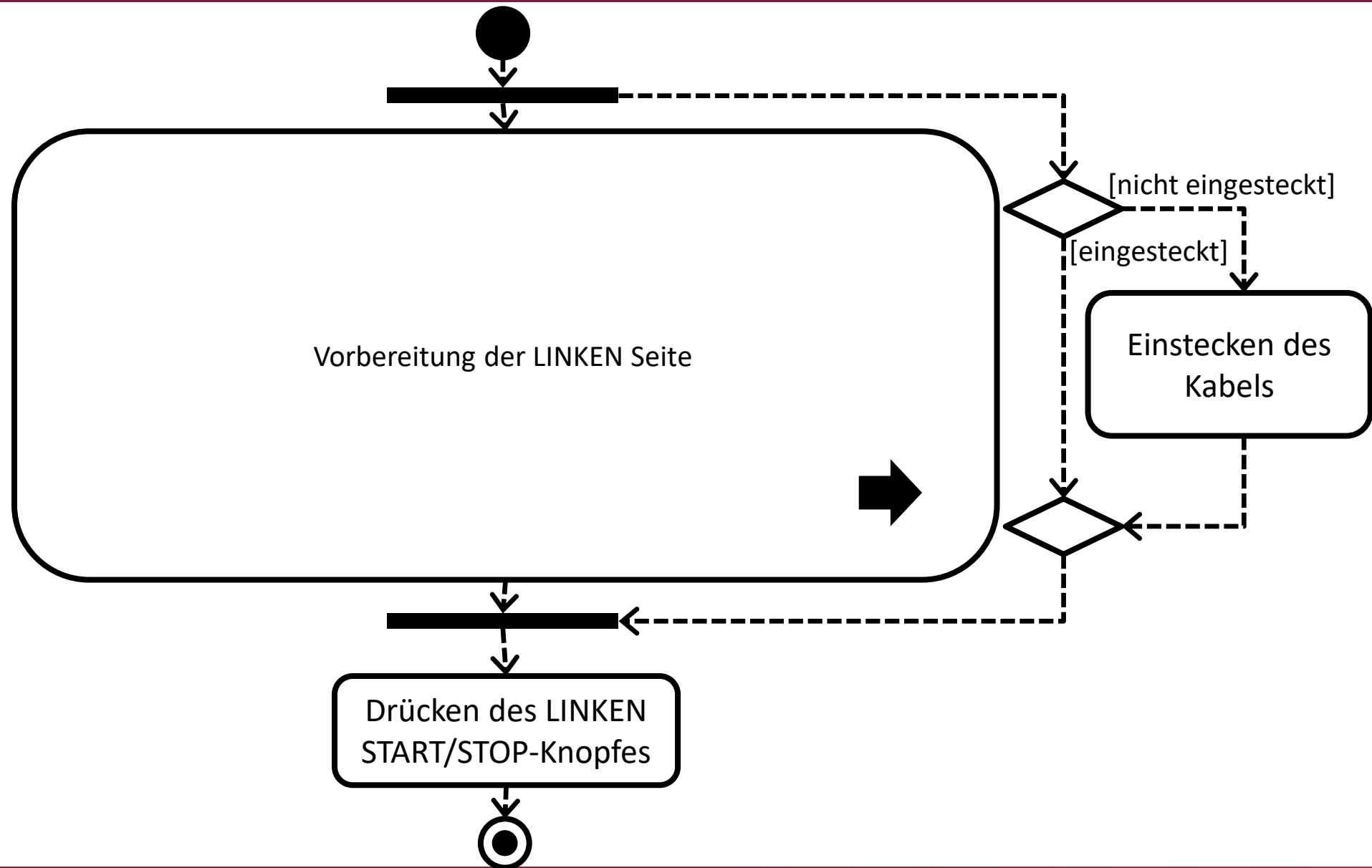
# Zwillingskaffeemaschine (Prozessmodell)



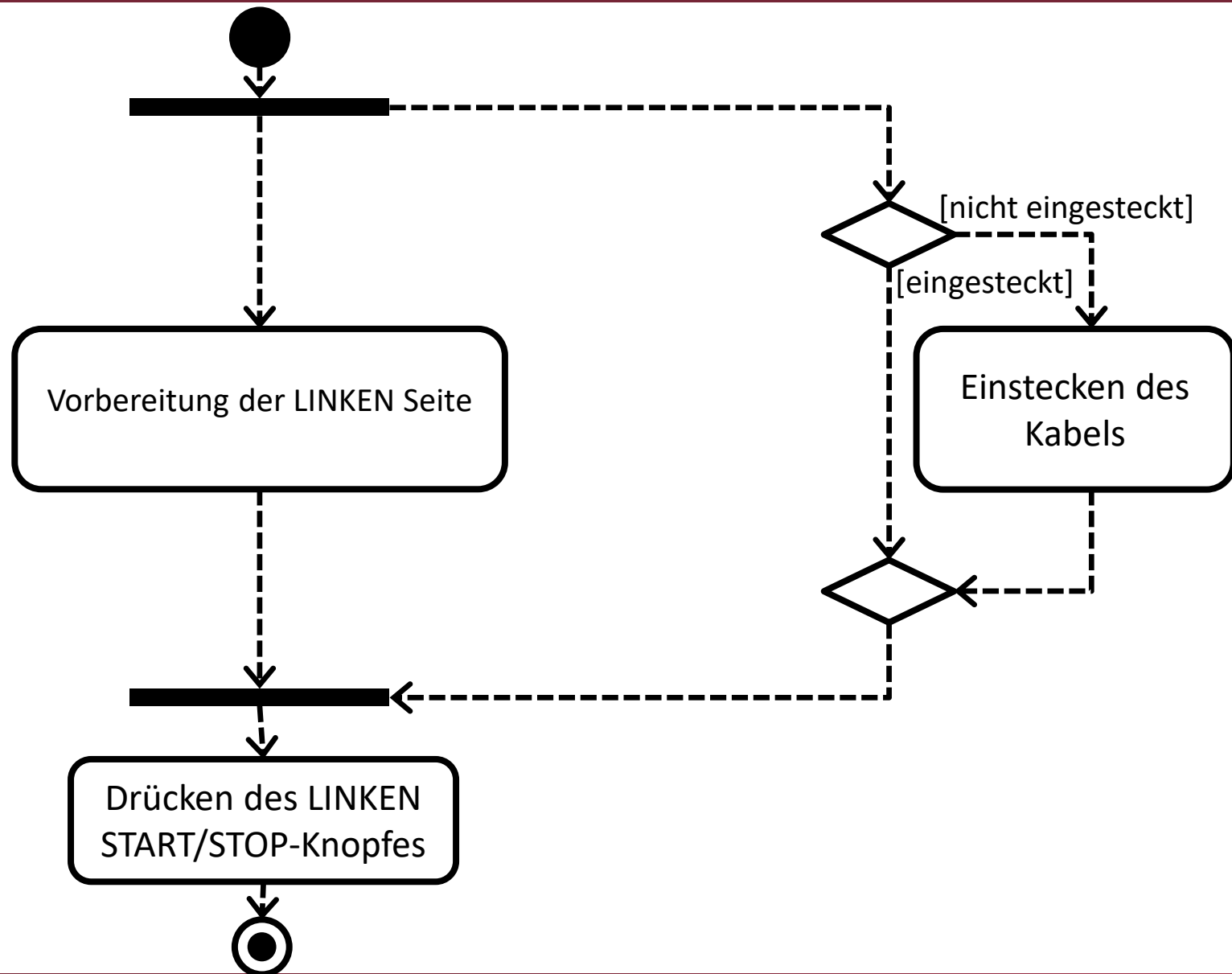
# Zwillingskaffeemaschine (Prozessmodell)



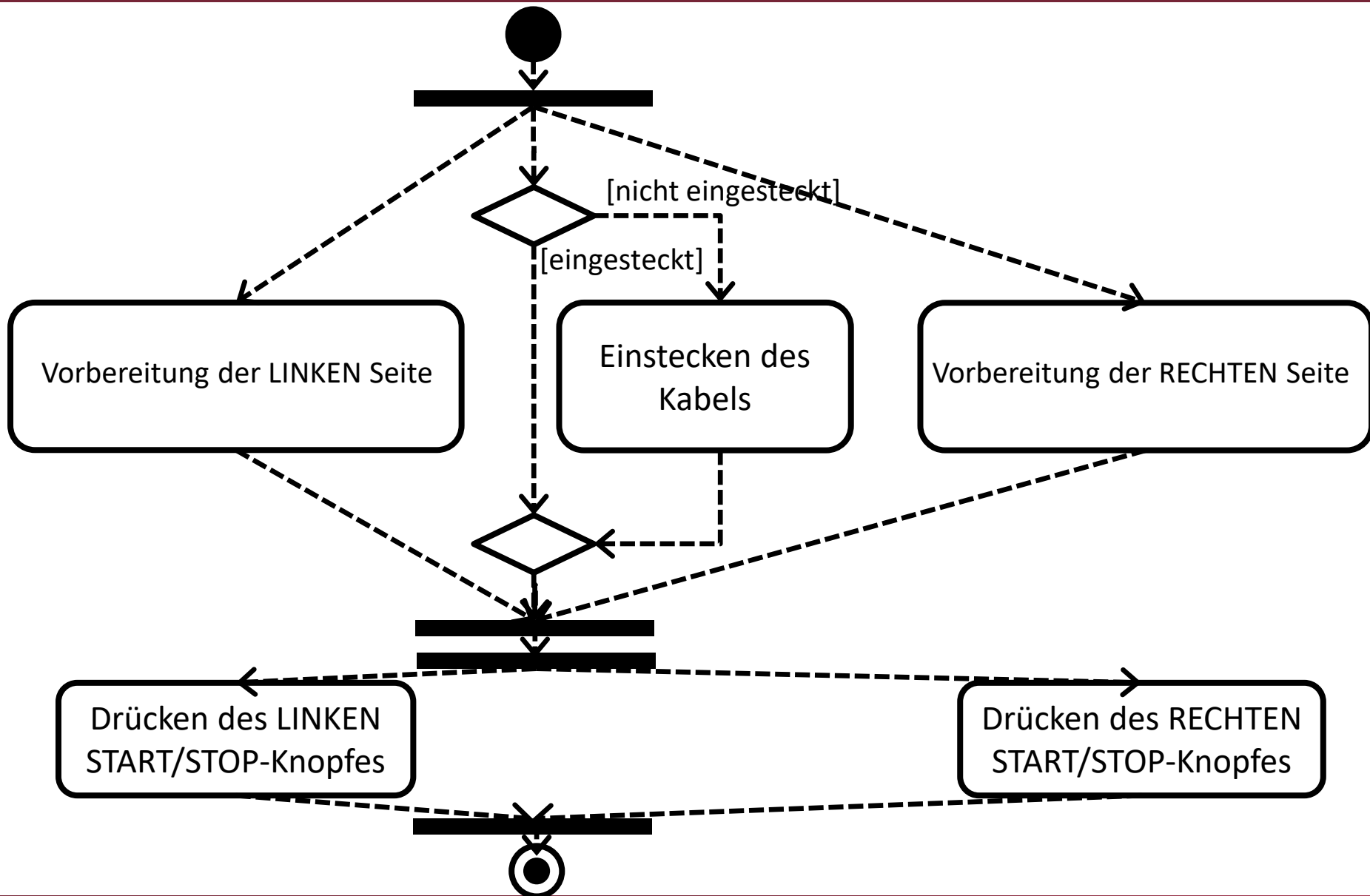
# Zwillingskaffeemaschine (Prozessmodell)



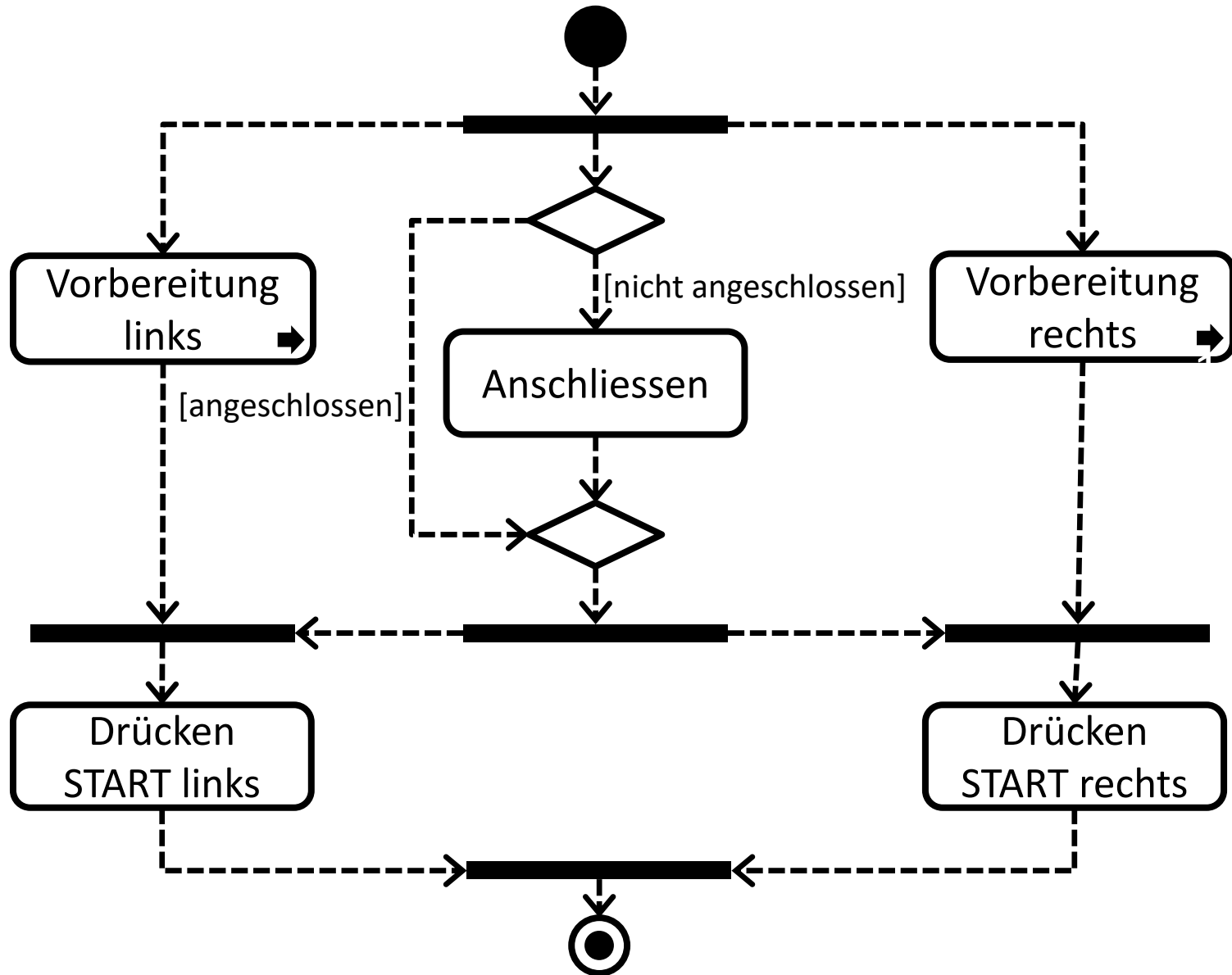
# Zwillingskaffeemaschine (Prozessmodell)



# Zwillingskaffeemaschine (Prozessmodell)

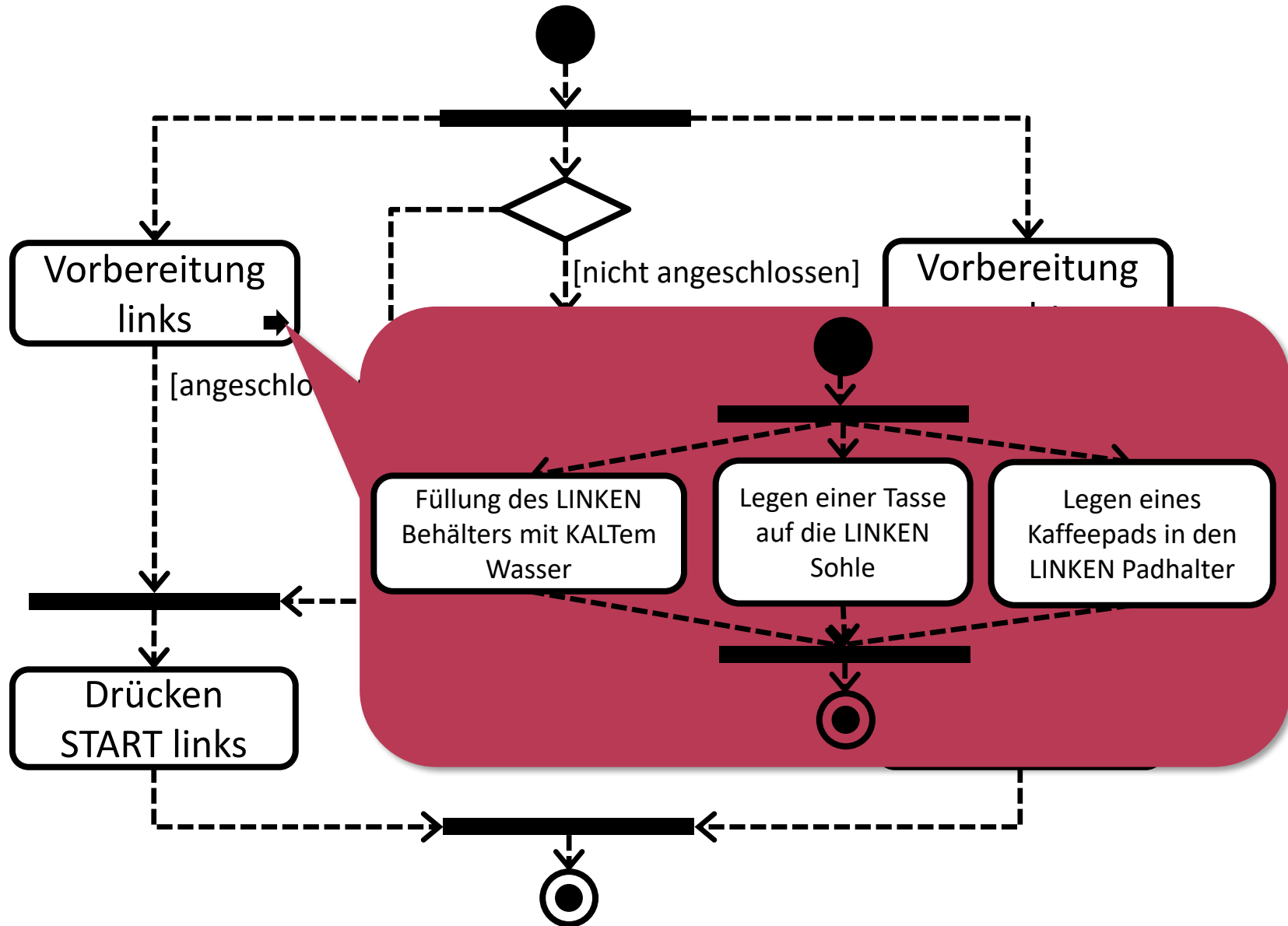


# Zwillingskaffeemaschine (Prozessmodell)





# Zwillingskaffeemaschine (Prozessmodell)



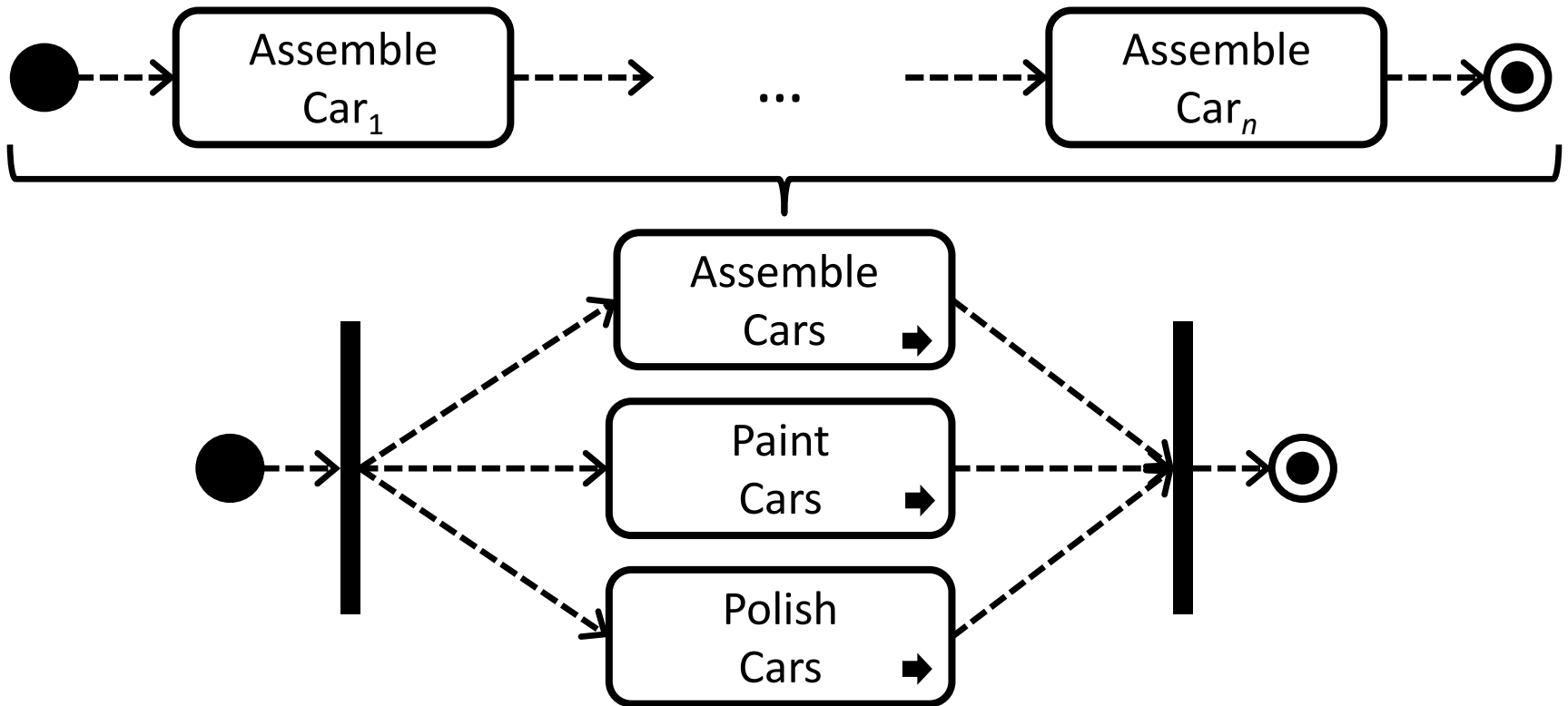
# Modellierung aus mehreren Aspekten



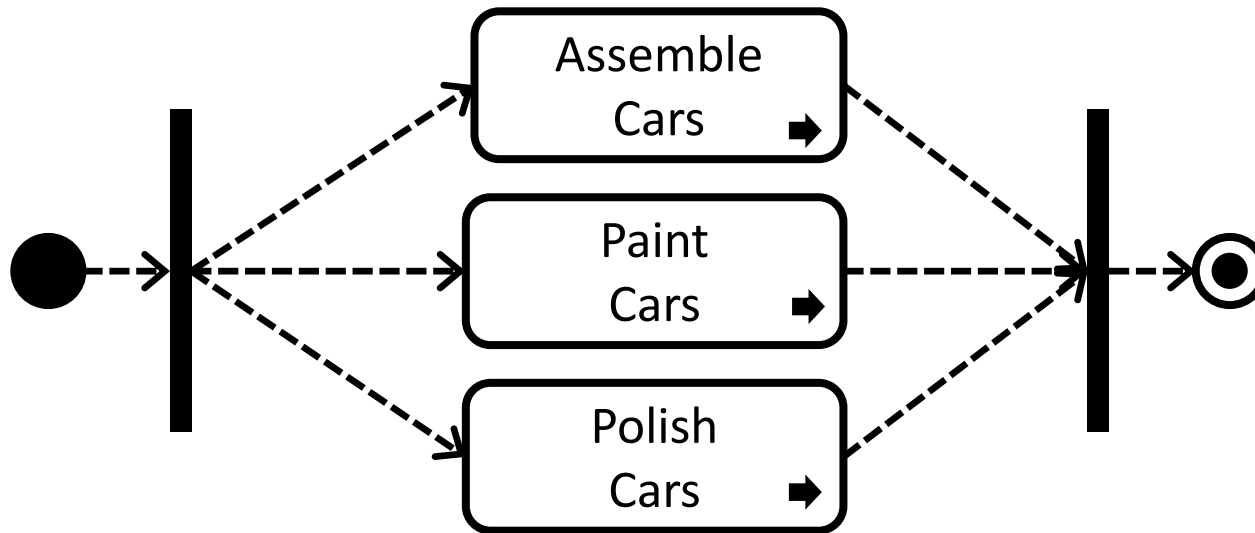
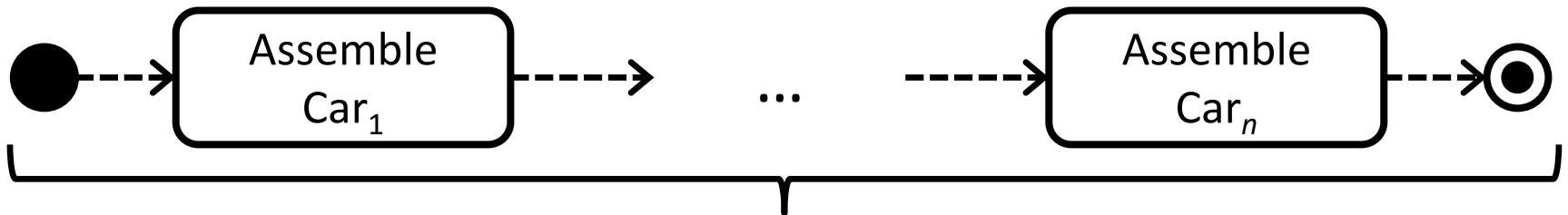
# Was passiert mit einem Wagen?



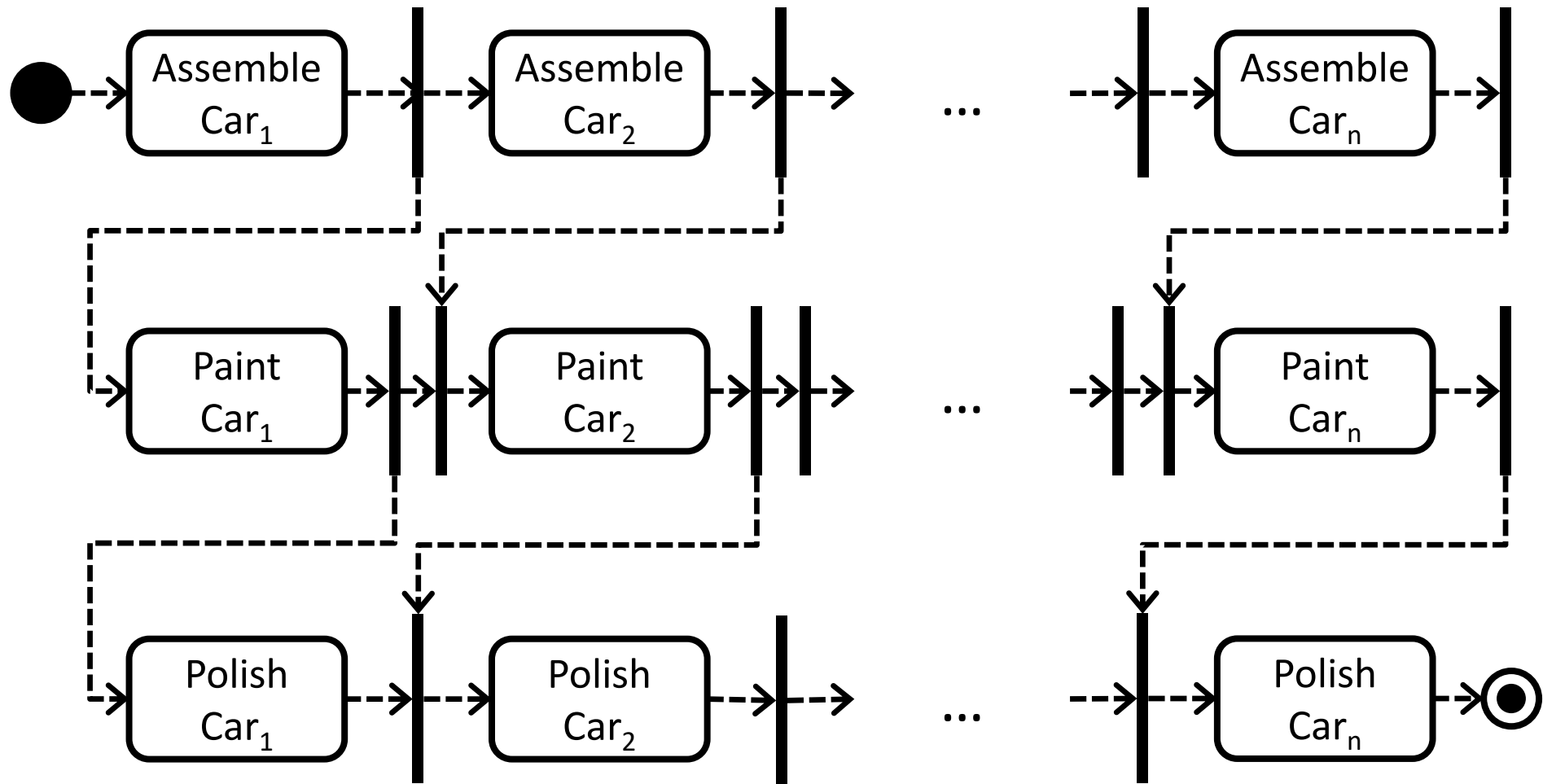
# Was passiert an einem Produktionsband?



# Modellierung aus mehreren Aspekten

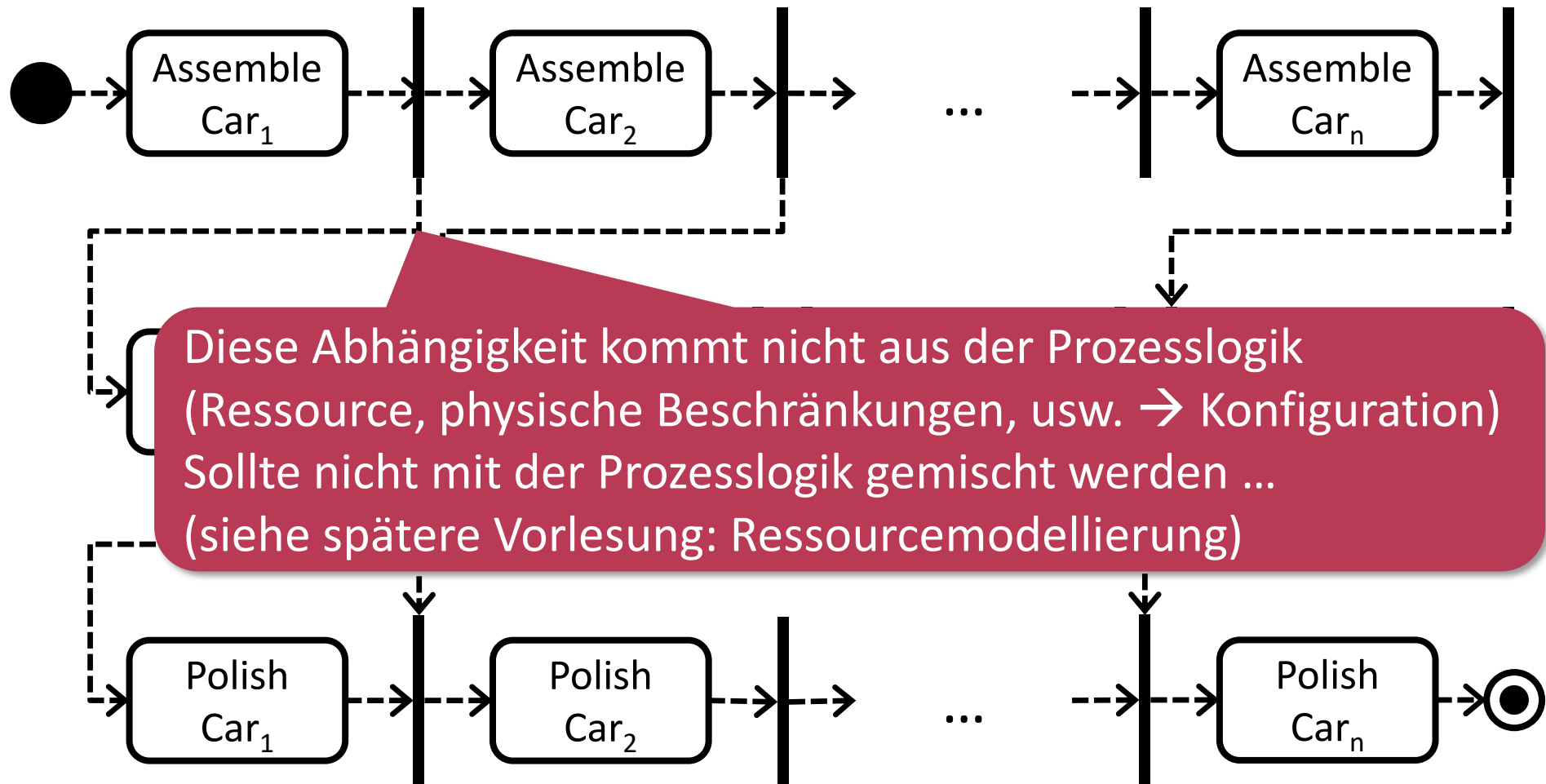


# Die gemeinsame Ansicht



- Enthält alles, ist aber nicht sehr praktisch

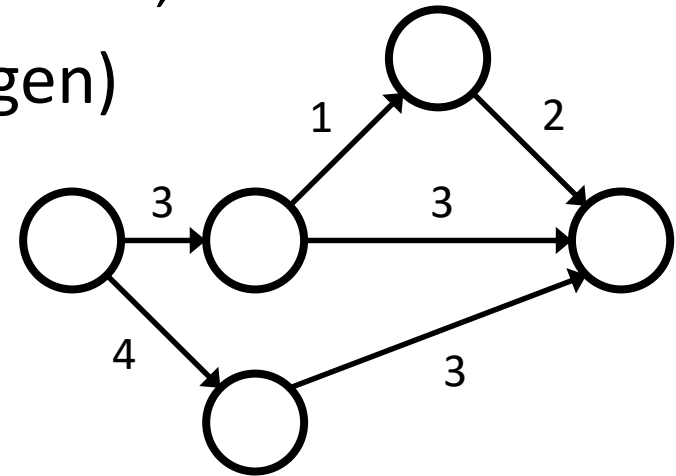
# Die gemeinsame Ansicht



- Enthält alles, ist aber nicht sehr praktisch

# Die gemeinsame Ansicht

- Das 2D-fork-join-Netz ist nicht sehr praktisch
  - Getrennte Prozesse für die verschiedenen Aspekte (Für die Lebensläufe der Wagen und der Produktionsbänder)
- Viele fork-join in kompakterer Form? → PERT chart
  - Program Evaluation and Review Technique
    - Für die Analyse der Ausführungszeiten, siehe Einf. in die Rechentheorie 2
  - (PERT kennt keine Verzweigungen)





Wiederholung

Ziel der  
Prozessmodellierung

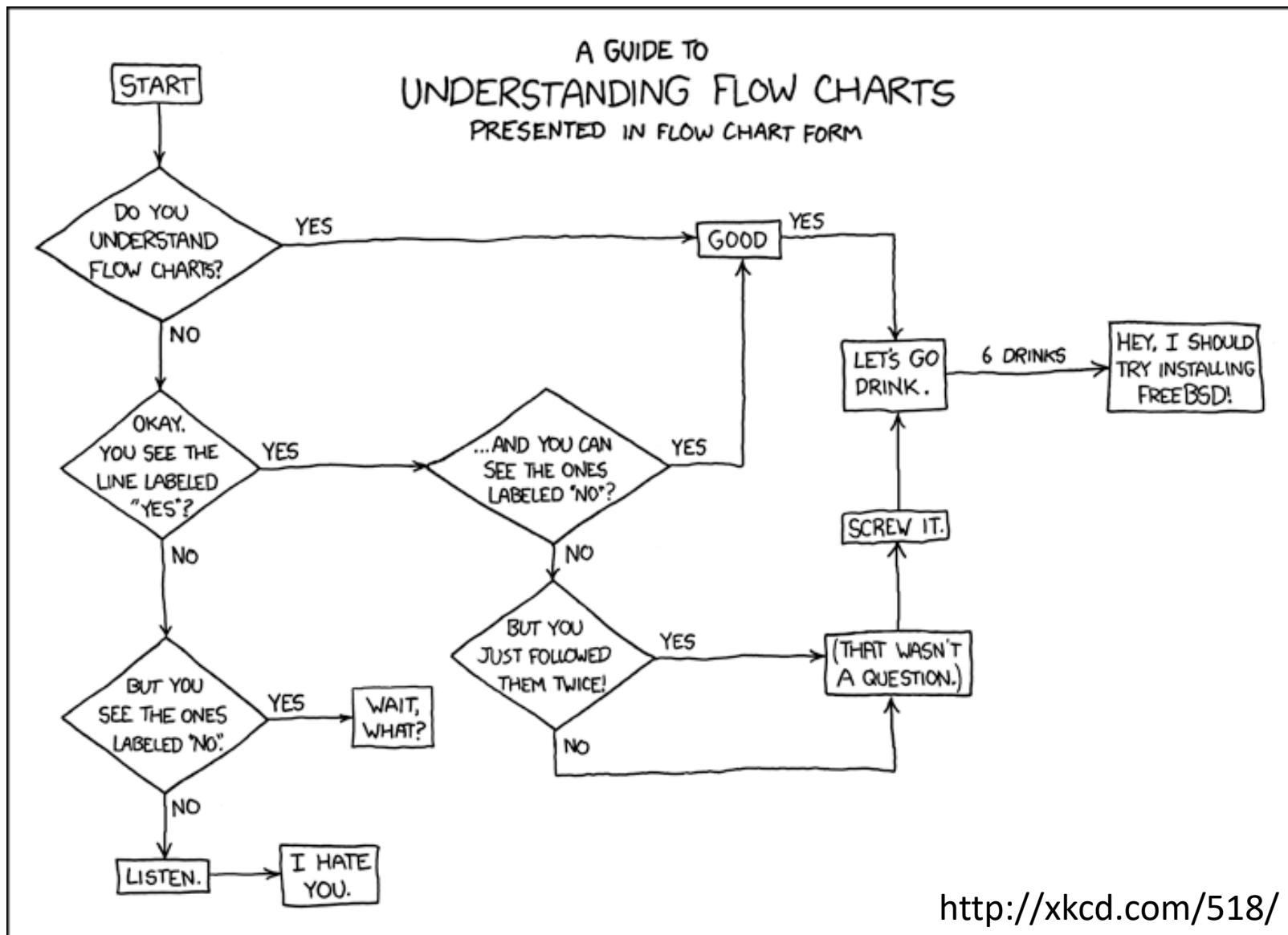
Prozess-  
modelle

Kontroll-  
fluss

Verwirklichung

# KONTROLLFLUSS-MODELLE

# Flowchart

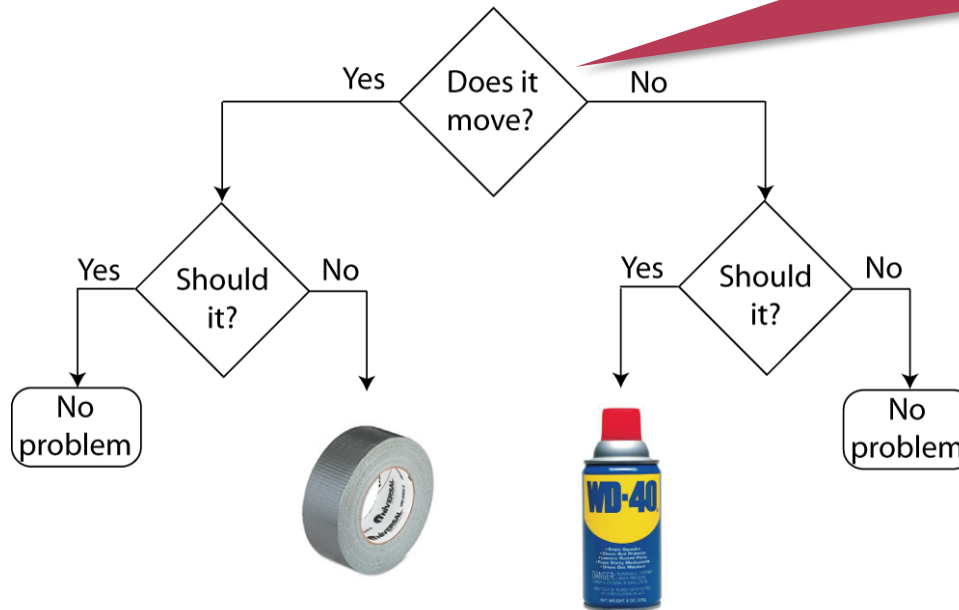


<http://xkcd.com/518/>

# Flowchart

- Flowchart / Entscheidungsdiagramm
  - Beschreibt Entscheidungsprozesse
    - führt zu einer Konklusion
  - Beschreibt keine zeitliche Sequenzen
- Spezieller Fall: Entscheidungsbaum

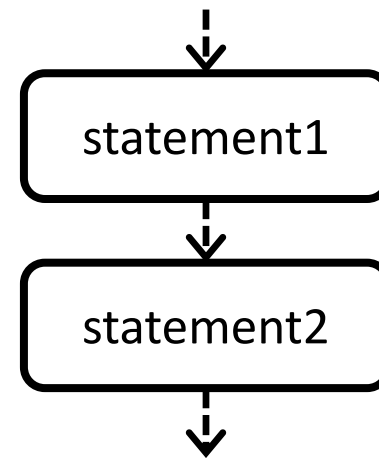
In der Praxis ist die Bestimmung der Entscheidungspunkte und deren Reihenfolge schwierig



# Kontrollfluss

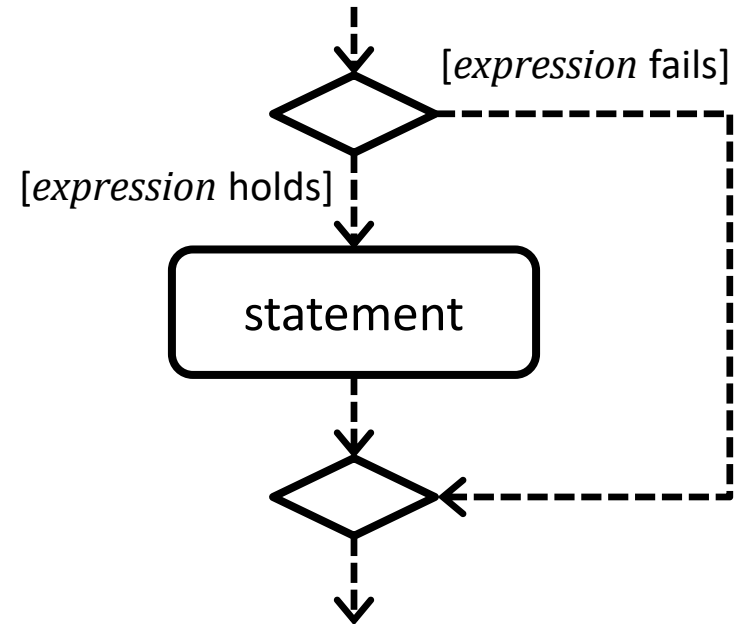
*<statement1>*

*<statement2>*



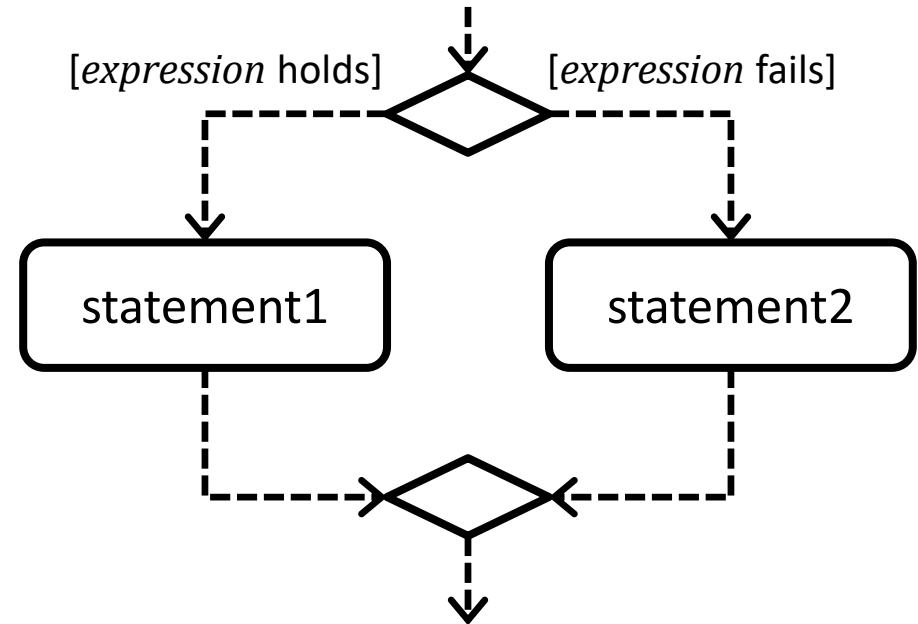
# Kontrollfluss

**if** (*<expression>*)  
*<statement>*



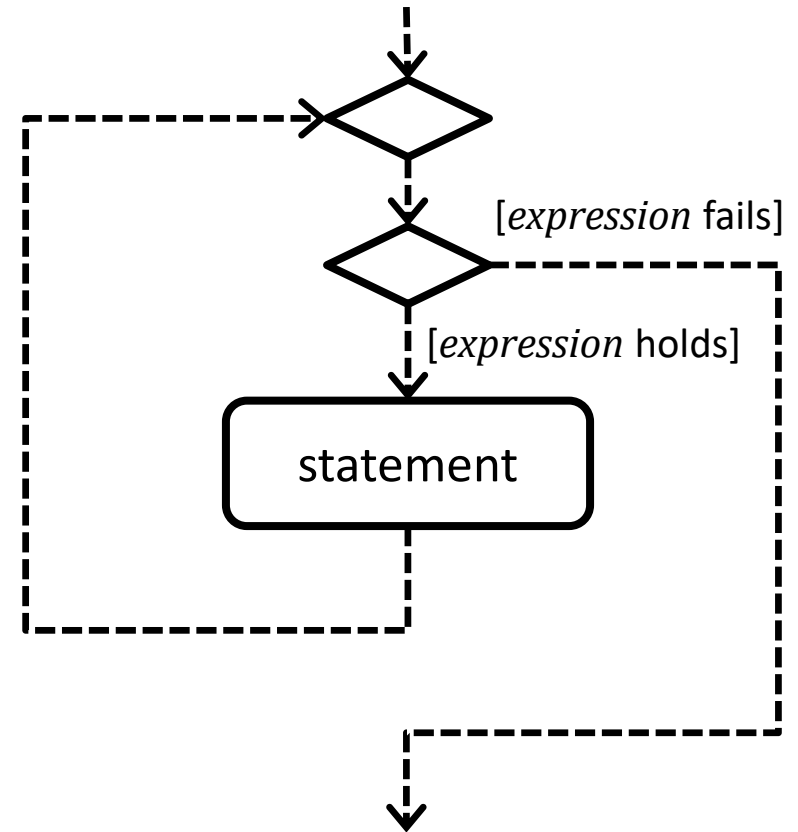
# Kontrollfluss

```
if (<expression>)  
    <statement1>  
else  
    <statement2>
```



# Kontrollfluss

**while** (*<expression>*)  
*<statement>*

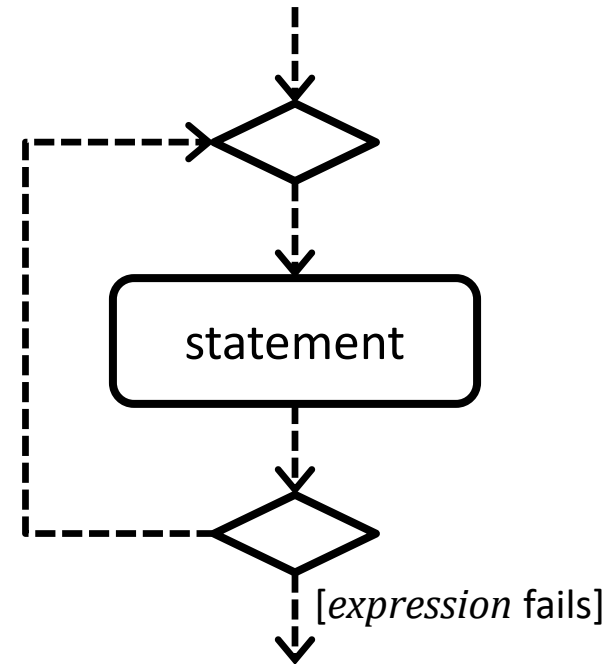


# Kontrollfluss

**do**

*<statement>*

**while** (*<expression>*)

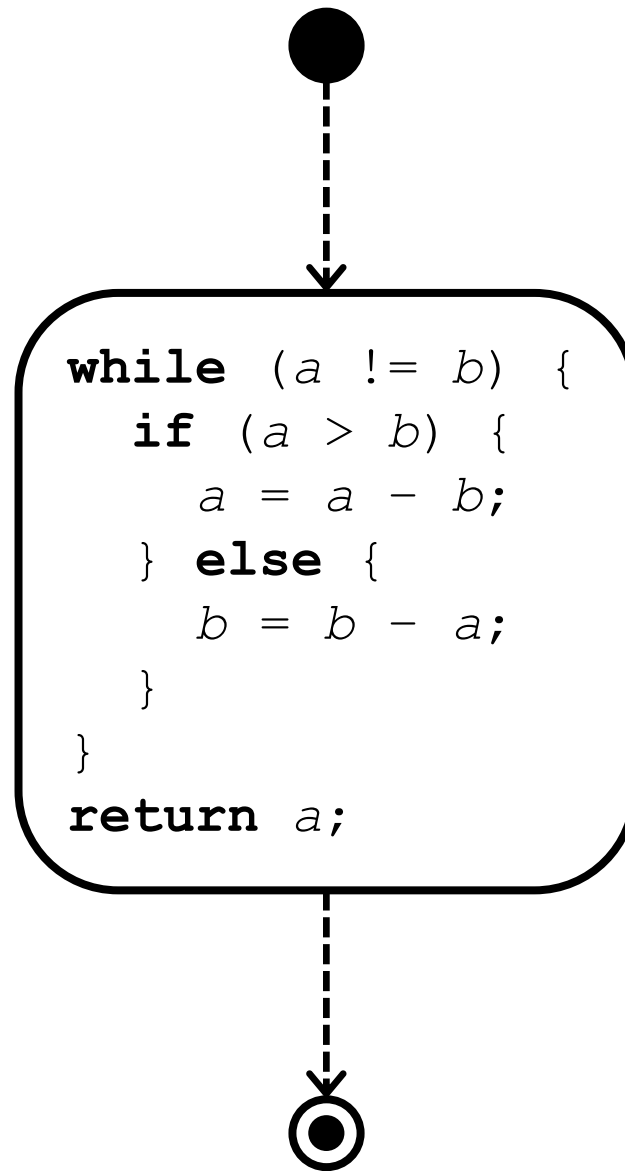




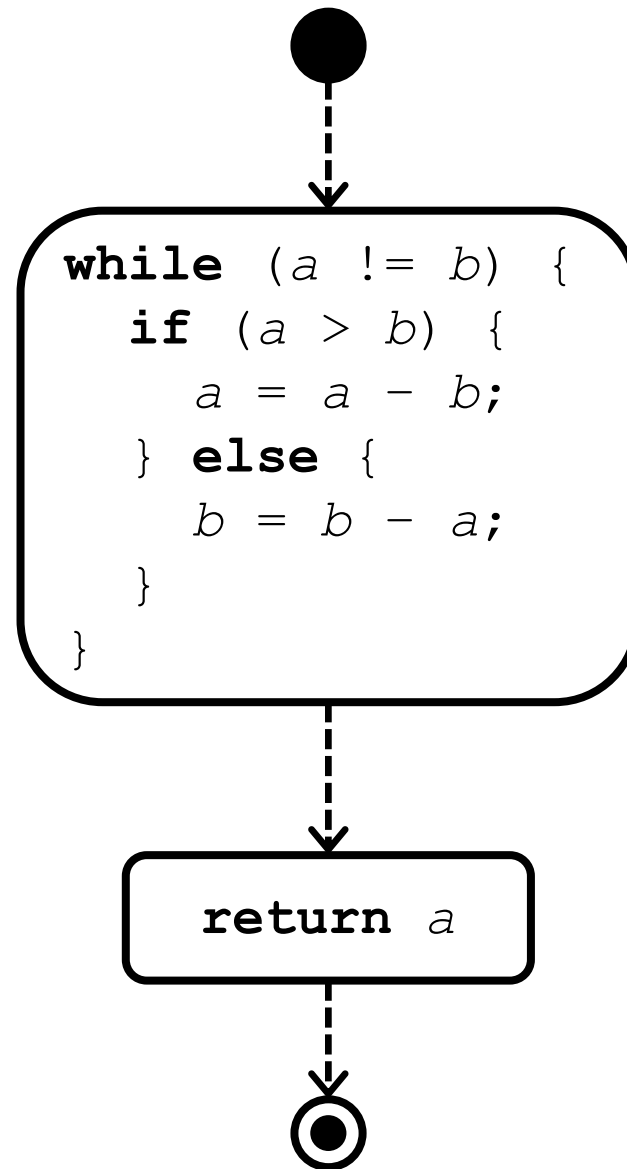
# Kontrollfluss – Beispiel

```
while (a != b) {  
    if (a > b) {  
        a = a - b;  
    } else {  
        b = b - a;  
    }  
}  
return a;
```

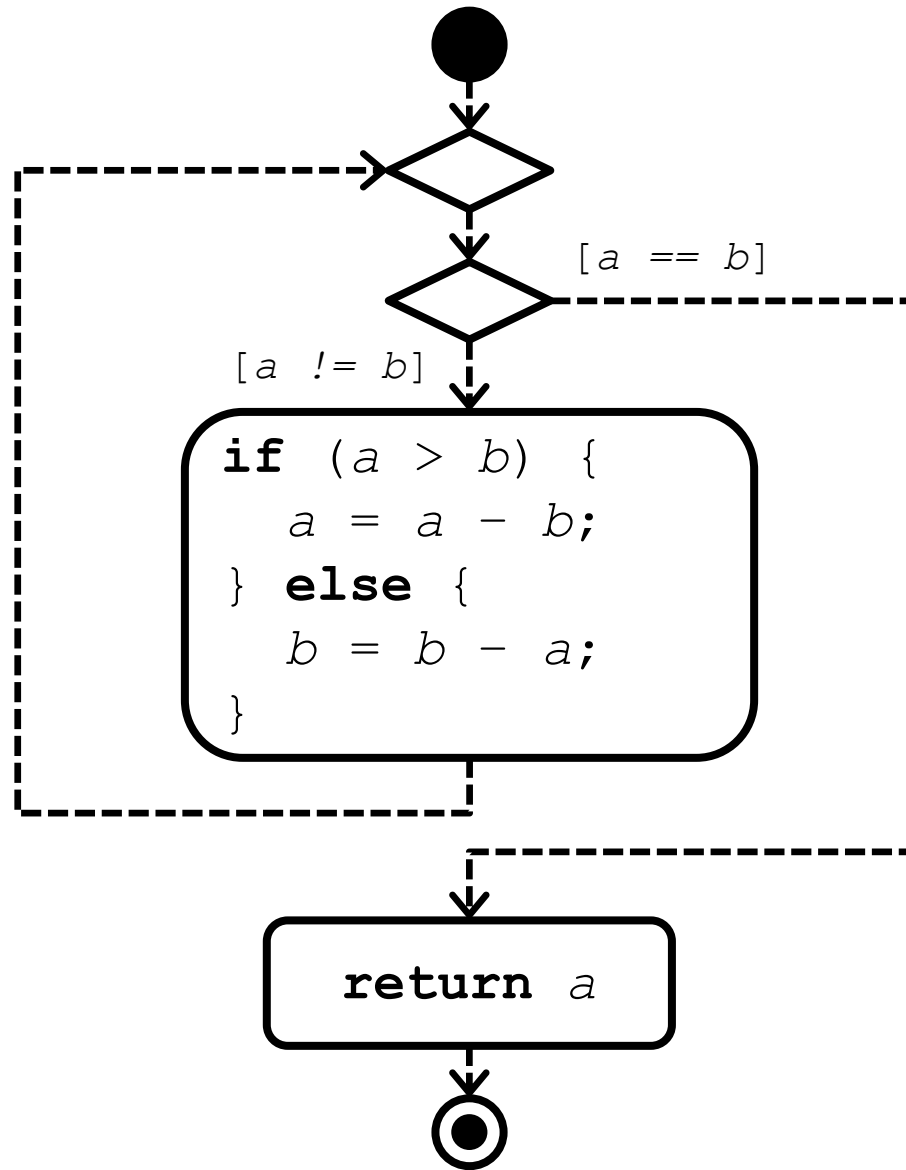
# Kontrollfluss – Beispiel



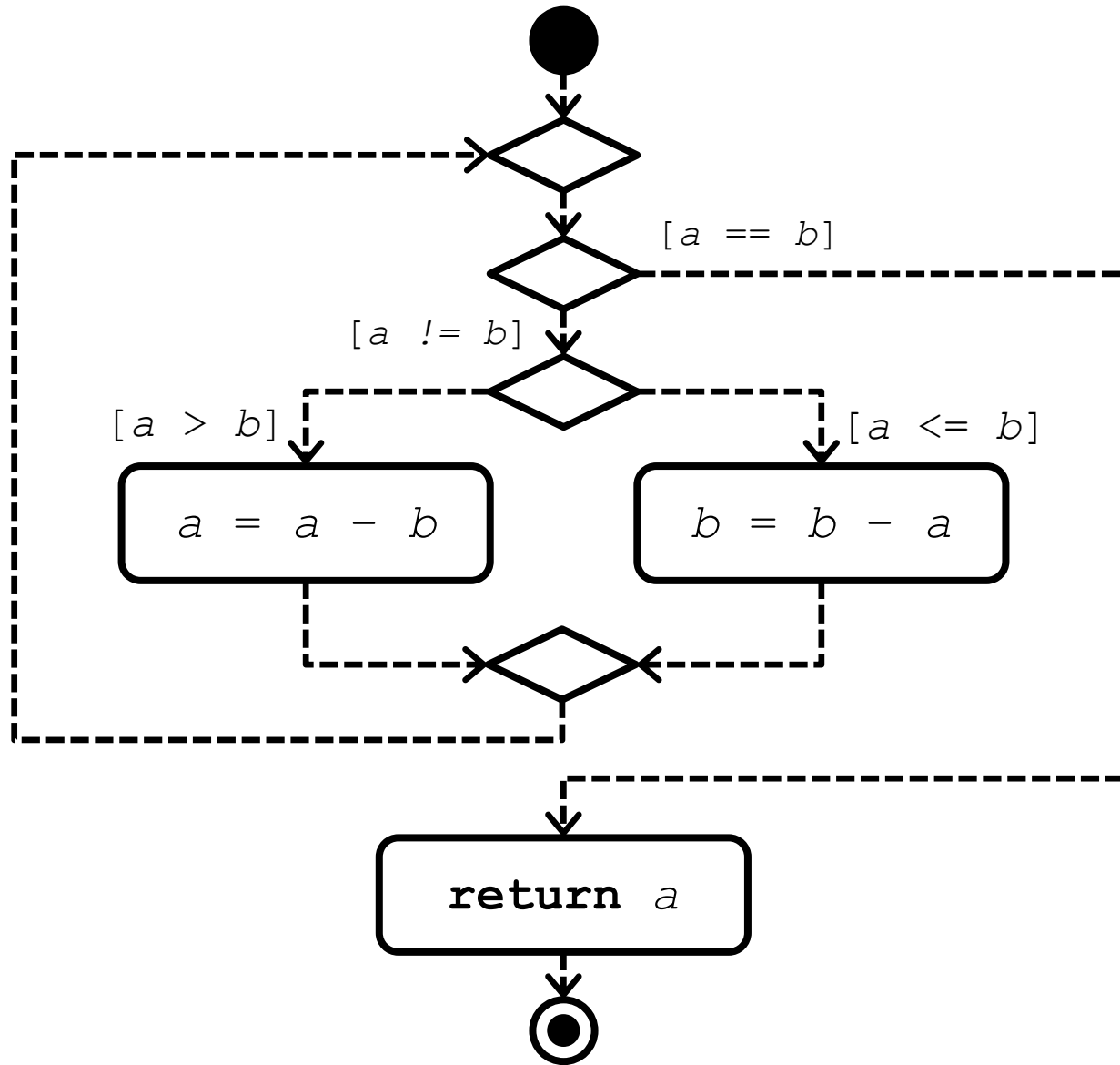
# Kontrollfluss – Beispiel



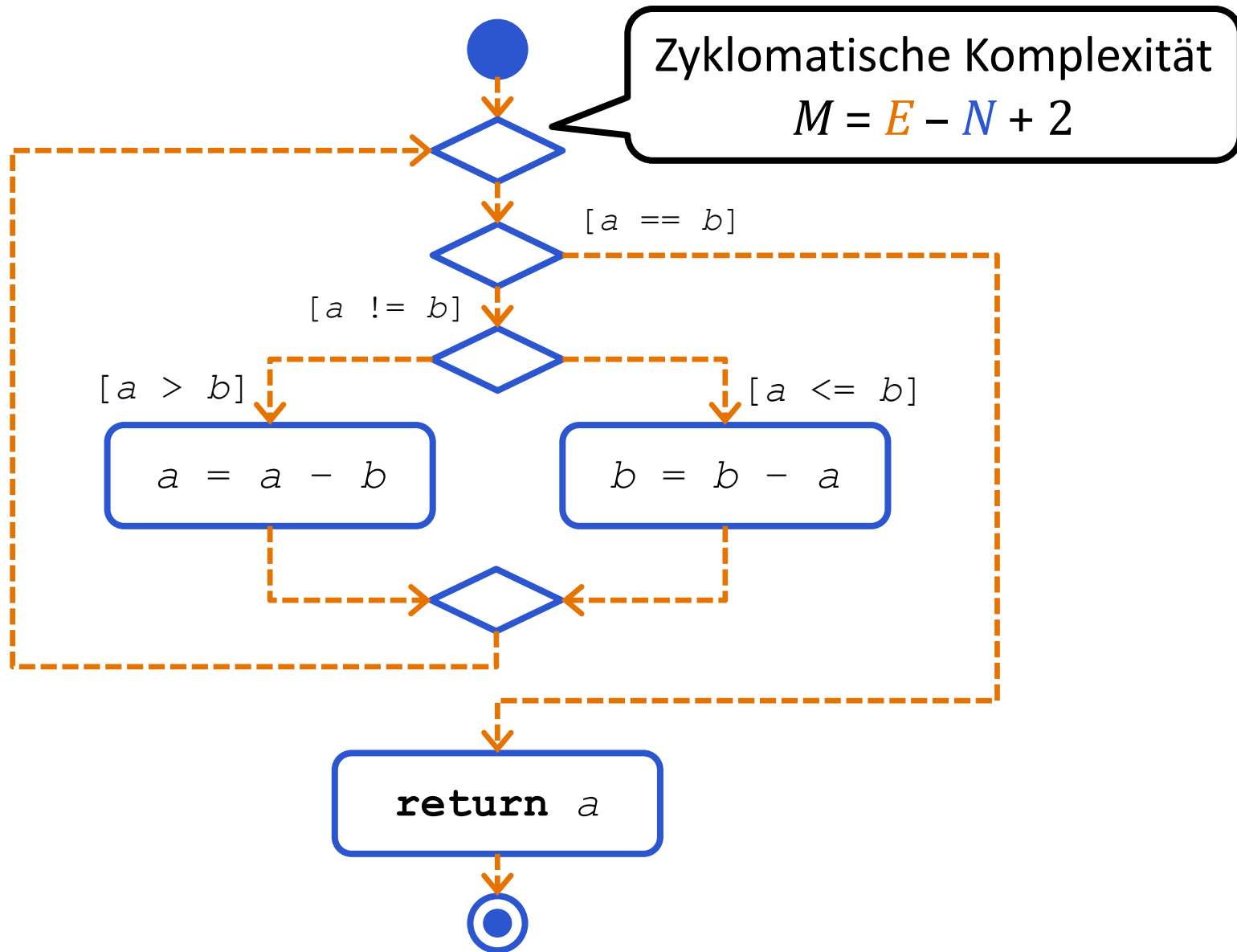
# Kontrollfluss – Beispiel



# Kontrollfluss – Beispiel



# Kontrollfluss – Komplexität



# Kontrollfluss – Rekursion

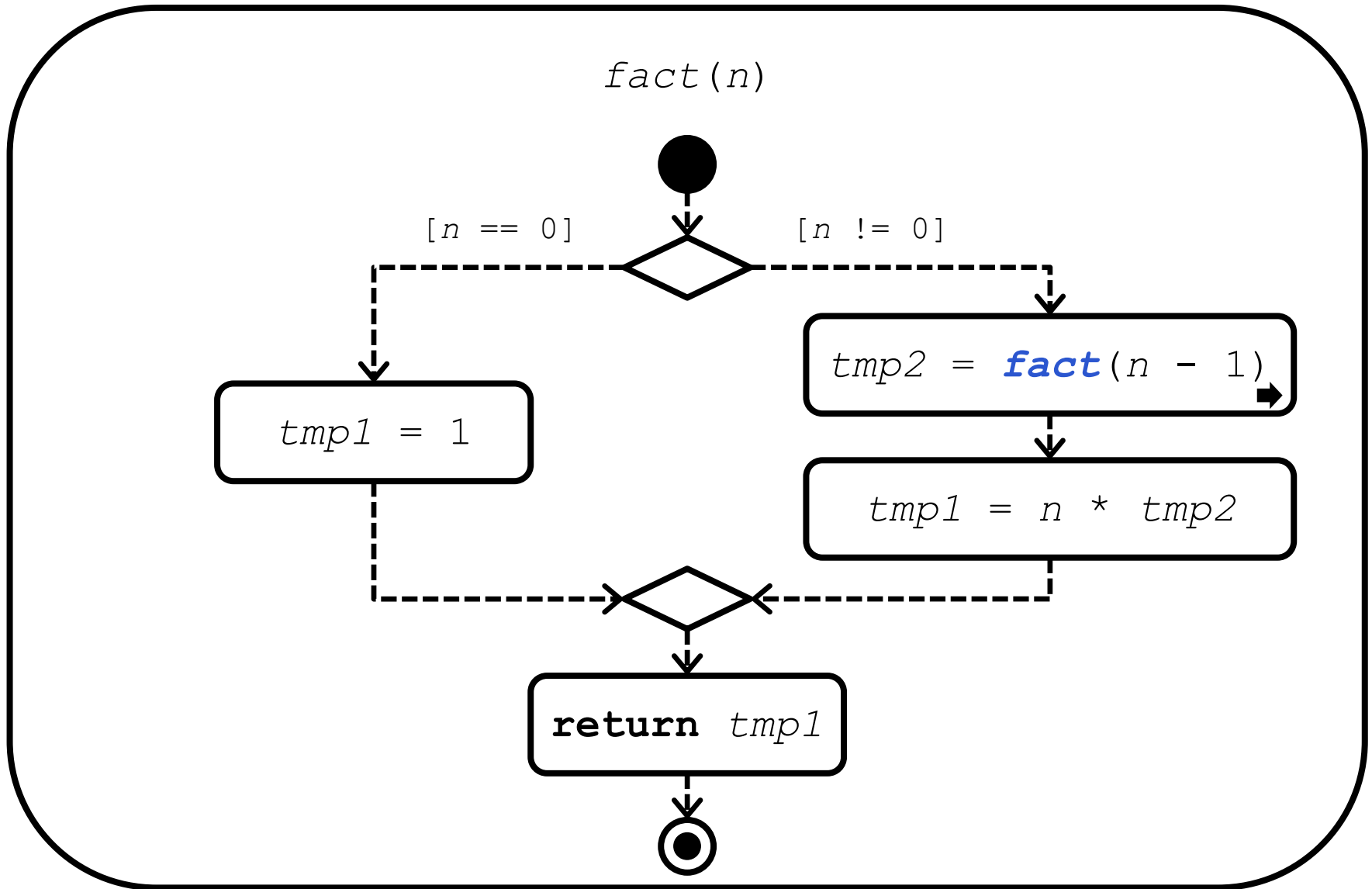
```
int fact(int n) {  
    return  
        (n == 0) ? 1 : n * fact(n - 1);  
}
```

# Kontrollfluss – Rekursion

```
int fact(int n) {  
    int tmp1;  
    if (n == 0) {  
        tmp1 = 1;  
    } else {  
        int tmp2 = fact(n - 1);  
        tmp1 = n * tmp2;  
    }  
    return tmp1;  
}
```



# Kontrollfluss – Rekursion



# Ausblick

- Es gibt Programmiersprachen, in denen es auch **fork** und **join** gibt
  - z.B. Cilk
- Was wenn die Sprache sie nicht unterstützt?
  - Die Ausführungsumgebung (OS) kann doch
    - Verzeichnisanrufe anstatt Sprachelemente

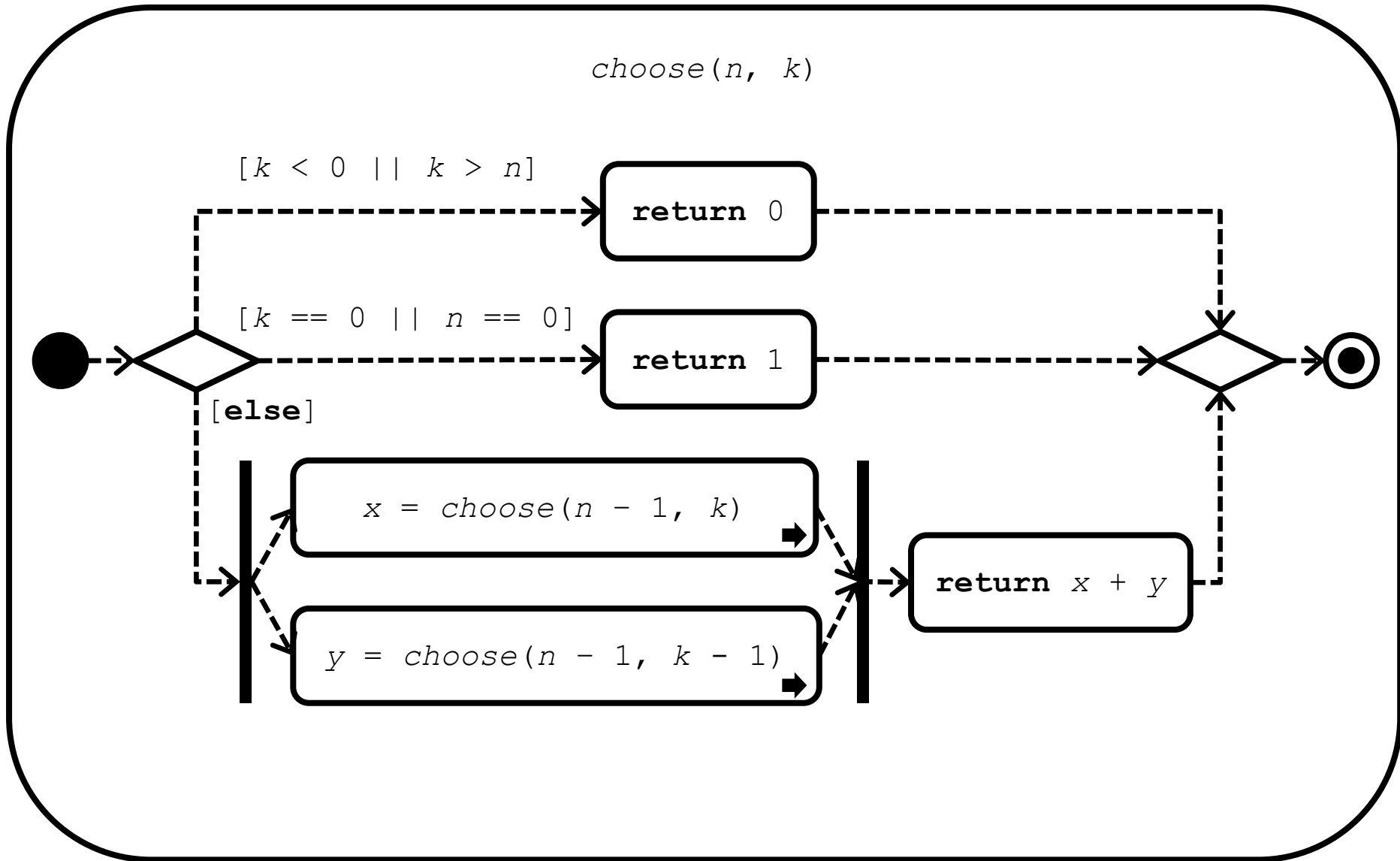
# Beispiel: $n$ über $k$

```
int choose(int n, int k) {  
    if (k < 0 || k > n) {  
        return 0;  
    } else if (k == 0 && n == 0) {  
        return 1;  
    } else {  
        int x = spawn choose(n - 1, k);  
        int y = spawn choose(n - 1, k - 1);  
        sync;  
        return x + y;  
    }  
}
```

$$\binom{0}{0} = 1$$

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

# Beispiel: $n$ über $k$



Wiederholung

Ziel der  
Prozessmodellierung

Prozess-  
modelle

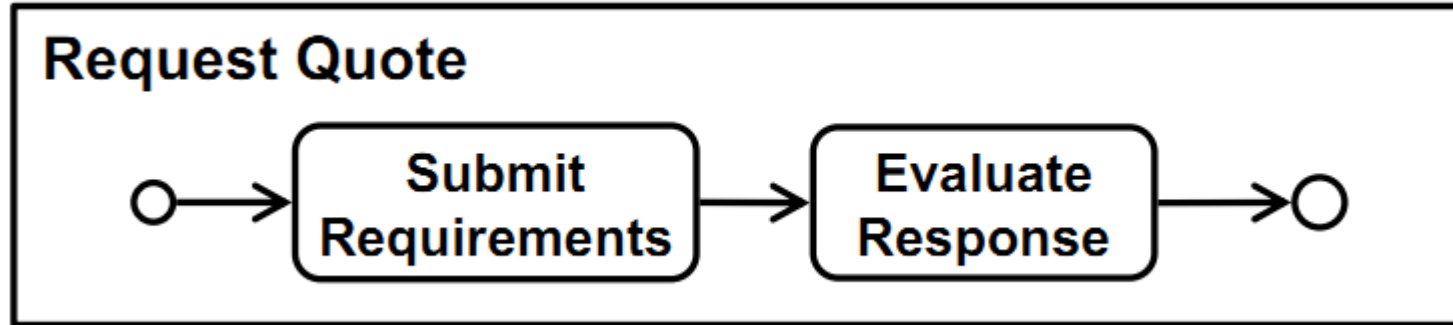
Kontroll-  
fluss

Verwirklichung

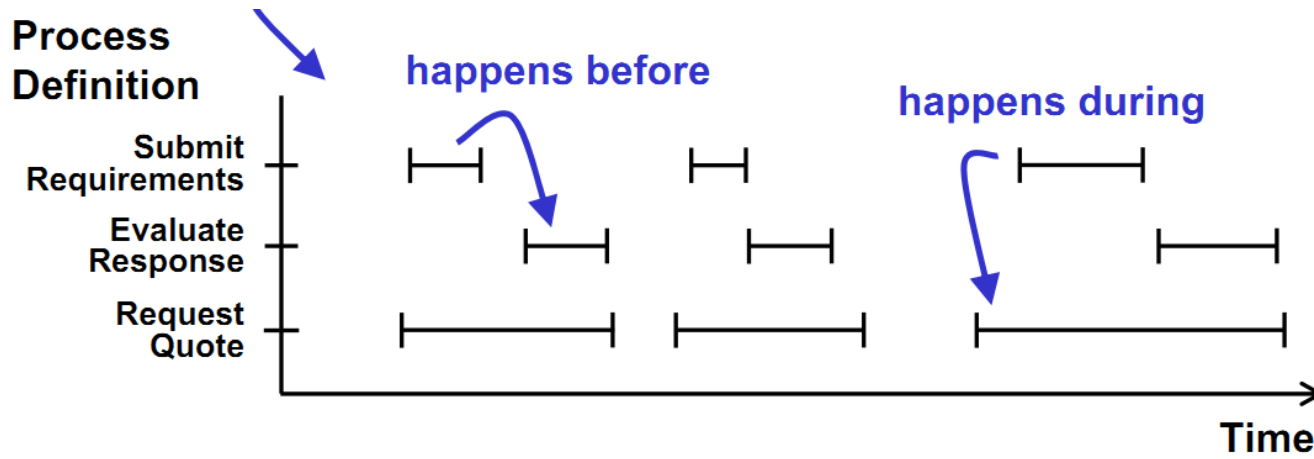
# AUSFÜHRUNG DER PROZESSE

# Semantik der Prozesse

- Aus der Sicht der Modellierung

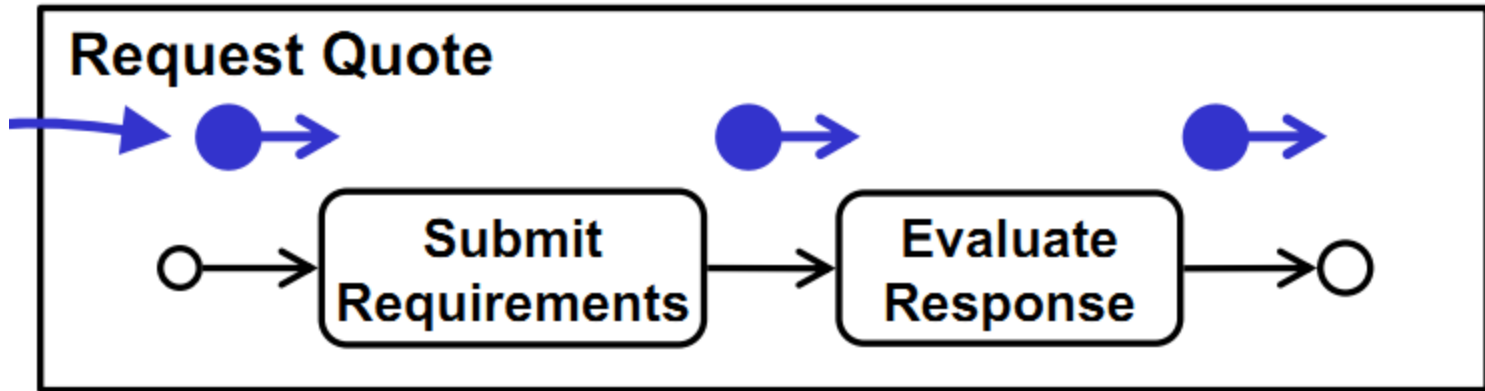


- Erwartete Abläufe



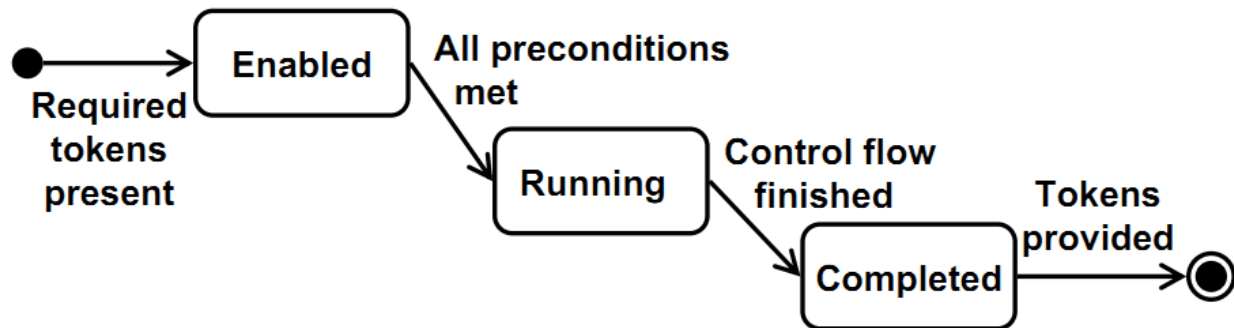
# Ausführung der Prozesse

## ■ Tokenfluss



## ■ Zustände des Prozesses

### State Machine



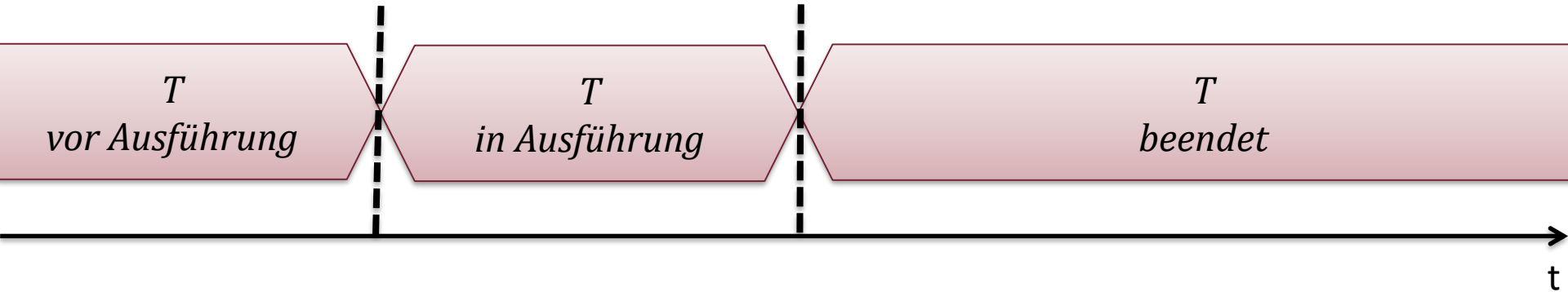
# Zustände der elementaren Aktivitäten

- Aktivität  $T$



Anfang der Ausführung

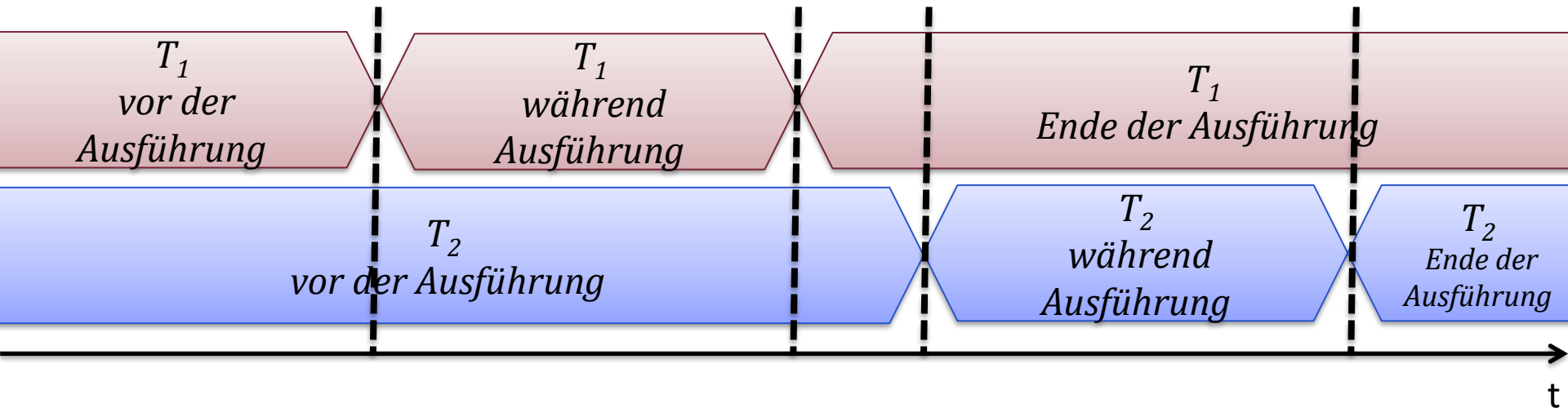
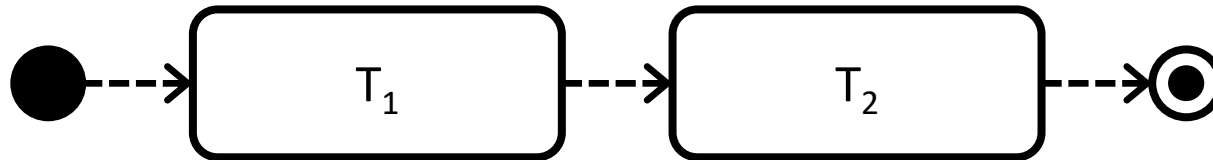
Ende der Ausführung





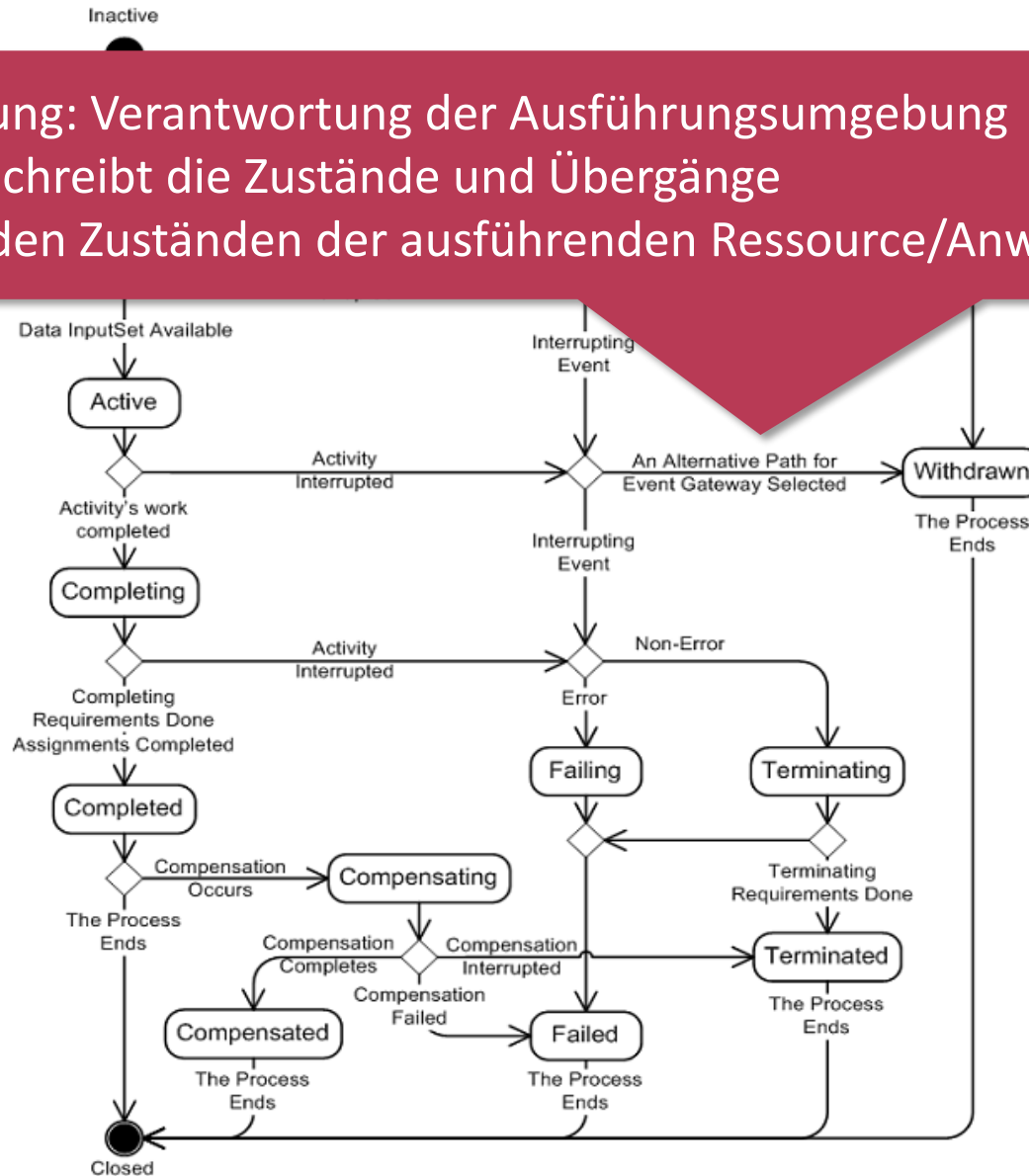
# Zustände der zusammengesetzten Prozessen

- Prozess  $T = T_1; T_2$



# Verfein. Zustandsmaschine der Aktivitäten

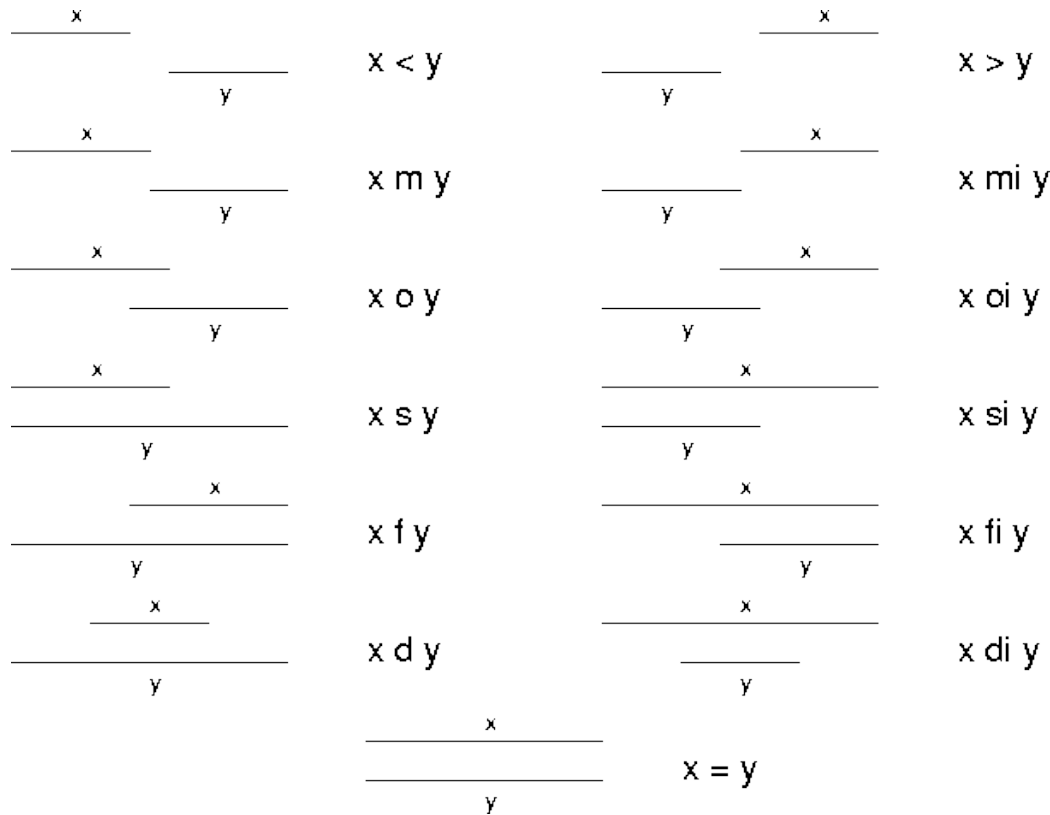
- Ihre Verwaltung: Verantwortung der Ausführungsumgebung
- Standard beschreibt die Zustände und Übergänge
- Nicht gleich den Zuständen der ausführenden Ressource/Anwendung



# Ausblick: Die Mathematik dahinter

- Allen-Intervallumlogik (1983)

- z.B. gut brauchbar beim Testen



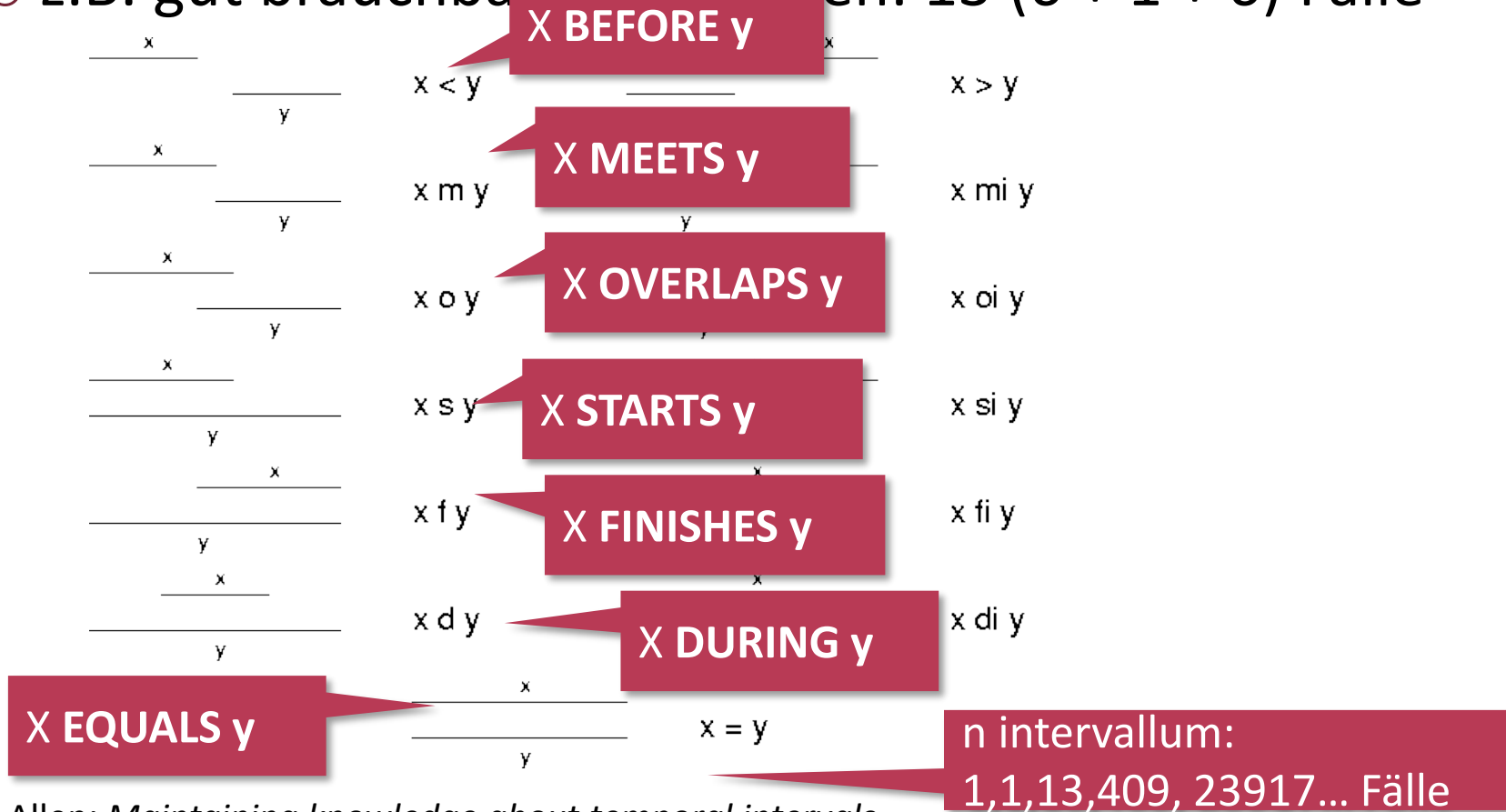
James F. Allen: *Maintaining knowledge about temporal intervals.*

In: *Communications of the ACM.* 26 November 1983. ACM Press. pp. 832–843, ISSN 0001-0782

# Ausblick: Die Mathematik dahinter

## ■ Allen-Intervallumlogik (1983)

- z.B. gut brauchbare Basis-Terme: 13 (6 + 1 + 6) Fälle



James F. Allen: *Maintaining knowledge about temporal intervals.*

In: *Communications of the ACM*. 26 November 1983. ACM Press. pp. 832–843, ISSN 0001-0782

# Was kann überprüft werden?

- Die Ausführung entspricht dem Prozess nicht
  - Entspricht sie den Anforderungen (Reihenfolge, Unabhängigkeit)?
- Was kann der Prozess hinter dem System sein?
  - Workflow mining
- Bei flexibleren Umgebungen
  - Wenn Aktivitäten ausgelassen werden können
  - Werden die Anforderungen erfüllt?
- Werkzeug: formelle Methoden
  - Logik, Petri-Netze, Modellüberprüfung, etc.

Wiederholung

Ziel der  
Prozessmodellierung

Prozess-  
modelle

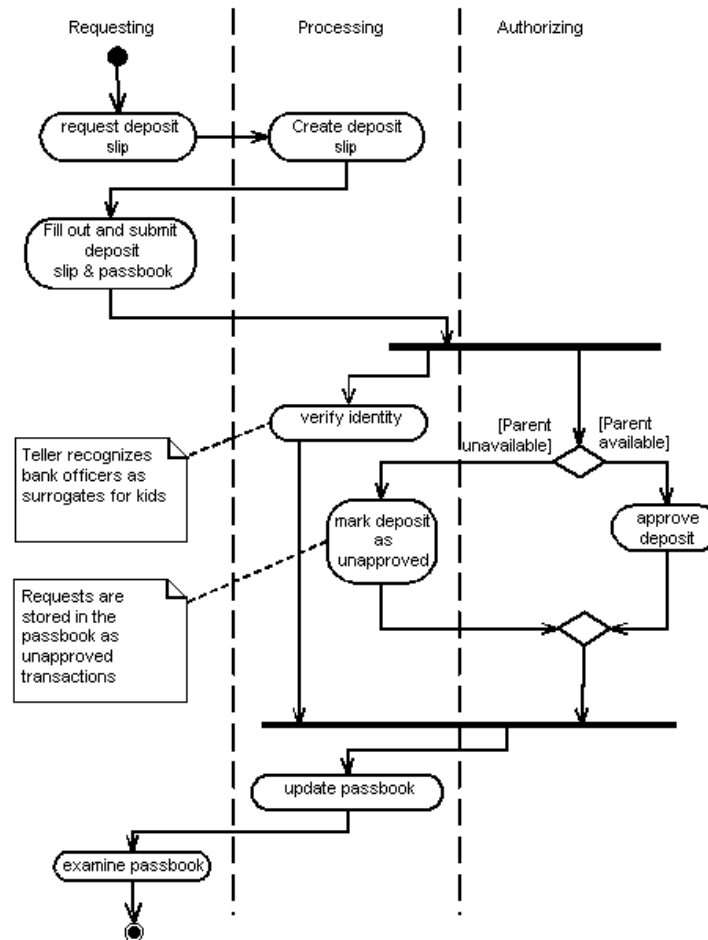
Kontroll-  
fluss

Verwirklichung

# GESCHÄFTSPROZESSMODELLE IN DER PRAXIS

# UML Activity Diagram

- Standardnotation mit Erweiterungen
  - Mehr dazu in SW-Technologien im 3. Semester

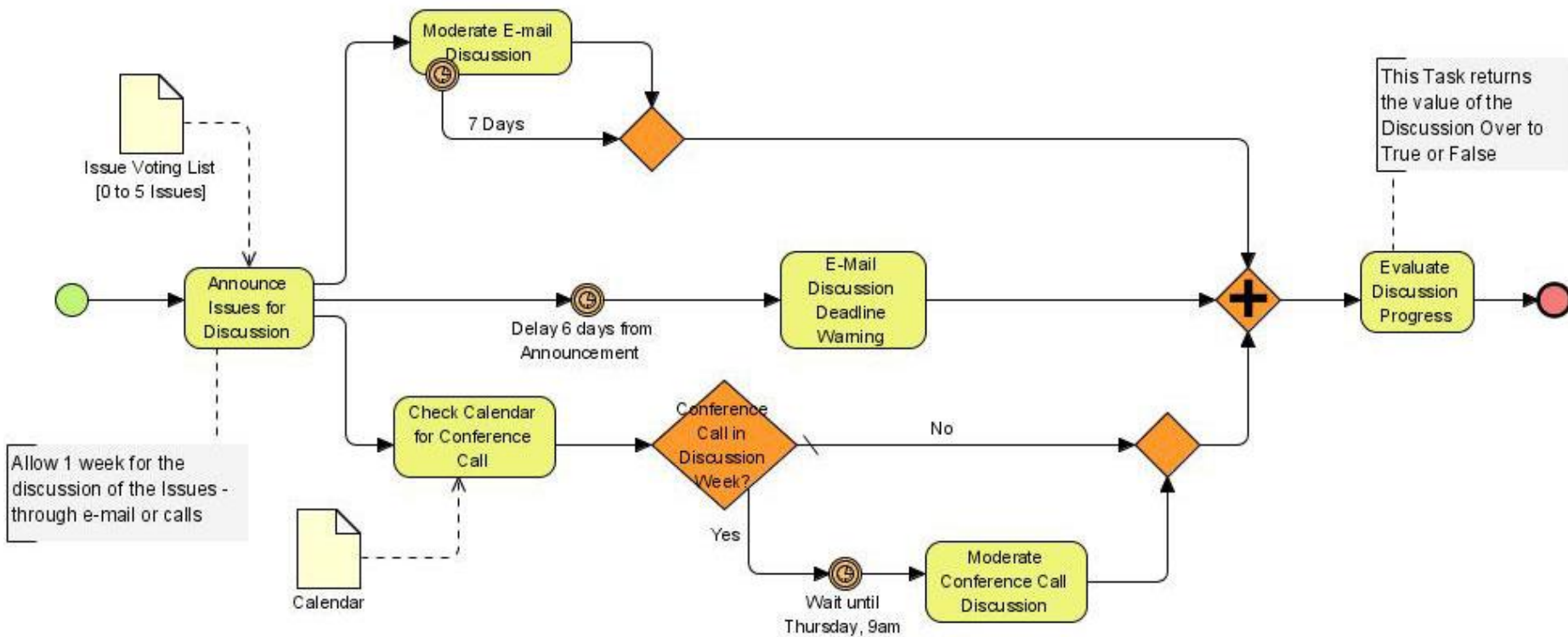


# Business Process Modeling Notation (BPMN)

- Business Process Management Initiative (BPMI)
  - Mai 2004: BPMN 1.0 Spezifikation
  - 2011: endgültiger BPMN 2.0
- Ziele
  - Verständlichkeit
    - Für Benutzer
  - Geschäftsanalysten
    - Initialer Prozessplan
  - Entwickler
    - Implementation
    - Internes Modell für automatische Codegenerierung
    - BPEL4WS
  - Verbraucher (Überwachung, Management der Prozesse)



# Beispiel: BPMN



# Datenfluss

Ereignisse

Zustandsübergänge  
Ursachen/Auswirkungen  
Ereignistypen:  
Start, Intermediate, End



Aktivitäten

Elementar/Zusammengesetzt  
Tasks/Subprozesse



Anschlussstellen/  
Kopplungen

Sequenzen  
Konvergenz/Divergenz  
AND, OR, XOR, ...



# Verbindungen

Sequenzen Reihenfolge der  
Aktivitäten in einem  
Prozess



Nachrichten Informationsaustausch zw.  
zwei unabhängigen  
Prozessteilnehmern



Assoziationen Zuordnung von Daten,  
Texten, usw.



# Dekomposition

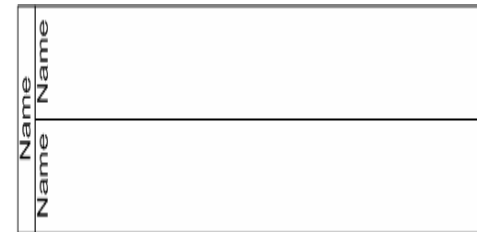
Pool

Darstellung der  
verschiedenen  
Teilnehmer



Bahn/  
Swimlane

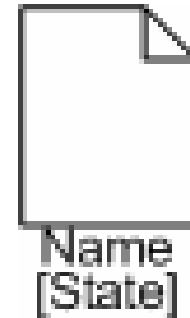
Gruppierung der  
Aktivitäten



# Produkte

Datenobjekte

Symbolisches  
Token (Artefacts)



Gruppen

Gruppierung der  
Aktivitäten

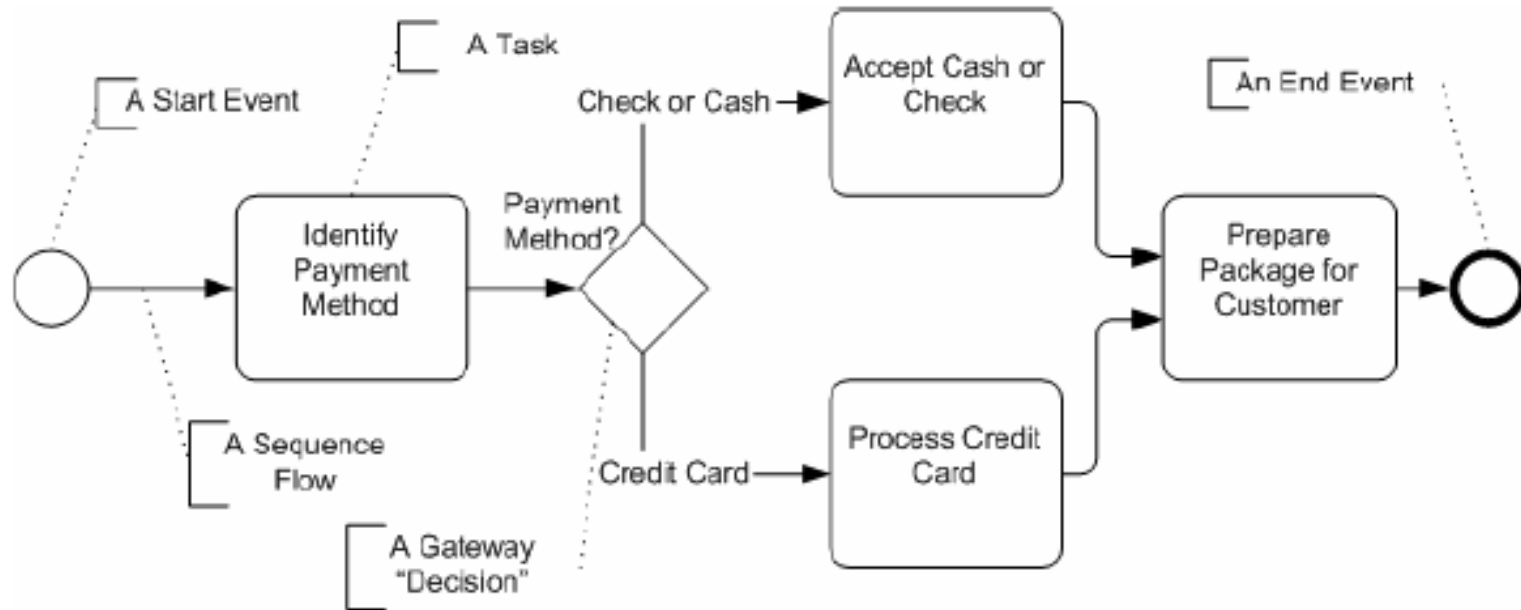


Annotationen

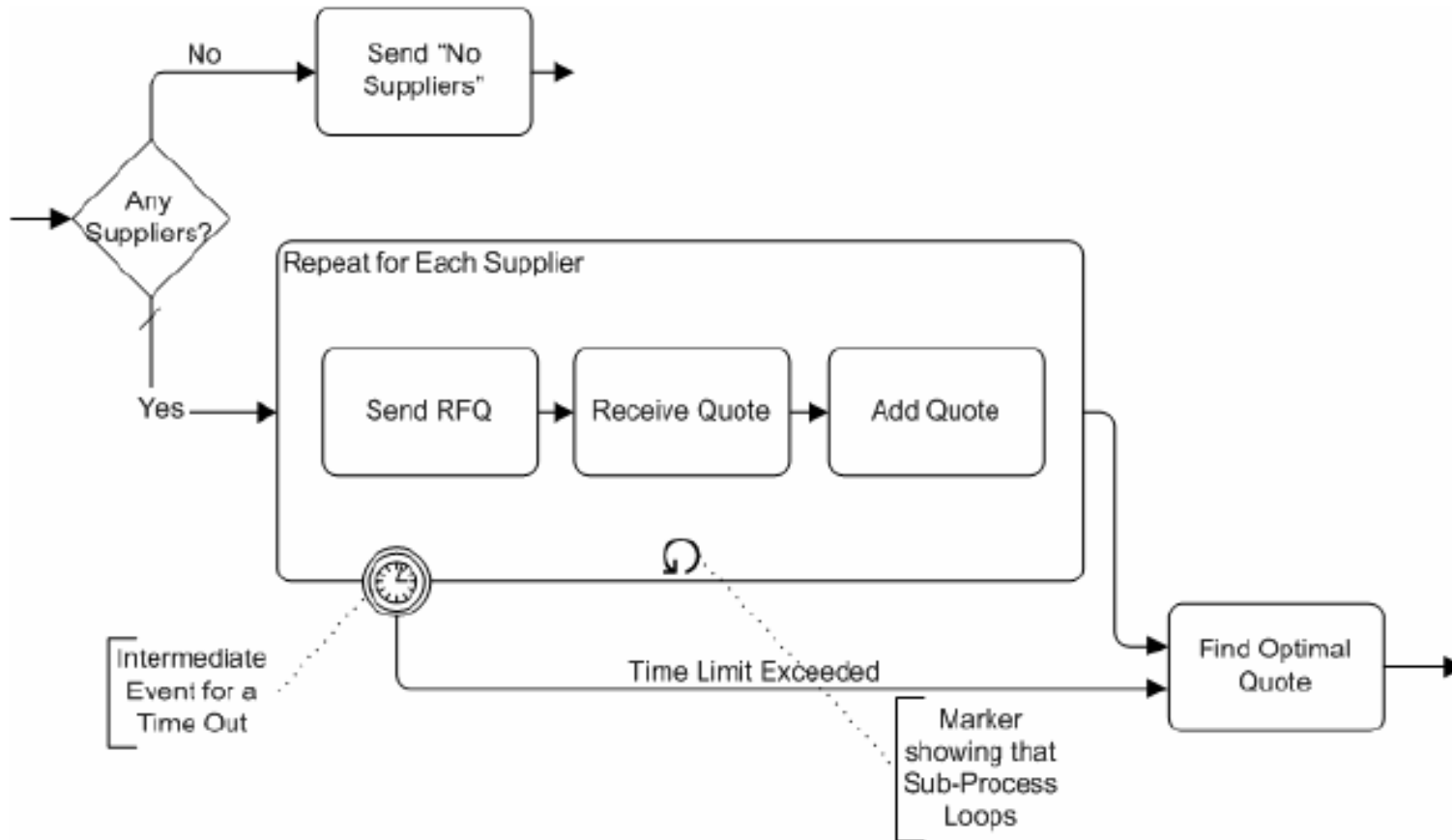
Ergänzende textuelle  
Information  
(Kommentar)



# Beispiel



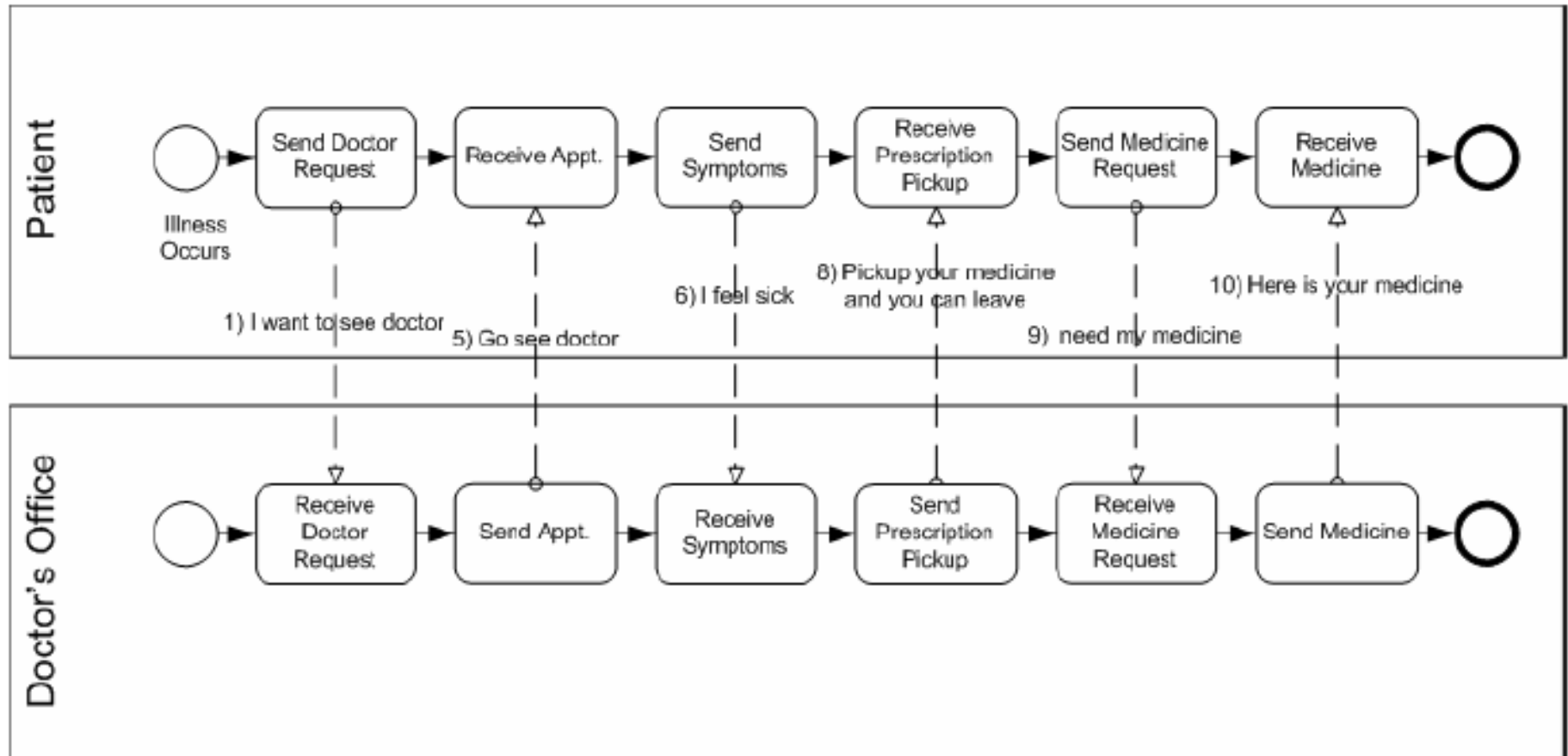
# Beispiel – Hierarchische Modellierung



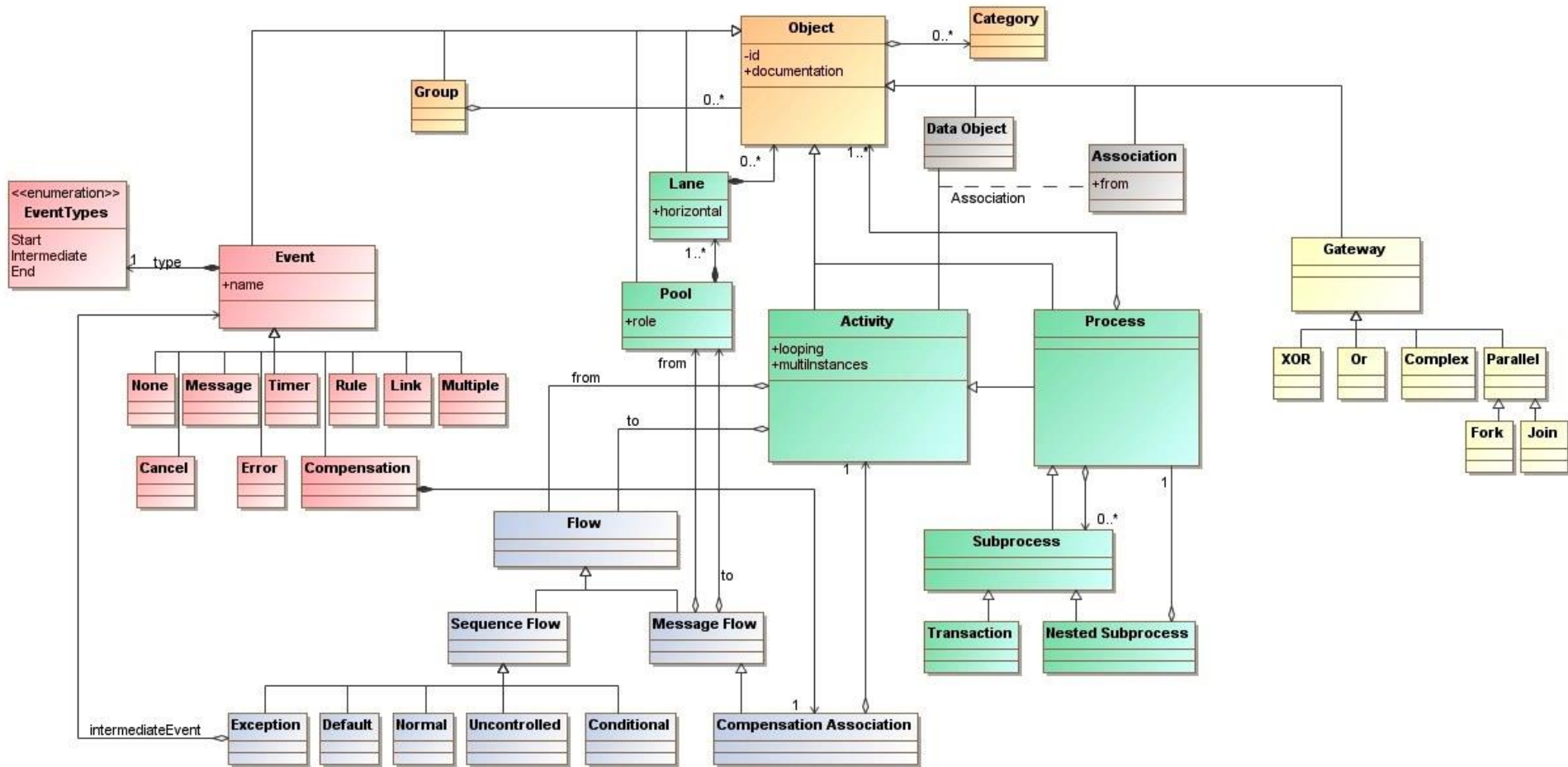




# Beispiel – Kooperierende (Sub-)Prozesse

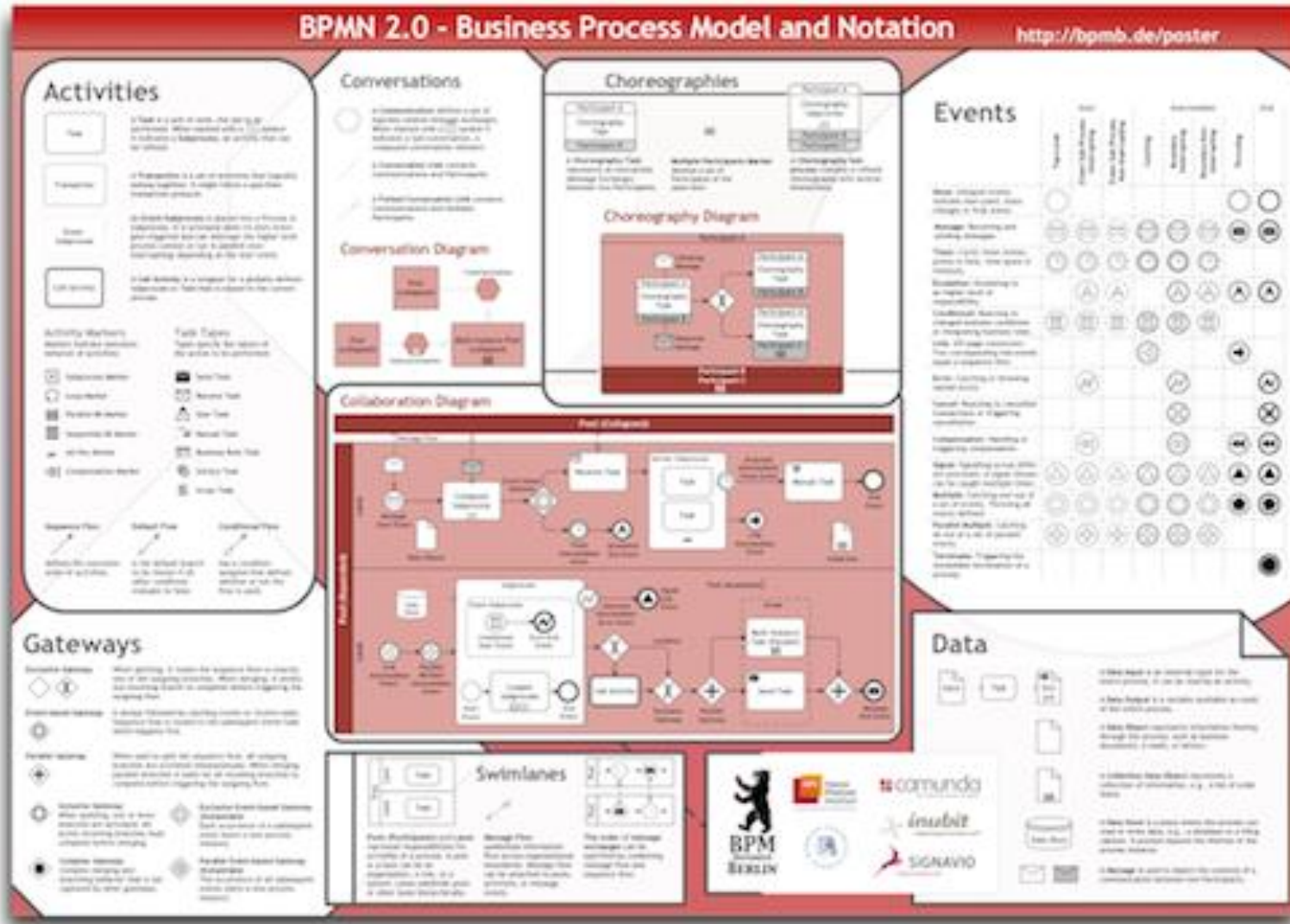


# BPMN Metamodel (vereinfacht)



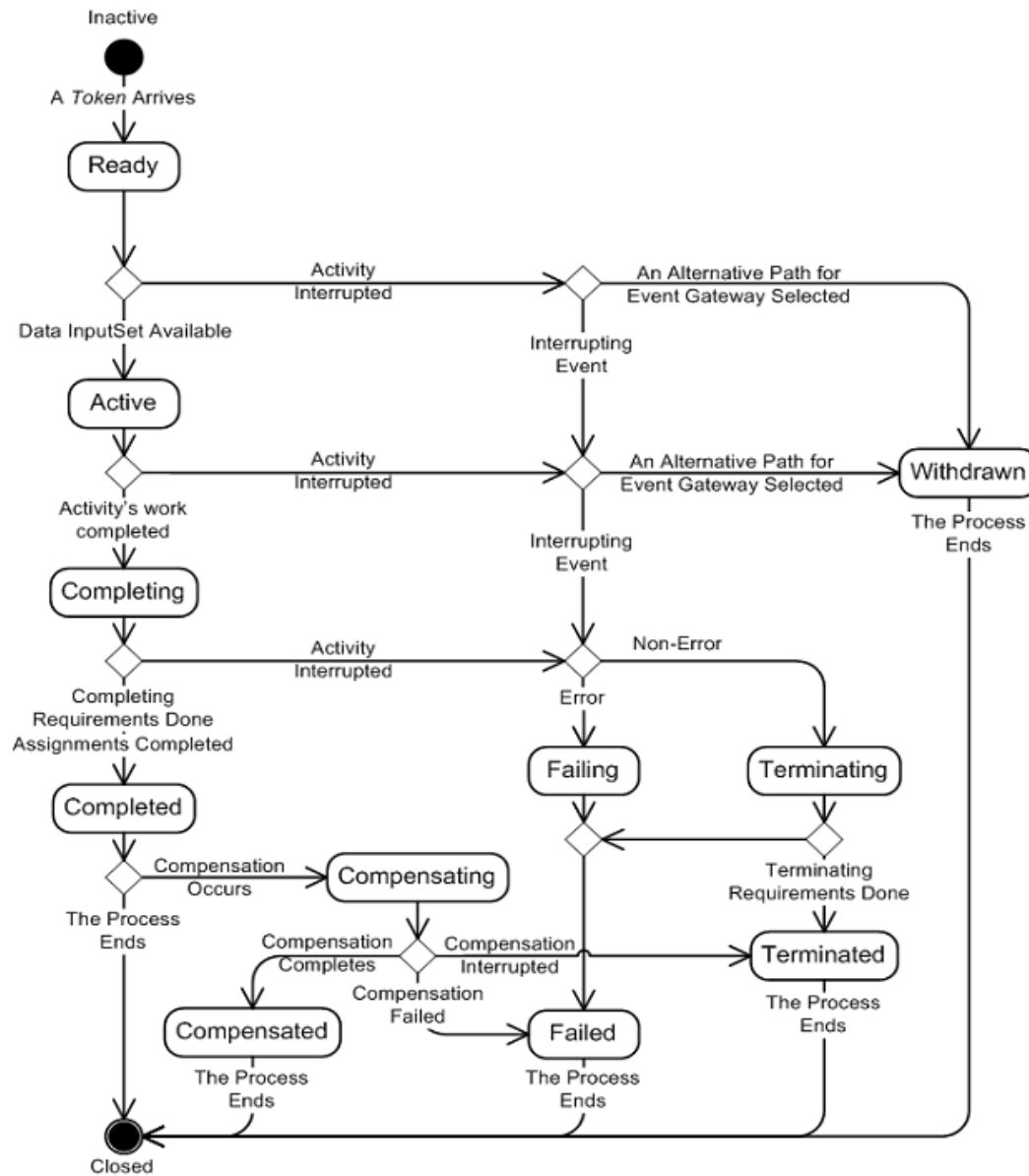
Source: <http://www.wsper.org/>

# BPMN Sprachelemente



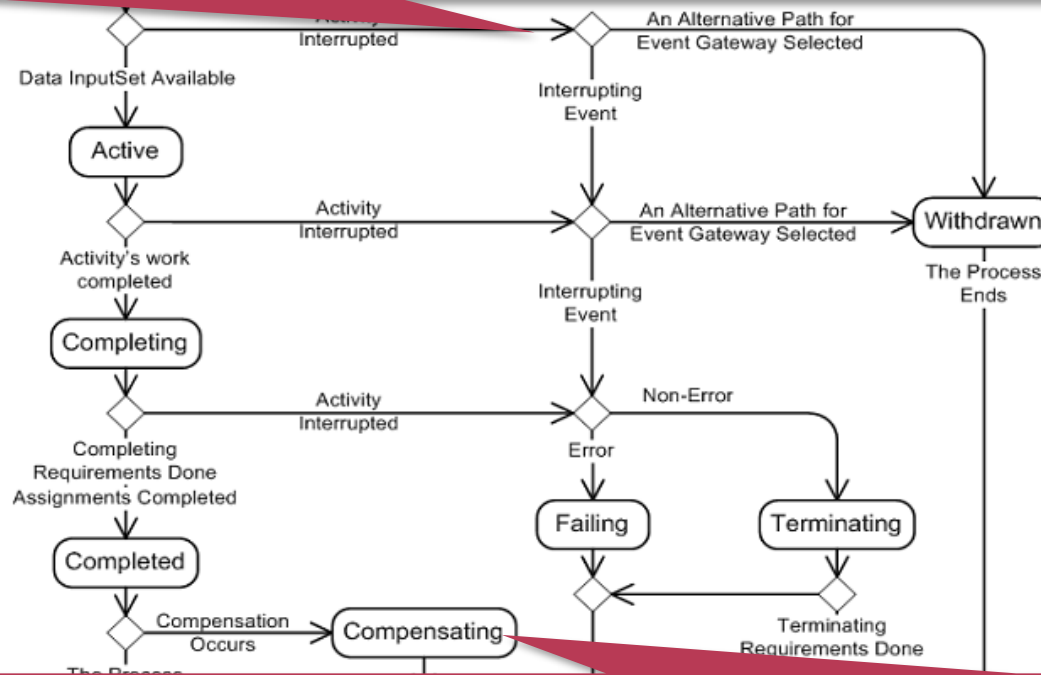
Source: <http://www.bpmb.de>

# Verfein. Zustandsmaschine der Aktivitäten



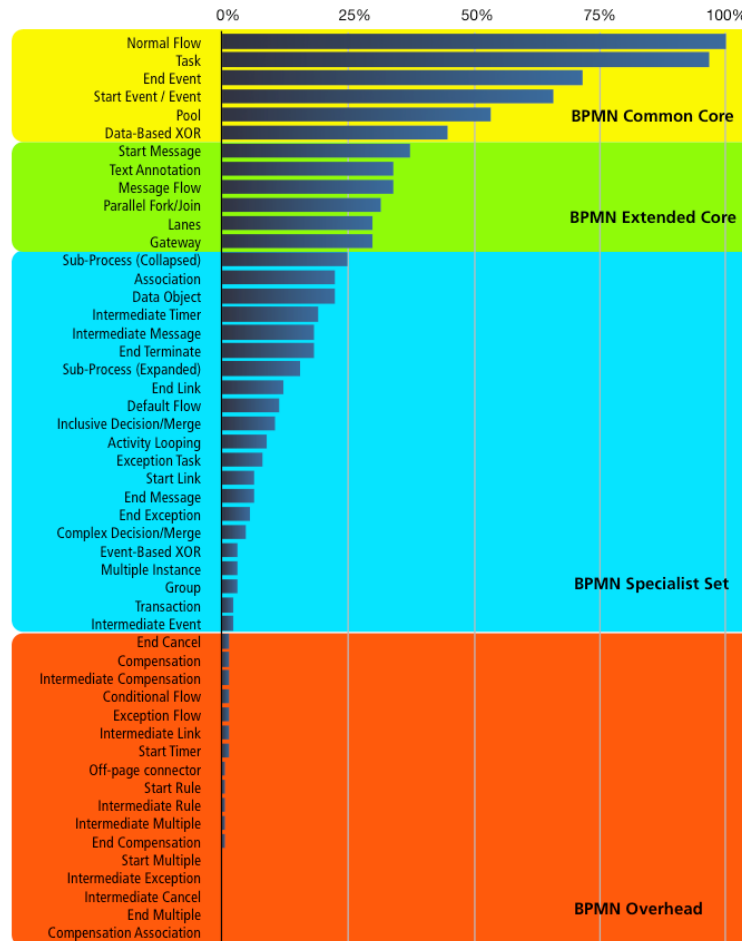
# Verfein. Zustandsmaschine der Aktivitäten

- Aktivität kann ...
  - unterbrochen werden, zurückgerufen werden,
  - fehlerhaft ablaufen



- Aufgabe des Entwicklers:
  - Was bedeutet das Zurückrufen eines abgeschickten Emails?

# „A statistics...”



Source: Process Modelling. What Really Matters  
 Keynote of Michael Rosemann @ UNISCON2009 conference

# Herausforderungen

- Formulierung des fachspezifischen Wissens
  - Verzeichnisse, Schablonen/templates
  - „Web2.0“
  - Effiziente Modellierung
- Konsistenz der Modelle
  - Statische Analyse: ~200 Fragen (BPEL2 Standard)
  - Abhängigkeiten der Prozessmodelle und anderen Modelle
    - Zustandmaschinen, ...
- Installation, Ressource-Konfiguration, ....

# BPMN Werkzeuge

- jBPM Designer
  - Eclipse BPMN
  - Tibco Business Studio
  - **IBM Websphere Business Modeler**
  - Intalio Designer
  - BPMN Composer
  - BPMN Designer
  - Bonita Open Solution
  - Adonis
  - Activiti
  - Obeo Designer
- + általános modellező eszközök



# Literatur

- <http://www.sdn.sap.com/irj/scn/index?rid=/library/uuid/609cb540-3ca6-2a10-60a7-dc470a9b7adf>
- <http://community.intalio.com/tutorials/exception-handling.html>
- <http://www.conradbock.org/bock-bpmn-2-business-process-semantic-web.pdf>
- Stephen A. White (IBM): *Introduction to BPMN*

# Ausführung: “workflow engine”

- Allgemein benutzt für ausführbare Prozesse
- Verwaltung des Lebenszyklus der Prozesse
  - Verwaltung der Prozessvorlagen
  - Instanziierung, Verwaltung der Instanzdaten
- Versionierung, online Aktualisierung
- API für Elemente, die eingebettet/referenziert werden
  - REST, WS, EJB...
- Verwaltung der Geschäftsregeln (-entscheidungen)
- Menschlicher Mitarbeit (human tasks)
  - Präsentation in Webbrowser
  - Verwaltung der Zugriffsrechte