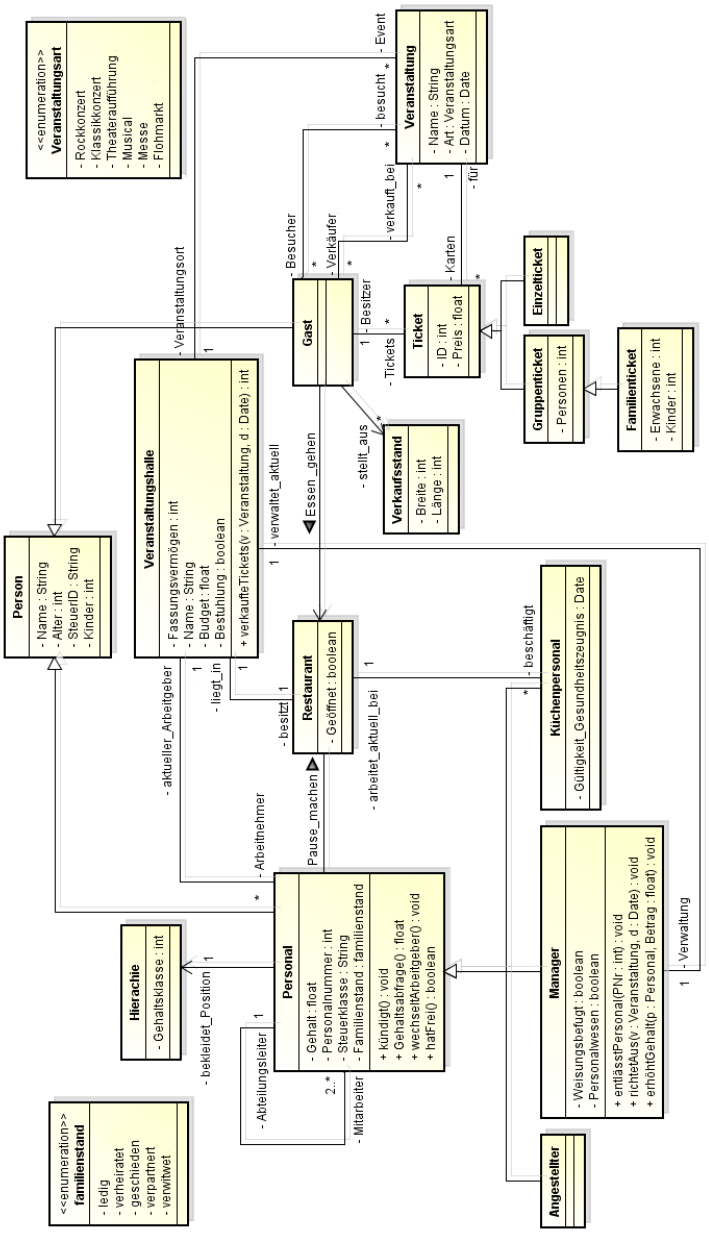
Softwarekonstruktion – Übung 2

2 OCL

2.1 OCL Anwendung III



Betrachten Sie das von Übungszettel 1 bekannte UML-Klassendiagramm in Abb.1.

Drücken Sie folgende Constraints in Form von OCL-Ausdrücken aus.
Verwenden Sie dabei ausschließlich die Operationen, die auf Übungsblatt 1 (in den Aufgaben 1.3 und 1.7) eingeführt wurden.

1. Die Anzahl von Personen eines Gruppentickets darf nicht negativ oder 0 sein!
```
context Gruppenticket
inv: self.Personen > 0
```

2. Für Tickets gelten besondere Bestimmungen: Gruppentickets werden erst ab 2 Personen ausgestellt, bei Familientickets müssen mindestens 2 Erwachsene und 1 Kind gebucht werden!
```
context Familienticket
inv: self.Erwachsene >= 2 and self.Kinder >= 1

context Gruppenticket
inv: self.Personen >= 2
```

3. Das Personal kann sein Gehalt über Gehaltsabfrage() abfragen. Dabei wird das aktuelle Gehalt zurück gegeben!
```
context Personal::Gehaltsabfrage(): float
post: result= self.Gehalt
```

4. Wird das Gehalt des Personals um einen Betrag erhöht, so ist das neue Gehalt danach um genau diesen Betrag größer als das Gehalt zuvor. Damit die Gehaltserhöhung überhaupt stattfindet, muss der Betrag, um welchen das Gehalt erhöht werden soll, mindestens 3 % des aktuellen Gehalts betragen!
```
context Manager::erhöhtGehalt(p:Personal, Betrag:float): void
pre: Betrag > p.Gehalt * 0.03
post: p.Gehalt= p.Gehalt@pre + Betrag
```

5. Bei dem Personal gelten die Bestimmungen, dass das Küchenpersonal über ein aktuell gültiges Gesundheitszeugnis verfügen muss!
```
context Küchenpersonal
inv: self.Gültigkeit_Gesundheitszeugnis > = today
```

6. Kündigt Jemand vom Personal einer Veranstaltungshalle, so gehört er anschließend nicht mehr zu den Arbeitnehmern der Veranstaltungshalle. Damit derjenige überhaupt kündigen kann, muss er natürlich vorher bei dieser Veranstaltungshalle angestellt gewesen sein!
```
context Personal::kündigt():void
pre: self.aktuellerArbeitgeber.Arbeitnehmer -> includes(self)
post: self.aktuellerArbeitgeber@pre.Arbeitnehmer -> excludes(self)
```

EPK

2.2 Grundlagen

2.2.1 Mit welchen EPK-Elementen würden Sie folgende Aussagen modellieren?

1. Router konfigurieren
2. Werbebroschüre im Briefkasten
3. DSL-Anschluss ist geschaltet
4. Störungsstelle anrufen
5. DSL-Hardware versenden

1. Router konfigurieren = Funktion
2. Werbebroschüre im Briefkasten = Ereignis
3. DSL-Anschluss ist geschaltet = Ereignis
4. Störungsstelle anrufen = Funktion
5. DSL-Hardware versenden = Funktion

2.2.2 Was versteht man bei EPKs unter einem Konnektor? Welche Arten gibt es? Erklären Sie diese in eigenen Worten.

Ein Konnektor verknüpft einzelne Kontrollflüsse miteinander bzw. splittet sie auf. Ob und welche(r) nachgelagerte(r) Kontrollfluss/ -flüsse zum Tragen kommen hängt vom jeweiligen Operator ab:

XOR-Konnektor ("entweder-oder") join: der ausgehende Kontrollfluss wird verfolgt, wenn genau einer der eingehenden Kontrollflüsse am Konnektor angelangt.
split: nachdem der Kontrollfluss am Konnektor angelangt ist, wird genau einer der ausgehenden Kontrollflüsse verfolgt.

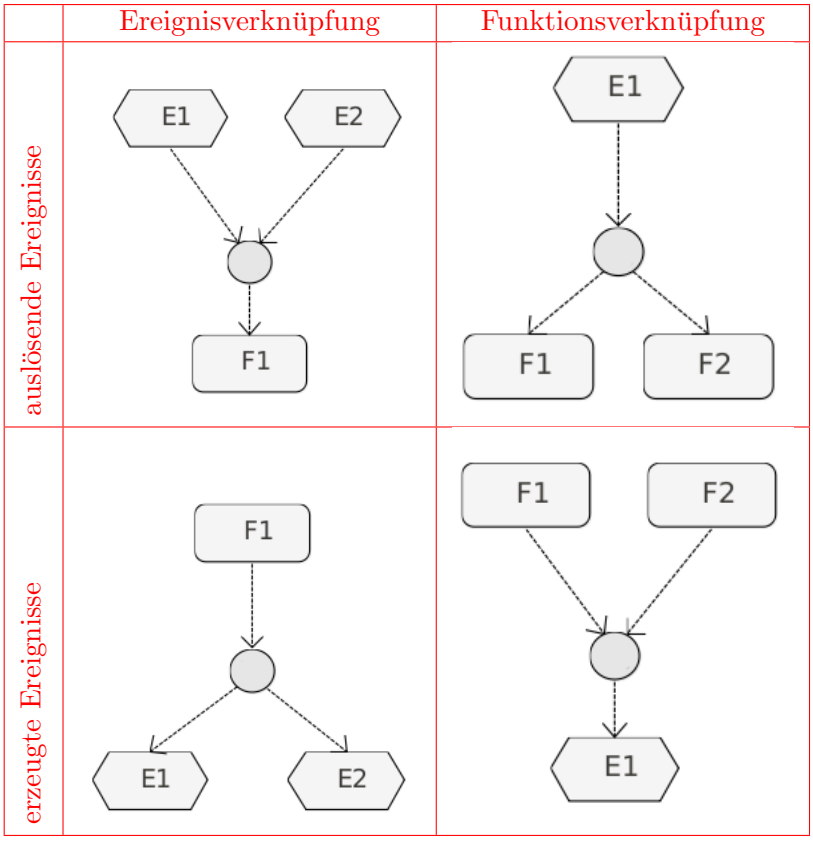
AND-Konnektor ("und") join: der ausgehende Kontrollfluss wird verfolgt, wenn alle der eingehenden Kontrollflüsse am Konnektor angelangen.
split: nachdem der Kontrollfluss am Konnektor angelangt ist, werden alle ausgehenden Kontrollflüsse verfolgt.

OR-Konnektor ("und/oder") join: der ausgehende Kontrollfluss wird verfolgt, wenn mindestens einer der eingehenden Kontrollflüsse am Konnektor angelangt.
split: nachdem der Kontrollfluss am Konnektor angelangt ist, wird mindestens ein Kontrollfluss verfolgt.

2.3 Konnektoren

2.3.1 Beim Einsatz von Konnektoren unterscheidet man nach Ereignis- und Funktionsverknüpfung, sowie auslösende und erzeugte Ereignisse. Was könnte man darunter verstehen? Modellieren Sie jeweils ein Beispiel.

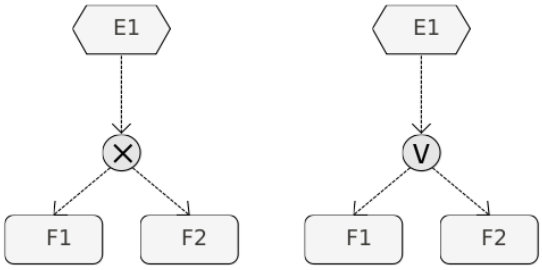
Ereignisverknüpfung: Mehrere Ereignisse werden mit einer Funktion verknüpft
Funktionsverknüpfung: Mehrere Funktionen werden mit einem Ereignis verknüpft



Die drei Möglichkeiten oben links, unten links und unten rechts sind für alle Konnektoren gültig. Für die Möglichkeit rechts oben siehe nachfolgede Aufgabe!

2.3.2 Eine der Variante aus Aufgabe 2.3.1 ist nur für den AND-Konnektor zulässig. Welche ist das und warum?

Da Ereignisse über keine Entscheidungsgewalt verfügen, darf nach einem Ereignis nur ein AND-Konnektor folgen. Würde man den Kontrollfluss nach einem Ereignis mit einem OR- oder XOR-Konnektor splitten, so gäbe es danach mehrere Möglichkeiten zur Fortsetzung. Dies widerspricht der Modellierungsannahme, dass in Ereignissen keine Entscheidung statt findet.
Die folgenden Varianten sind also unzulässig:

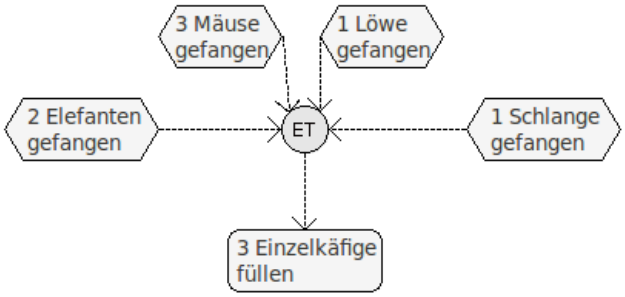


2.3.3 Funktionieren die in der Vorlesung vorgestellten Konnektoren auch für das zusammenführen von mehr als 2 Flüssen?

Ja, für die Standardkonnektoren immer.

2.4 Erweiterte Konnektoren

2.4.1 Folgender Prozesskettenauschnitt aus dem Arbeitsfluss eines Großwildfängers ist gegeben. Wie muss die Entscheidungstabelle aussehen damit die Funktion sinnvoll ausgeführt werden kann?



Im Grunde entspricht dieser ET-Konnektor der Aussage *genau 3 Tiere sind gefangen.*

Zulässige Kombinationen für das Weiterschalten des Kontrollflusses am ET-Konnektor sind also:

1. 2 Elefanten und 1 Löwe gefangen
2. 2 Elefanten und 1 Schlange gefangen
3. 3 Mäuse gefangen

Dies führt zu folgender Entscheidungstabelle:

E gefangen	M gefangen	L gefangen	S gefangen	Ergebnis
1	0	1	0	1
1	0	0	1	1
0	1	0	0	1

Für alle anderen Kombinationen aus E gefangen, M gefangen, L gefangen und S gefangen ist der Wert 0, der ET-Konnektor schaltet also nicht.

2.4.2 Modifizieren Sie das Diagramm aus Aufgabe 2.4.1 mit Hilfe eines Kombinationskonnektors so, dass parallel zum Käfig füllen Futter beschafft wird.