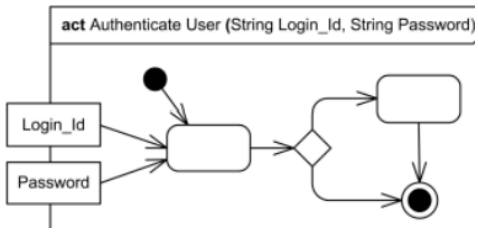
Dynamisches Verhalten

Aktivitätsdiagramme beschreiben **Ausschnitt** aus Ablaufmöglichkeiten eines Systems mit *Hilfe von Aktivitäten*

Einsatz: Detaillierte Sicht auf Anwendungsfälle | den Ablauf innerhalb von Klassen | Geschäftsprozessmodellierung mit der UML

Aktivität – Parametrisiertes Verhalten, dargestellt als koordinierter Fluss von Aktionen

Aktivitätsknoten:

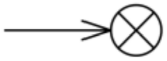


Aktionen: abgerundete Rechtecke
Objekte: Rechtecke
Kontrollflusselemente: verschiedene Symbole für Entscheidungen, parallele Ausführung
Kontrollfluss: Verbinden von Knoten mit gerichteten Kanten

Aktion – Benanntes Element, das einen einzelnen atomaren Schritt innerhalb der Aktivität darstellt

Kann eingehende und ausgehende *Aktivitätskanten* enthalten, die *Kontroll- und Datenfluss* von und zu anderen Knoten definieren
Kann andere *Aktivitäten* aufrufen (call activity action)
Beginnt mit der *Ausführung*, wenn alle *Eingabebedingungen* erfüllt sind

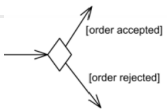
---

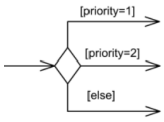
Start- und Endzustand 

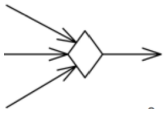
Startknoten – Start des Kontrollflusses in einer Aktivität
Aktivitäts-Endknoten – Ende jeglicher Kontrollflüsse in einer Aktivität
Fluss-Endknoten – Ende eines einzelnen Kontrollflusses

Endzustände können mehrfach vorkommen, Startzustände ideal nur einmal

---





Entscheidung 

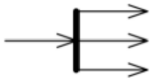
Steuerknoten, der eingehenden Kontrollfluss akzeptiert und Kontrollfluss aus mehreren ausgehenden Kontrollflüssen auswählt

Annotation von Bedingungen in Form von **booleschen Ausdrücken** an Transitionen
Unterstützt auch einen vordefinierten **else -Kontrollfluss**
Optionaler **Merge-Knoten** fügt mehrere eingehende alternative Flüsse zusammen

---

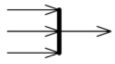
Parallele Ausführungspfade

Fork – Steuerknoten, der einen *eingehenden Kontrollfluss* in mehrere nebeneinander laufende Kontrollflüsse aufteilt

Die Ausführung des Diagramms durchläuft alle parallelen Pfade 

Die Ausführungsreihenfolge nebenläufiger Pfade ist *irrelevant*: nacheinander, gleichzeitig, abwechselnd

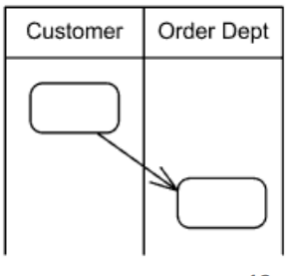
Join – Steuerknoten, der mehrere eingehende nebenläufige Kontrollflüsse synchronisiert

UND-Synchronisation: Die Kontrollflüsse aller Transitionen müssen vorliegen, bevor die Transition starten kann 

Probleme mit Fork und Join

- Es dürfen nur *Zweige* in einem Join *zusammengeführt* werden, die zuvor aus *demselben Fork hervorgegangen* sind.
- *Zweige*, die aus *verschiedenen* (unabhängigen) Forks stammen, können nicht *zugleich aktiv* sein.

Verantwortlichkeitsbereiche

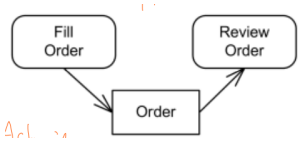


> Gruppierung von Aktionen anhand gemeinsamer Merkmale

Definieren **organisatorische Verantwortlichkeiten**
Können hierarchisch aufgebaut werden: Klassenzugehörigkeit, Organisationseinheiten, Akteuren

---

Objekte und Objektfluss



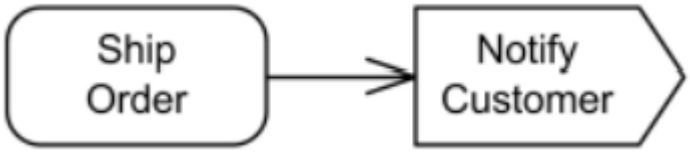
> **Objektflusskanten** – Kanten, mit denen der Datenfluss zwischen Aktionsknoten angezeigt wird

> **Objektknoten** – Knoten, mit dem Objekte definiert werden. Vorliegen eines Objektes zu Zeitpunkt innerhalb einer Aktivität

> **Pin** – Objektknoten für Ein- und Ausgänge von Aktionen

Data Stores – Zentraler Speicherknoten für nicht-transiente Informationen

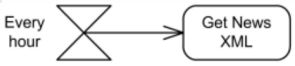
Signale senden und empfangen



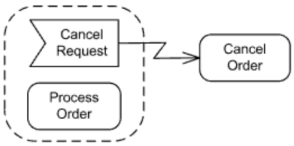
Sendesignalaktion – Senden eines Signals

Annahmeereignis – Aktion, die auf das Eintreten eines bestimmten Ereignisses (z. B. Signal) wartet

Sender- und Empfängerobjekte von Signalen müssen nicht spezifiziert werden



Wartezeitereignisse und Unterbrechungen



Wartezeitaktion – Warten auf das Eintreten eines Zeitereignisses

Unterbrechbare Aktivitätsregionen – Aktionen, deren Ausführung durch Ereignisse unterbrochen werden können

Revision #5
Created 26 September 2022 16:14:14 by Merith Holtmann
Updated 27 September 2022 06:37:04 by Merith Holtmann