OST
Ostschweizer
Fachhochschule

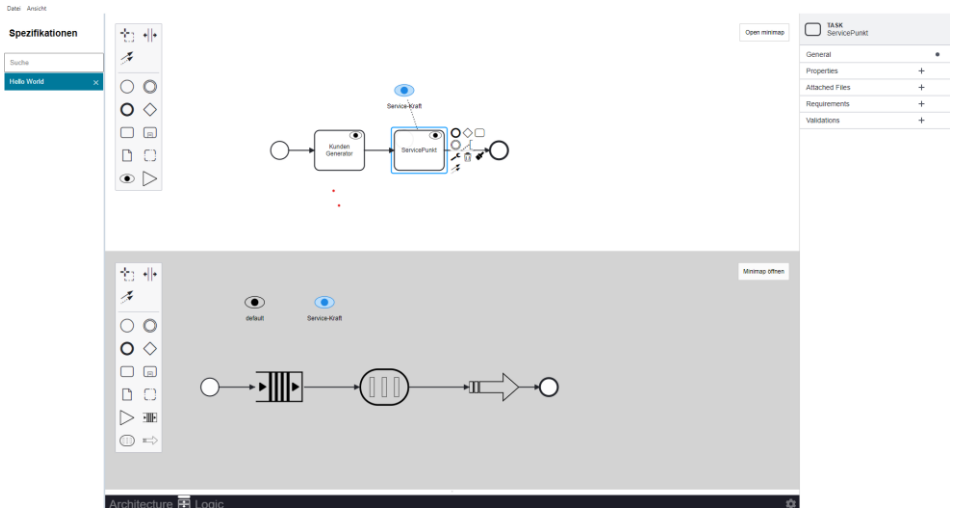
Semiformale Modellierung von Simulationsmodellen mit der simBPMN

Eine Einführung in 90 min

- **Andreas Rinkel**
  andreas.rinkel@ost.ch
- **Thomas Kehl**
  thomas.kehl@ost.ch
- **Heiderose Stein**
  heiderose.stein@unibw.de

Einführung in die simBPMN in drei Schritten

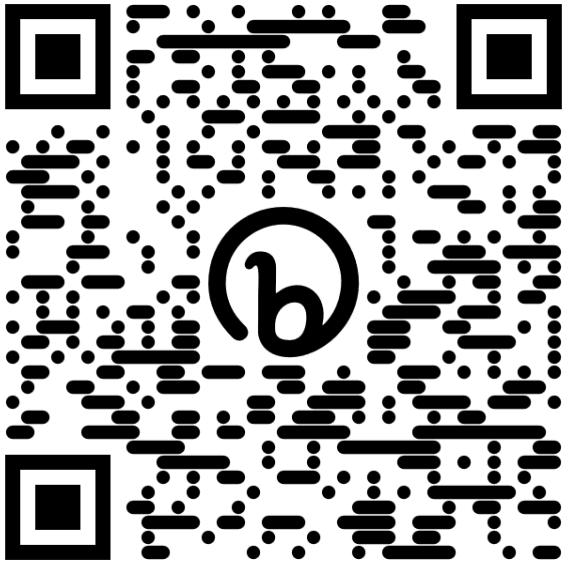
- **Theorie (45 min):**
  Sie erhalten einen Einblick in die formalen Aspekte der Beschreibungstechnik und deren Grundelemente.
- **Praxis (35 min):**
  Sie sehen eine Demo des Tools (15 min) und können selbst eine kleine Spezifikation erstellen (20 min) .
- **Ausblick, Diskussion und Fragen (10 min)**
  Sie hören und diskutieren, wie das Tool weiterentwickelt wird.



Zugriff auf die Unterlagen

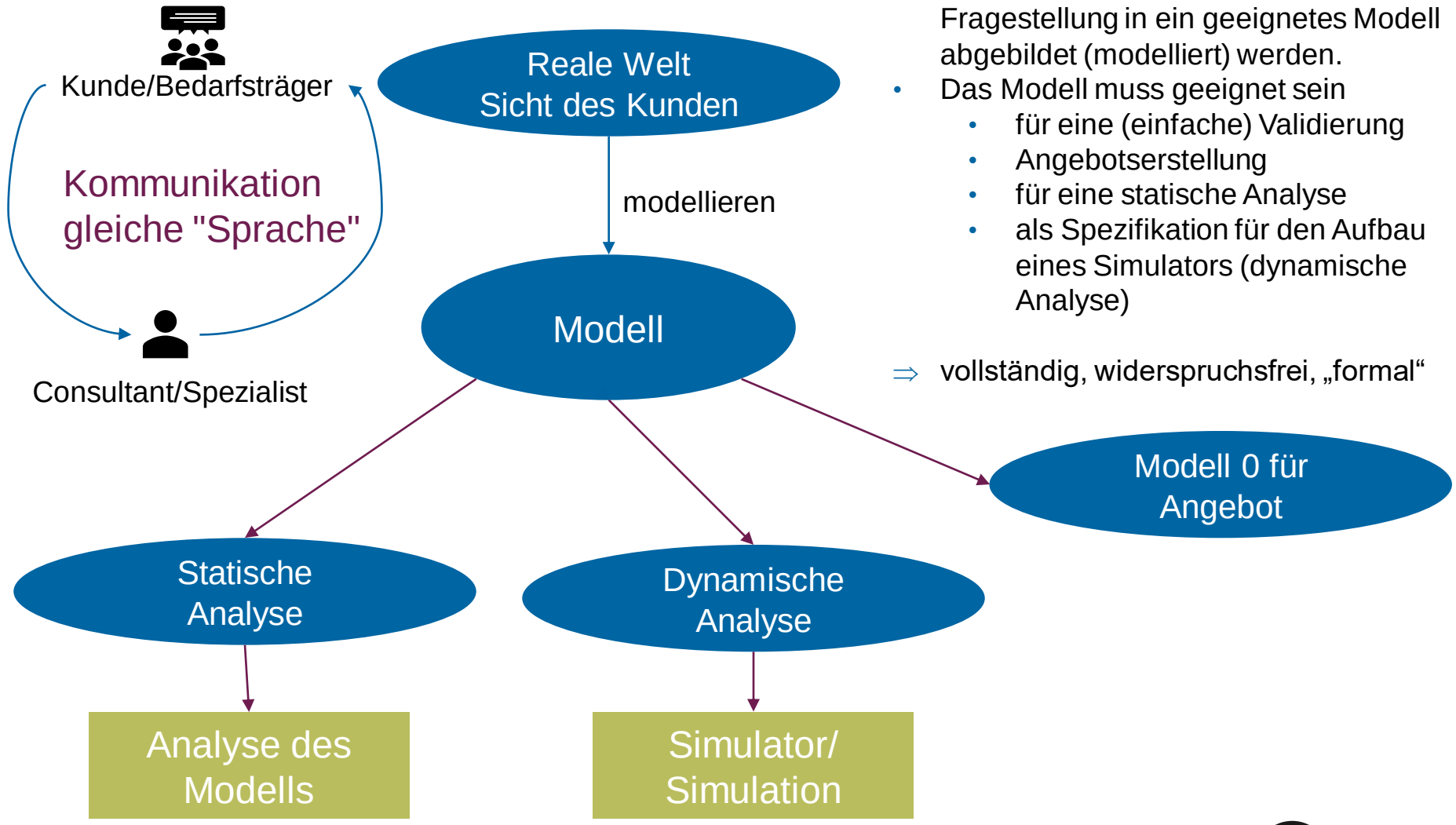
- **Die Folien**
- **Den simBPMN-Visualizer**
- **Das Beispiel: "Kaffeepause"**
- **Und nach der Veranstaltung ein Lösungsvorschlag zur Aufgabe**

https://bit.ly/simBPMN-Ilmenau oder



OST

Einführung in die simBPMN: Ausgangslage



Ausgangspunkt:

- Die reale Welt muss bzgl. der Fragestellung in ein geeignetes Modell abgebildet (modelliert) werden.
- Das Modell muss geeignet sein
  - für eine (einfache) Validierung
  - Angebotserstellung
  - für eine statische Analyse
  - als Spezifikation für den Aufbau eines Simulators (dynamische Analyse)

⇒ vollständig, widerspruchsfrei, „formal“

OST

Theorie: Anforderungen an eine einfache Beschreibungssprache I(II)

Funktionale Anforderungen

- **die Architektur des Systems und dessen Abgrenzung beschreiben mit**
  - System und Subsystem (Nesting)
  - Ressourcen
    - Ressource zum Betrieb (Kapazität)
    - Material (Bill of Material)
    - Waste (Co2)
  - Fluss der Entities durch das System
    - Transformation der Entities
- **die Systemlogik (zustandsabhängig)**
  - das logische Verhalten (Ablauf, Entscheidungen, Zustände, ..)
  - Nesting unterstützen
  - das temporale Verhalten

OST

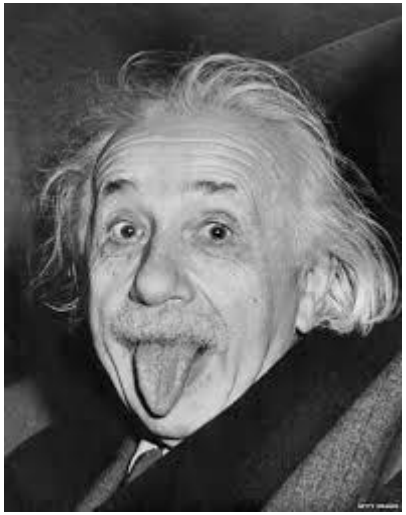
Theorie: Anforderungen an eine einfache Beschreibungssprache II (II)

Nicht-Funktionale Anforderungen

- die Sprache soll einen eingeschränkten Sprachumfang besitzen ($\Rightarrow$ Intuitiv verständlich)
- Freitext-Beschreibungen zulassen
- Eine einfache Dokumentenverwaltung unterstützen

- **Grundsatz:**
  Keep it as simple as possible

  but not simpler!

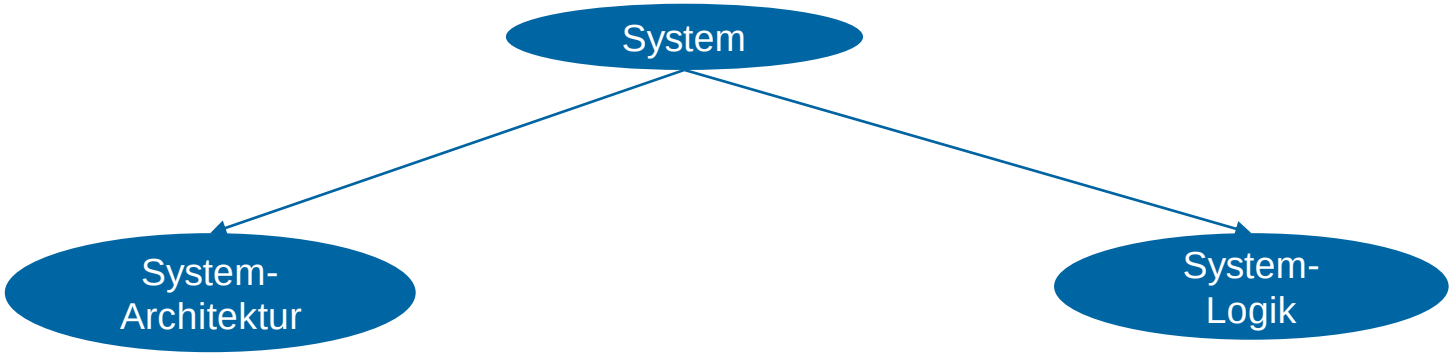


OST

Das Ziel ist der BPMN Business Process Model and Notation nach dem Begründer der BPMN, Stephen A. White

The primary goal of the BPMN effort was to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes. BPMN will also be supported with an internal model that will enable the generation of executable BPEL4WS. Thus, BPMN creates a standardized bridge for the gap between the business process design and process implementation.

Theorie: Aufbau



Beinhaltet:

- Systemkomponente und Subsystem (nesting)
- Ressourcen, Material, Waste
- Entities
- Beschreibung des Entityflow durch das System

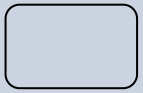
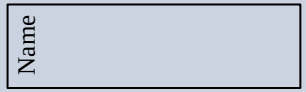
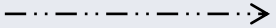
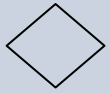
Jede Systemkomponente hat:

- Eine logische Beschreibung diese beinhaltet:
  - Task und Subtask (nesting)
  - Token (zur Ablaufsteuerung)
  - Beschreibung des Tokenflow
  - temporale Beschreibung
  - Ressourcen-Nutzung

*Die Beschreibung der System-Architektur und der System-Logik basieren auf der BPMN - Syntax **plus** Ergänzungen $\Rightarrow$ simBPMN*

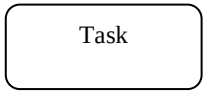
OST

Theorie: Grundelemente der BPMN

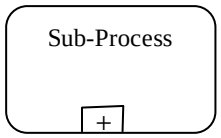
Flow Objects	Connectors	Artifacts	Swimlanes
Activity/Task Process 	Sequence Flow 	Data Object 	Pool 
Sub-Process 			
Events 	Message/Material Flow 	Text Annotation 	Lanes
Gateways 	Association	Group 	

OST

Theorie: Grundelemente der BPMN- Task und Sub-Task



- **Ein Task wird hier als *atomic unit* beschrieben, die Bestandteil eines Processes ist und nicht weiter unterteilt werden kann, Terminal.**
- **Ein Sub-Prozess enthält wiederum weitere Tasks, Subprozesse plus Ablauflogik.**



Kommentar:

Die Unterteilung Aktivität, Task und Sub-Prozess ist etwas verwirrend.

Interpretation:

- Der Task oder die Aktivität kann als atomarer Prozess oder nur als Processschritt angesehen werden.
- Ein Subprozess besteht aus weiteren Prozessen (Tasks) oder weiteren Sub-Prozessen. So kann eine verschachtelte (nested) Systemstruktur beschrieben werden.

OST

Start

Intermediate

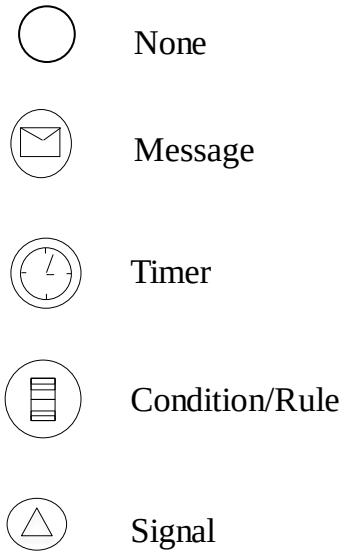
End

Ein Event

- **ist etwas, das zu einem bestimmten Zeitpunkt (Modellzeit) »passiert»**
- **beeinflusst den Ablauf eines Prozesses**
- **Ein Token**
  - Kann beim Durchlauf selbst wieder Entitys oder Token erzeugen, verzögern oder zerstören lassen
  - besitzt Attribute,
  - Die Attribute können beim Durchlauf eines Token durch einen Prozess verändert werden
  - Die Attribute können bestimmen, wie mit dem Token innerhalb eines Prozesses verfahren wird

OST

Theorie: Grundelemente der BPMN: Beispiele Events



- **Markiert, wo ein Prozess startet, Auslöser ist nicht erforderlich**
- **Trigger können durch interne Zeichen oder Marker differenziert werden**
  - None- Marker: wird zum Start eines Sub-Prozesses benutzt oder wenn der Start nicht definiert ist
  - Message-Marker: eintreffen einer Nachricht
  - Timer-Marker: Ablauf, einmalig, periodisch
  - Conditional/Rule-Marker: eintreffen einer bestimmten Bedingung, z.Bsp. Das Überschreiten eines Grenzwertes
  - Signal: erhalt eines Signals

Theorie: Grundelemente der BPMN: Intermediate-Event


Message


Timer


Error

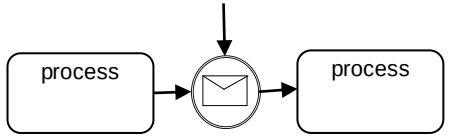

Cancel


Rule

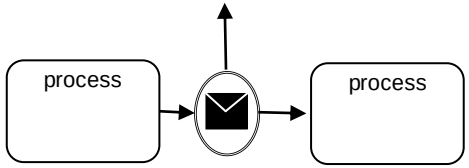

Signal

- Diese Events treten auf, nachdem der Prozess bereits gestartet wurde
- Die Trigger oder Auslöser der Events sind durch die Piktogramme im Kreis verdeutlicht, es können weitere Piktogramme definiert werden.
- Es gibt zwei Stellen, an denen diese Events platziert werden können
  - zwischen den Prozessen

Catching event:



Throwing event:



- Anmerkung:

Fangende Events markieren einen Systemzustand,
der nur verlassen werden kann, wenn das definierte Event auftritt!
Dieses Verhalten kann auch explizit durch eine EFSM (extended finite state machine) beschrieben werden

OST

BPMN: End-Event

None

Message

Cancel

Terminate

Error

Signal

- Ende- Events oder Stellen, zeigen wo ein Prozess aufhört oder terminiert
- Oder wo ein Fluss von Token endet und stellen damit auch Senken des Tokenflusses dar.
- Die Senken können wieder spezialisiert werden

OST

BPMN: Gateways

Data-based Exclusive Gateway 

Parallel Gateway 

Inclusive Gateway 

Event-based Exclusive Gateway 

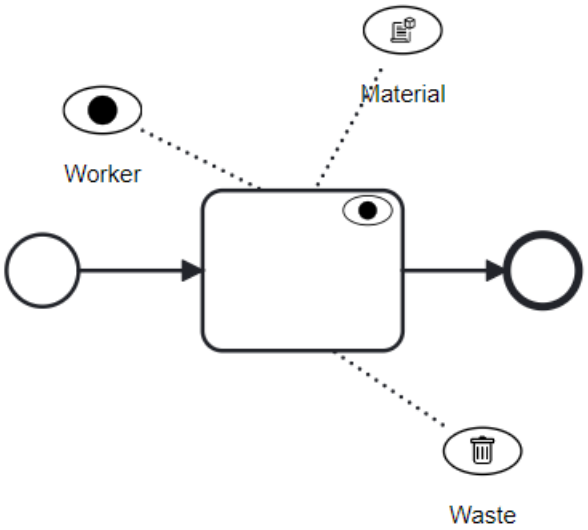
Complex Gateway 

- Gateway erlauben es den Fluss von Token/Entities zu steuern oder zu kontrollieren
- Alle Gateways sind *Diamonds*
- Durch die internen Marker werden verschiedene Typen von Gateways definiert
- Alle Gateways teilen oder vereinigen den "Fluss der Token"

OST

Systemstruktur oder Architektur eines Modells

- Eine Systemarchitektur kann durch Systemkomponenten und Subsystemkomponenten einen Fluss von Entitys durch das System beschrieben werden.
- Diese Beschreibung ist die höchste Abstraktionsebene und bildet quasi die Architektur des Simulationsgegenstands ab.
- Diese wird durch die Standarddarstellung der BPMN mit Task, Sub-Prozess, Flow und Gateway dargestellt.
- Jede Systemkomponente besitzt eine erweiterbare Default-Logik, die durch das Eintreffen eines Events abgearbeitet wird.
- Eine Systemkomponente kann verschiedene Zustände annehmen, diese Zustände können durch interne, externe oder spontane Events/Ereignisse verlassen werden.
- Einer System- oder Subsystemkomponente können weitere Ressourcen zugeordnet werden.



Ressource/Material/Waste

Um eine Systemkomponente (terminal) auszuführen

- benötigt sie für eine bestimmte Zeit
  - mindestens eine Ressource (default) und eventuell
  - Material und produziert eventuell
  - Waste
- die Assoziation zu einer Ressource kann angezeigt werden
- Ressourcen können nur in der Architektursicht erzeugt werden
- Nicht Terminal-Systemkomponenten haben keine Default-Ressource, da sie nur als Strukturelement dienen, und müssen durch Terminal-Systemkomponenten beschrieben werden

Verhaltenslogik einer terminalen Systemkomponente

- **Beschreibung der Verhaltenslogik:**
  - die Abarbeitung einer Serie von Tasks werden durch den Ablauf von Token gesteuert
    - Ein Token wird ausgelöst durch ein Event
    - Ein Event wird in der Regel bei einer Zustandsänderung eines Entitys oder allgemein eines Objektes ausgelöst
  - ein Token ist mit dem auslösenden Entity assoziiert, so das über das Token auf die Attribute des assoziierten Entitys zugegriffen werden kann
  - Der Fluss von Token kann durch Gates, und catching Events beeinflusst werden

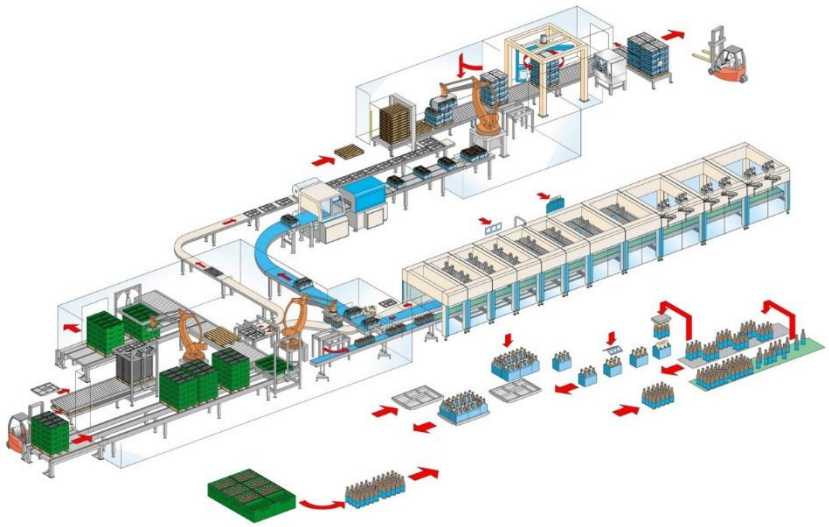
- **Beschreibung der Verhaltenslogik:**
  - Ein Token kann folgende Aktivitäten auslösen:
    - Events erzeugen
    - den eigenen Zustand, den Zustand eines anderen Prozesses oder den globalen Zustand verändern
    - den Zustand des Entitys verändern
    - Entitys erzeugen oder zerstören
- **zur Ausführung benötigte Ressourcen/Material/Waste**
- **Eine Task benötigt zur Ausführung eine Modellzeit:**
  - 1. explizite Zeitangabe incl. Verteilung
  - 2. keine Zeitangabe bedeutet, dass keine Modellzeit benötigt wird

Zusammenspiel Token und Entity

- **Architektur/Struktur vs. Prozesslogik**
  - Architektur/Struktur (Entity-Flow)
  - Prozesslogik (Token-Flow)
- **Ressourcen, besitzen in der Regel Assoziationen zu den logischen Systemkomponenten**
- **Eine Systemkomponente besitzt Zustände (EFSM)**

Architektursicht vs. Logiksicht

- **Architektur mit Entity-Flow**

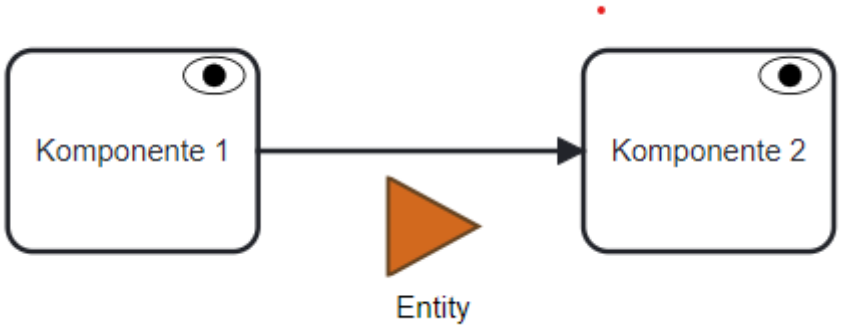


- **Logik einer Maschine durch den Token-Flow gesteuert**

OST

Architektur (Entity-Flow)

- **Systemkomponente**
  - Besitzt eine ID
  - Führt eine Verhaltensvorschrift aus (beschrieben durch die hinterlegte Logik)
  - Kann verschiedene Zustände annehmen, diese Zustände können sich durch ausführen der Verhaltensvorschrift oder spontan ändern
  - Die Verhaltensvorschrift kann geändert werden, ohne die Komponente zu zerstören (Konzept der Delegation)
  - Es können mehrere Komponenten mit der gleichen Verhaltensvorschrift inkarniert werden (Objekte, Kapazität)
  - ….



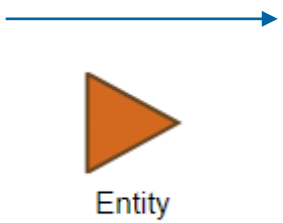
OST

Architektur (Entity-Flow)

- **Systemkomponente**
  - Benötigt zur Ausführung der Verhaltensvorschrift i.d.R.
    - Ressourcen
    - Material und produziert Waste
    - Zeit
  - Besitzt i.d.R. einen Eingang mit einem Eingangspuffer
  - Besitzt i.d.R. Ausgang mit einen Ausgangspuffer
  - Kann ein Eingangselement in ein Ausgangselement transformieren oder zerstören
  - Kann spontan Ausgangselemente erzeugen
  - …

Architektur (Entity-Flow)

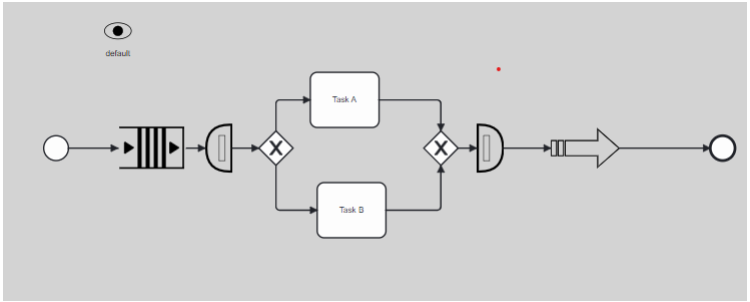
- **Entity:**
  - Besitzt einen Zustand
  - Besitzt Attribute
  - Wird über einen Sequence-Flow von Prozess zu Prozess weitergeleitet
  - kann Nachrichten übertragen



OST

Logik (Token-Flow)

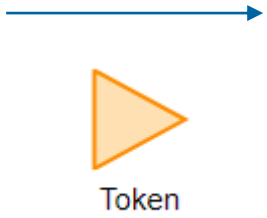
- Die Abarbeitung einer Serie von Steps wird durch den Ablauf von Token gesteuert
  - Ein Token wird ausgelöst durch ein Event
  - Ein Event wird in der Regel bei einer Zustandsänderung eines Entitys oder allgemein eines Objektes ausgelöst
- Ein Token ist mit dem auslösenden Entity assoziiert, so dass über das Token auf die Attribute des assoziierten Entitys zugegriffen werden kann
- Der Fluss von Token kann durch Gates und Catching-Events beeinflusst werden



OST

Token

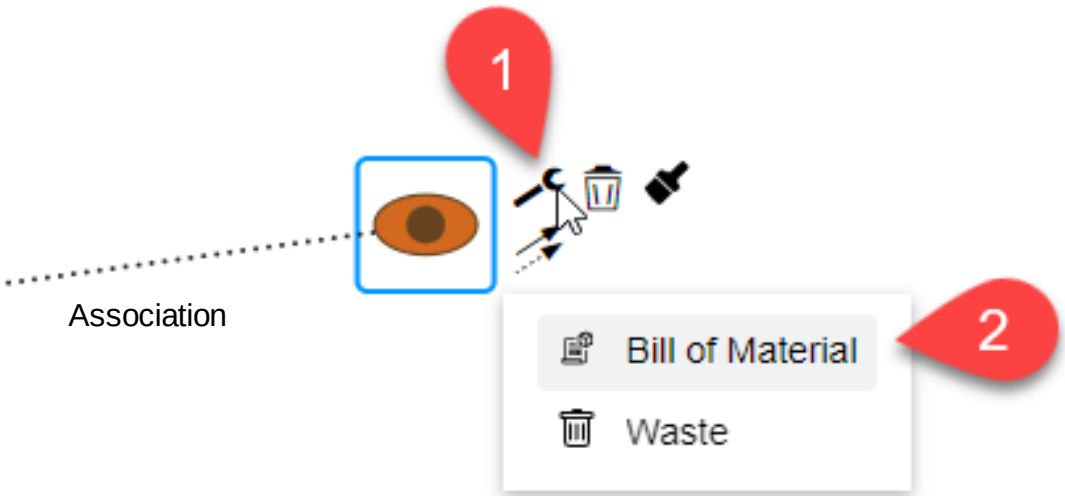
- Wird durch ein Event ausgelöst
  - Entity kommt bei Maschine an
  - Timer/Intervall abgelaufen
  - etc.
- Besitzt einen Zustand
- Kann Zustandsänderungen auslösen
  - Eigenen Zustand
  - Den Zustand des Entity
  - Den Zustand von Ressourcen
  - Den Zustand eines anderen Prozesses
  - Den globalen Zustand
- Kann das Werfen von Events auslösen
- Kann das Erzeugen oder Zerstören auslösen
- Beschreibung des Token-Flow



OST

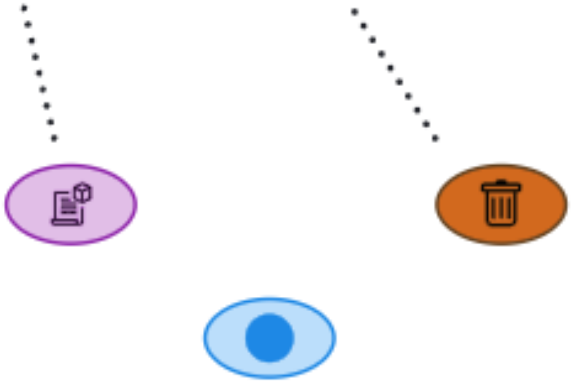
Ressource

- **Ressource ist mit einem oder mehrere Komponenten/Prozessen assoziiert und wird für dessen Ausführung benötigt.**



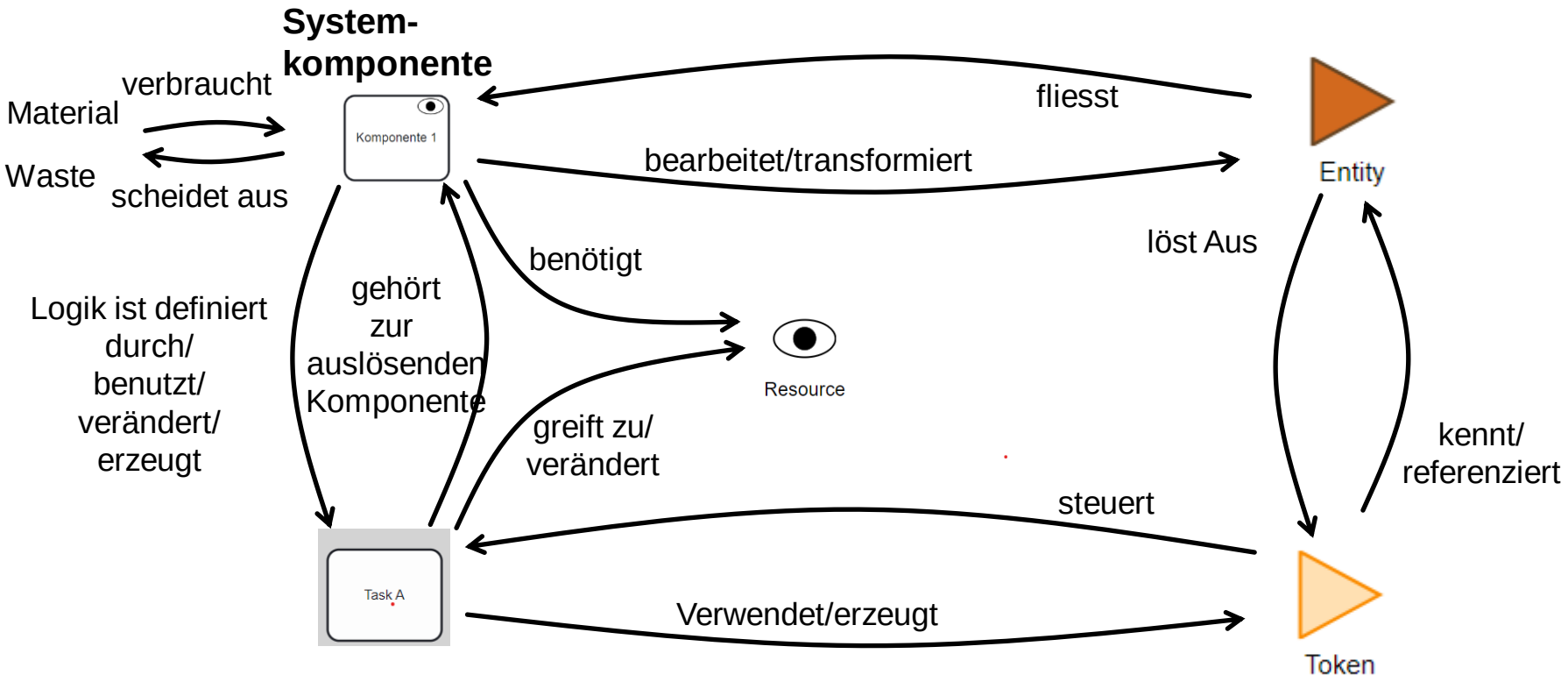
- **Ressource, Bill of Material, Waste**
  - Wird von einem oder mehrere Systemkomponenten für dessen Ausführung benötigt/verbraucht
  - Kann einen Zustand und Attribute besitzen
  - Der Materialfluss/Waste wird angezeigt durch

Beispiel: Kaffeeausgabe mit einem Kaffeeautomaten.



OST

Zusammenhang der Koponenten



Einführung in die simBPMN

1 Theorie

2 Praxis

2.1 Der simBPMN Visulizer: Präsentation

2.2 Ein einfaches Beispiel

2.3 Hands on: Ausbau des Beispiels

3 Collaboration und Ausblick

Ein einfaches Beispiel: Vom Word-Model zur Spec

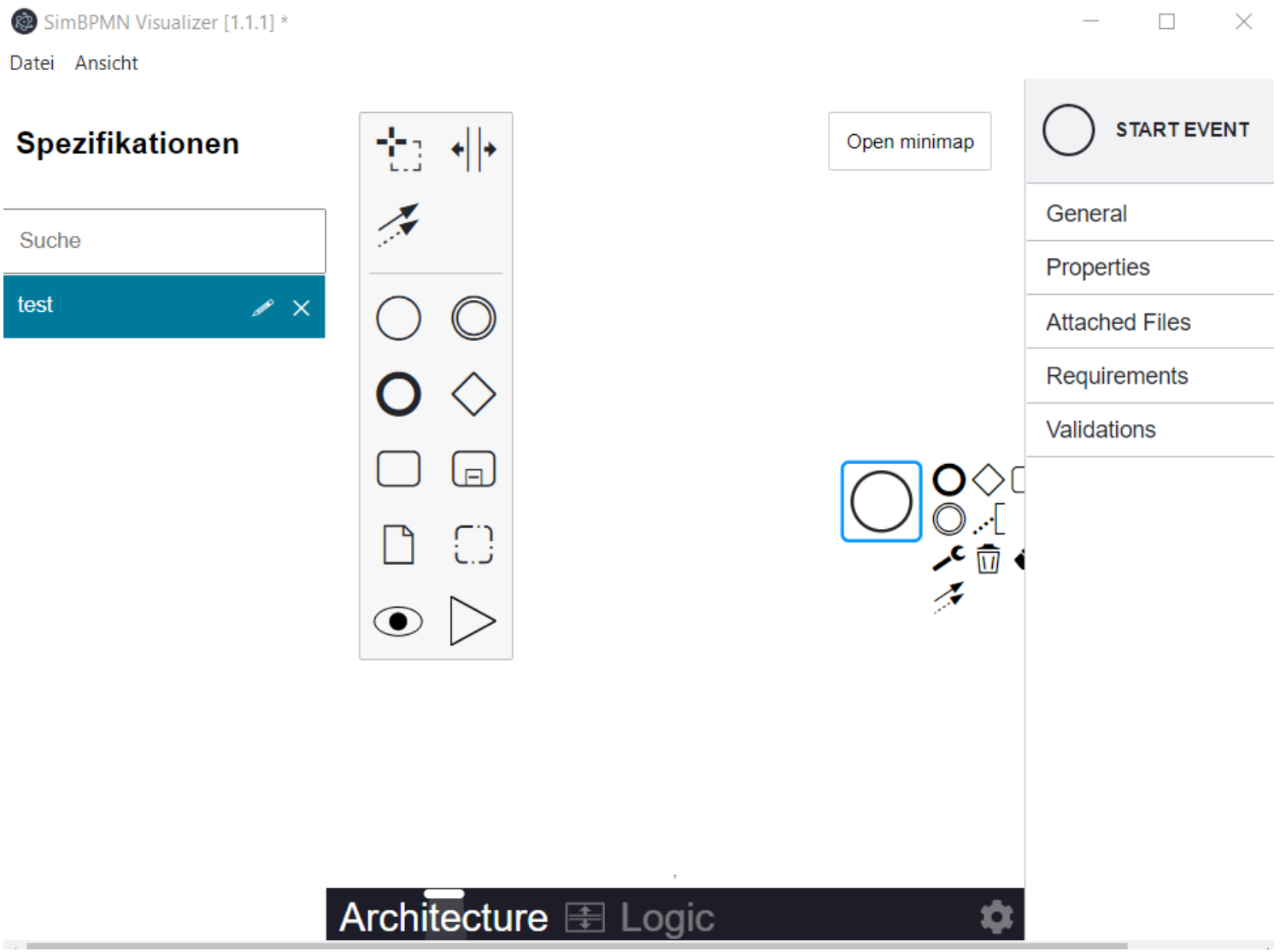
- **Kaffeepause**

Für die Kaffeepause wird der Pausenraum um eine professionelle Kaffeebar erweitert:

Jetzt werden Heißgetränke an einer Bar von einem Barista zubereitet, es gibt einen Kühlschrank mit Snacks und Softgetränken zur freien Verfügung und der Raum ist mit gemütlichen Sitzplätzen umgestaltet worden.

Nach Erfahrungswerten der Kantinenbetreiber ist zu Stoßzeiten mit 200 Besuchern zu rechnen. Davon nehmen 20% direkt Platz, 40% bedienen sich an Softdrinks und Snacks bevor sie sich setzen, und 40% holen sich erst einmal einen Kaffee, bevor sich die eine Hälfte direkt setzt, und die andere sich noch zusätzlich mit Snacks versorgt bevor sie zum Platz geht.

Präsentation



OST

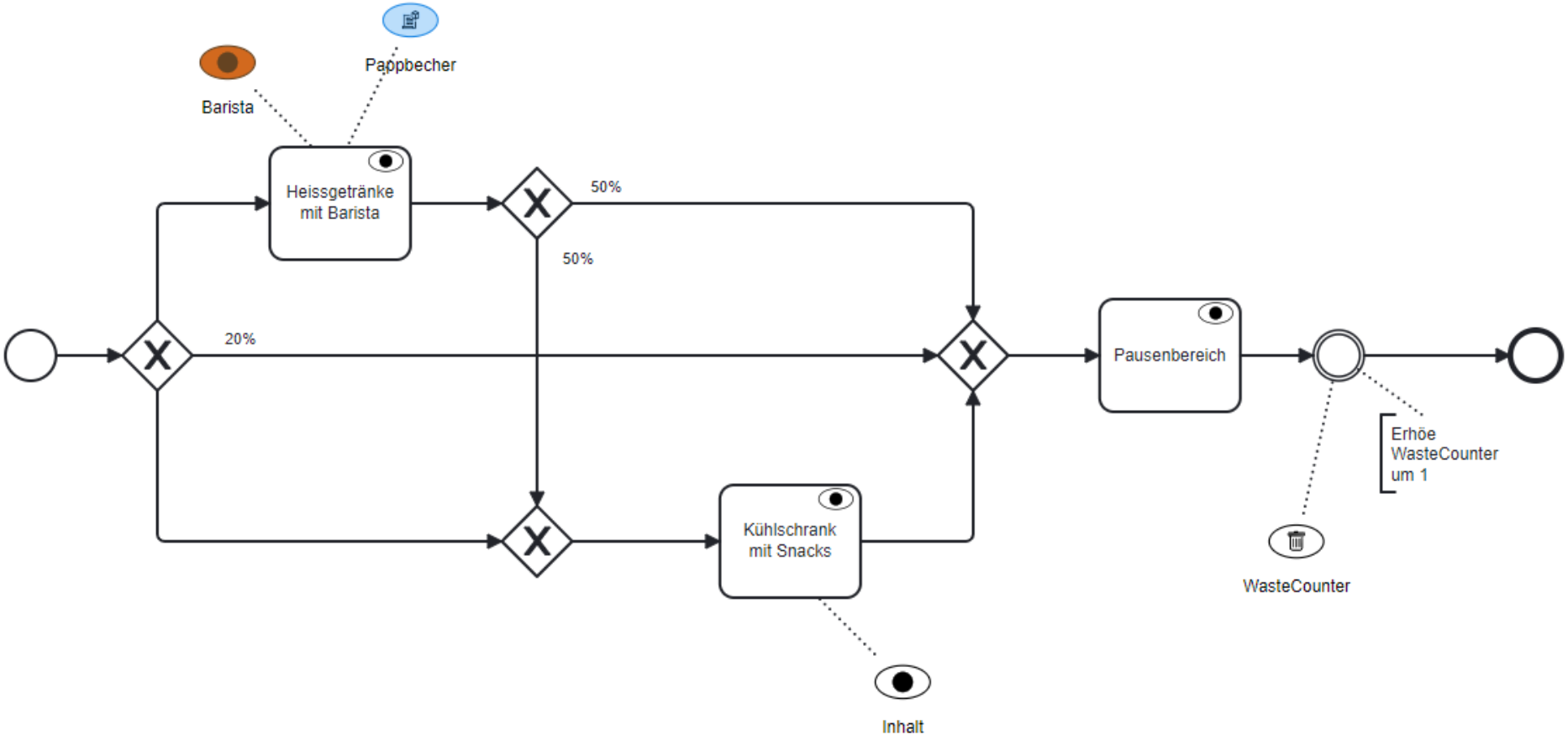
Hands on: Ausbau des Beispiels

- **In der Beschreibung sind die Ressourcen noch nicht berücksichtigt worden:**

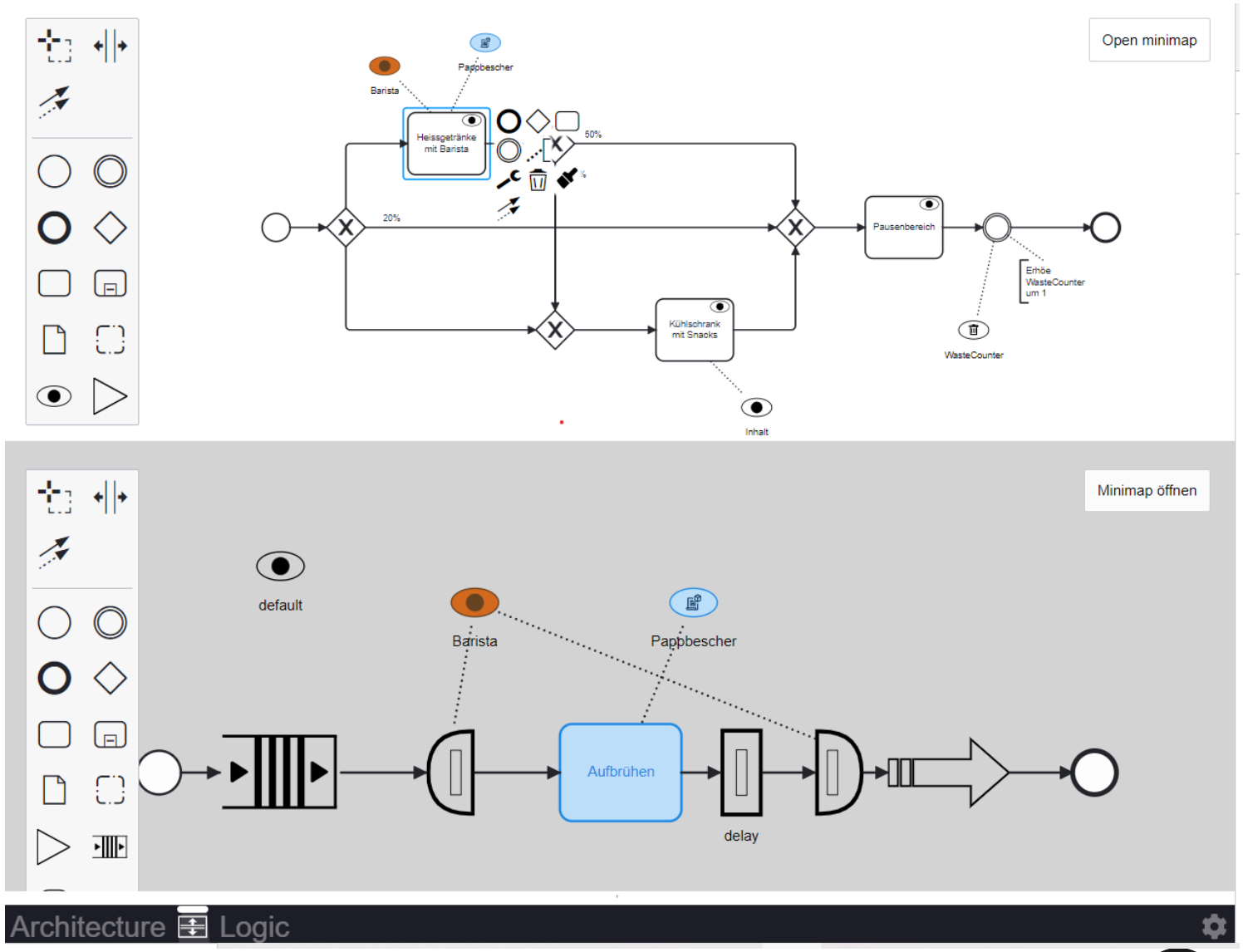
Wasseranschlüsse und Strom sind verlegt, Kaffeebohnen und Milch sind Verbrauchsmaterialien, die in großer Menge zur Verfügung stehen.

Es werden Pappbecher für die Heißgetränke verwendet, die in großer Menge vorrätig sind.

Hands on: Mögliche Lösung - Architektur



Hands on: Mögliche Lösung – Logik für Heissgetränke



Hands on: Ausbau des Beispiels

- **Ergänzen Sie das Beispiel um: nachhaltige Kaffebecher sprich Tassen und deren Reinigung!**

Aktuell werden die Heißgetränke in Pappbechern ausgeschenkt. Es ist jedoch eine Umstellung auf Kantinengeschirr geplant, um Abfall zu vermeiden. Dieses Geschirr muss zur Rückgabe gebracht werden

- **Sie können den simBPMN Visualizer herunterladen:**

https://bit.ly/simBPMN-Illmenau

Sie finden dort ebenfalls das Beispiel “Kaffeepause”

Hands on: Ausbau des Beispiels: mögliche Lösung

- Der Code wird auf GitHub gehosted
  - https://github.com/INSRapperswil/simbpmn-visualizer
- Resultat einer Bachelorarbeit sowie Weiterentwicklung im Institut
- Nächste Schritte
  - CI-Pipeline
  - Features vervollständigen
  - Refactoring
  - Testbasis implementieren
- Collaboration ist ausdrücklich erwünscht
  - GitHub Issues: https://github.com/INSRapperswil/simbpmn-visualizer/issues
    - bitte Labels beachten
  - Entwicklung / PullRequests

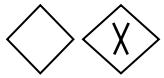
Einführung in die simBPMN: Ausblick

- Bereinigung und Schärfung der Parametersettings
- Einfache statische Analysen
- Integration eines Toolzusatzes zu Beschreibung von Zustandsautomaten
- die automatische Transformation der Spezifikation in eine geeignete Implementierungssprache (Erzeugung eines Skeleton)
- Einladung zur Mitarbeit bei der Weiterentwicklung des Tools

Fragen

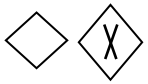
Backhand Ergänzungen

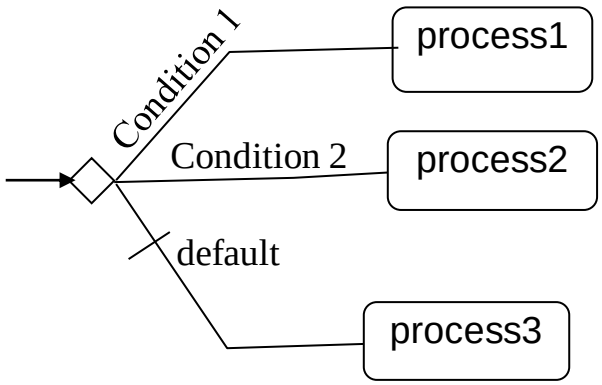
BPMN: Gateways: Exclusive Gateway

Data-exclusive Gateway: 

- Exclusive (XOR) Gateway erlauben es, den Fluss von Token zu steuern
- Die Entscheidung, welchen Pfad das Token weitergereicht wird kann:
  - Datenbasiert sein
    - Zufällig
    - An eine Bedingung geknüpft
  - Eventbasiert sein
- Können ebenfalls zum zusammenfügen der Token ströme genutzt werden

Data-based Exclusive Gateway



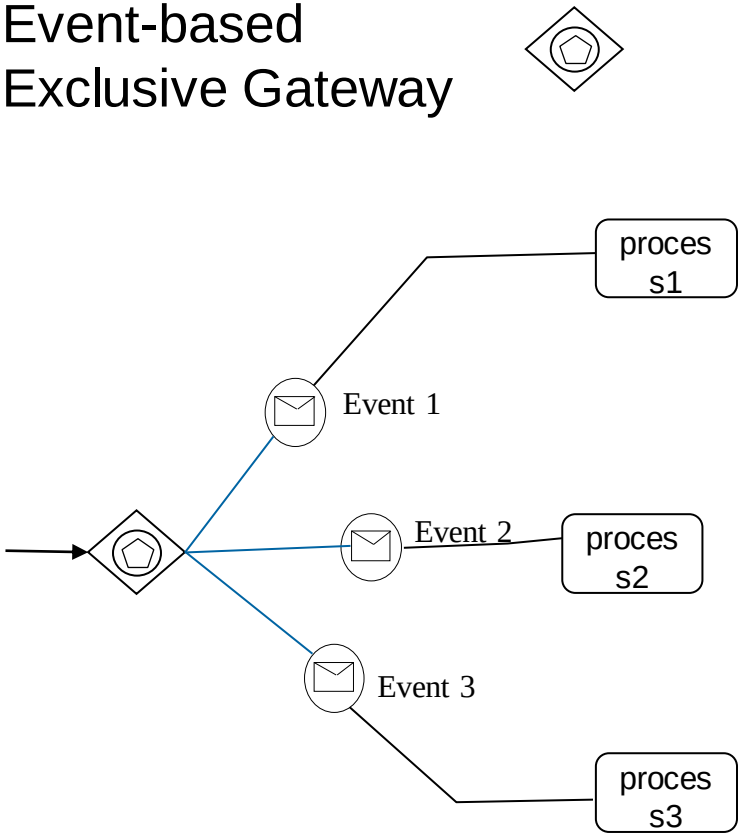


- Oft benötigtes Entscheidungskriterium
- Der alternative Pfad wird anhand von Bedingungen ausgewählt
- Bedingungen:
  - über den Systemzustand
  - über den Zustand des eintreffenden Token
  - Gewichtung des Pfades
- Trifft keine der Bedingungen zu
  - kann ein Default-Pfad angegeben werden
  - Oder eine Warteschlange definiert werden, in der die Token warten, bis eine der Bedingungen erfüllt wird

OST

BPMN: Gateways: Event-based Exclusive Gateway

Event-based Exclusive Gateway



- Welcher Weg eingeschlagen wird, hängt von dem Event ab, das geworfen wird, z.B.
  - Eintreffen einer Nachricht, eines Signals
  - Ablauf eines Timers
  - Das erste eintreffende Event löst den entsprechenden Folgeprozess aus
- Eintreffende Token/Entities müssen im Gateway zwischengespeichert werden, bis ein Event die Weiterleitung des oder der Token festlegt
- Ggf. ist die Queue weiter zu spezifizieren

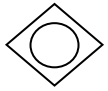
Parallel Gateway 

- Flow teilt sich auf und läuft parallel weiter (erzeugen eines weiteren Token)
- Zusammenführung "UND".

- Problem bei der Zusammenführung
  - kann über Parameter der Token gelöst werden

- *Wie passt das zu Entitys?*

BPMN: Gateways: Inclusive Gateway

Inclusive Gateway 

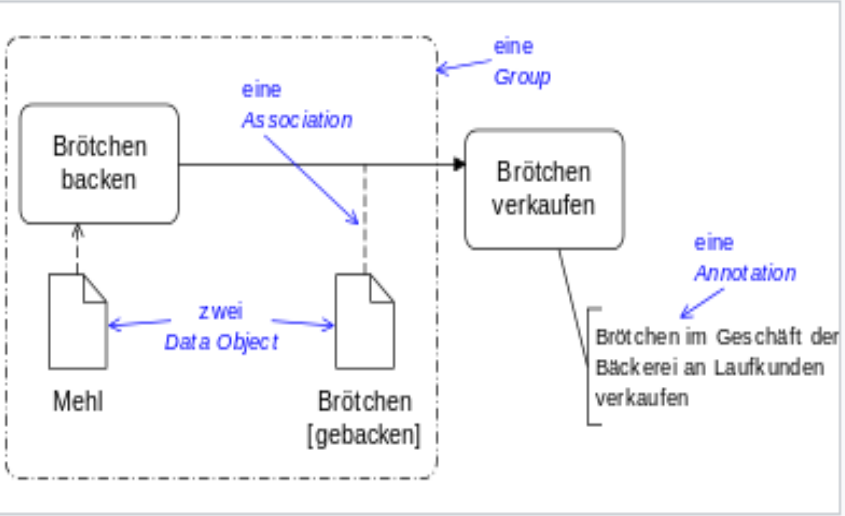
- Je nach Situation Token teilt bzw. vervielfacht sich das und läuft parallel weiter (erzeugen weiterer Token)
- *Wie pass das zu Entitys?*
- eine oder mehrere oder alle Möglichkeiten finden statt
- Problem bei der Zusammenführung
  - kann über Parameter der Token gelöst werden

BPMN: Gateways: Inclusive Gateway

Complex Gateway 

- **Joker, kann frei definiert werden**

Artefacts (Quelle: https://de.wikipedia.org/wiki/Business_Process_Model_and_Notation)



Beispiele für *Artifacts*

Eine **Annotation** ist ein Kommentar, der einem Element eines Geschäftsprozesses zugeordnet werden kann.

Ein **Data Object** repräsentiert ein Artefakt, das der Geschäftsprozess bearbeitet. Mit *Data Objects* können sowohl elektronische Objekte wie Dokumente oder Datensätze, als auch physische Objekte wie Brötchen oder Bücher dargestellt werden.

Eine **Group** ist ein Hilfsmittel, um Elemente eines Geschäftsprozess visuell zusammenzufassen. Sie ist nicht zu verwechseln mit einem *Subprocess*.