Vorlesung Wintersemester 2010/11

Konzepte und Anwendung von Workflowsystemen

Kapitel 5:

Workflow Nets

Lehrstuhl für Systeme der Informationsverwaltung, Prof. Böhm

Institut für Programmstrukturen und Datenorganisation (IPD)

Überblick Kapitel 5

- Einfache Workflow Nets
  - Wiederholung High-Level Petrinetze
  - Definition Workflow Nets
  - Vor- und Nachteile
- Erweiterte Workflow Nets
  - Motivation
  - YAWL (Yet Another Workflow Language)
    - Überblick
    - Beispiele
    - Workflow Spezifikation
    - Ausführungssemantik
    - Datenaspekt
    - Zusammenfassung
    - Übung

Wiederholung High-Level Petrinetze

- High-Level Petrinetze sind erweiterte (klassische) Petrinetze (S/T-Netze, „P/T-Nets“) um
  - Farben (gefärbte Petrinetze)
    - Daten (z.B. Identifikation von Fällen)
  - Zeit
    - Tokens (Marken) erhalten Zeitstempel
  - Hierarchien
    - Einbindung von Sub-Workflows

- Ein Petrinetz, das den Kontrollfluß eines Workflows darstellt, wird **Workflow Net (WF-Net)** genannt.

Definition Workflow Net

- Definition: Ein Petrinetz PN=(P,T,F) ist ein **Workflow Net (WF-Net)**, wenn folgende Eigenschaften zutreffen:
  - Es gibt eine Anfangsstelle i ∈ P (input), für die gilt: $\bullet i = \{\}$
  - Es gibt eine Endstelle o∈ P (output), für die gilt: $o\bullet = \{\}$
  - Jeder Knoten $x \in P \cup T$ liegt auf dem Pfad von i nach o

- Ein Workflow Net spezifiziert den Lebenszyklus eines Falles („case“)
- Weitere Konstrukte (Wdh)
  - Hierarchie, OR- und AND- Split (Verzweigung) /Join (Verknüpfung)
  - Trigger

Vorteile Workflow Nets

- Starke Ausdrucksmächtigkeit bei der Modellierung von Workflows
- Formale Semantik
  - Ausführungssemantik von Petrinetzen
- Graphische Repräsentationsform
  - Kommunikationsgrundlage bei Modellierung
- Analyse von Prozesseigenschaften
- Unabhängigkeit von Herstellern (Formalismus)

(Bisherige) Nachteile von Workflow Nets

- Zeitloses Schalten von Transitionen
  - Entspricht nicht dem Verhalten von Tasks in Workflows
- Keine explizite Darstellung der Daten
  - Gefärbte Token (Marken): keine Spezifikation der Nutzung dieser Datenstrukturen
- Aufwändige Identifikation von Marken und Zuordnung zu Workflow-Instanzen
  - Umsetzungsproblematik (z.B. Synchronisation) bei gefärbten Petrinetzen
- Nicht-deterministisches Verhalten von Petrinetzen
  - Explizite Spezifikation der erwünschten Kontrolle über Ausdrücke
- Unter Umständen sehr komplexe Modelle

- (weitere Nachteile werden folgen …)

Erweiterte Workflow Nets

- Motivation:
  - Existenz zahlreicher Workflow-Modellierungssprachen und WfMS
    - Bewertung schwierig bzw. basierend auf Tool-Funktionalität
  - „Workflow Patterns“ (Workflow-Muster)
    - Systematische Sammlung von Anforderungen (Kontrollfluß, Daten, Ressourcen, Ausnahmen) an Workflow-Modellierungssprachen
    - Grundlage zur funktionalen Bewertung und Vergleich von Modellierungssprachen bzw. WfMS
  - Entwicklung einer Workflow-Modellierungssprache zur Unterstützung des Kontrollflusses von „Workflow Patterns“

(Weitere) Nachteile (einfacher) Workflow Nets

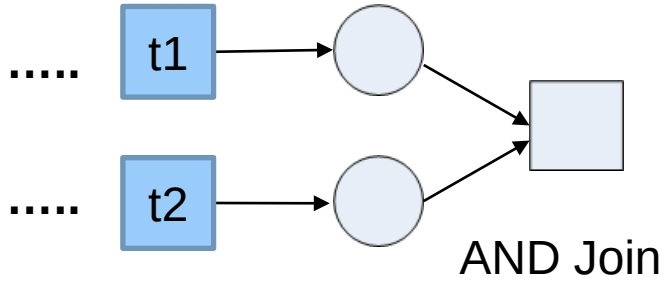
- Anforderungen der „Workflow Pattern“, die von (einfachen) Workflow Nets nur eingeschränkt unterstützt werden
  - Multiple Instanzen: mehrmaliges Ausführen einer Task
    - Anzahl der Ausführungen zu Modellierungszeit ggf. nicht bekannt bzw. ausführungsspezifisch
  - Synchronisationskonstrukte
    - Oder-Verzweigung (OR-Split) und Oder-Zusammenführung (OR-Join) bei gefärbten Petrinetzen aufwändig umzusetzen
  - Nicht-lokales Schaltverhalten
    - z.B. Modellierung von Abbruch-Kriterien

Beispiele Multiple Instanzen von Tasks

- Zeugen bei einem Versicherungsfall
  - Anzahl a priori nicht bekannt
- Reviewer (Gutachter) für einen Review-Prozess
  - Verantwortliche entscheiden im Zweifelsfall ad-hoc, ob weitere Reviews eingeholt werden
- Produkte bei einer elektronischen Bestellung
  - Anzahl a priori nicht bekannt

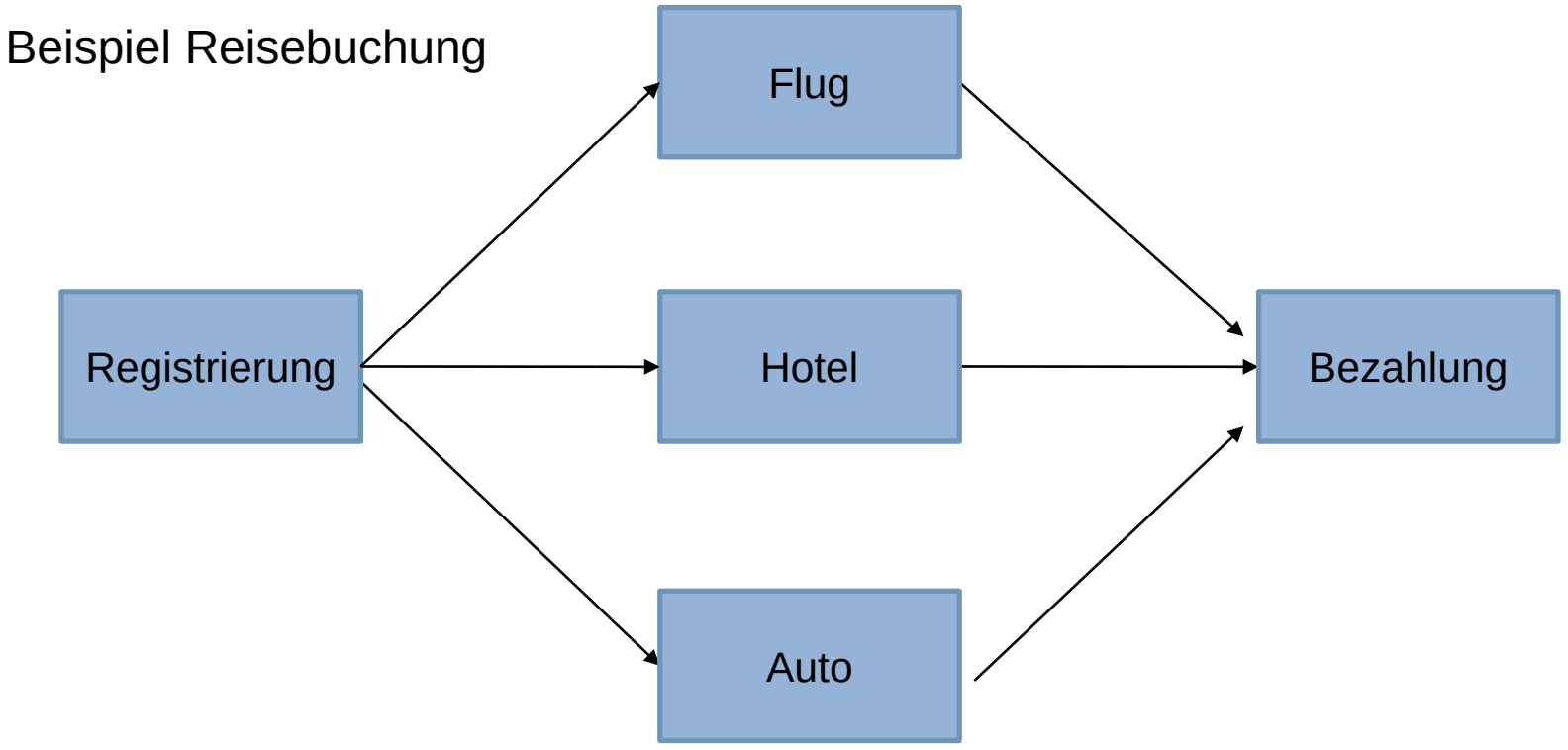
Multiple Instanzen von Tasks

- Bei einfachen Workflow Nets ist es die Aufgabe des Workflowdesigners, eine korrekte Synchronisation zu gewährleisten
  - Korrekte Synchronisation erfordert Verwaltung von:
    - Instanz-Informationen der einzelnen Fälle (Prozess-Instanzen), z.B. Instanz x, Instanz y
    - Tasks-Instanzen (Vater-Kind-Beziehungen), z.B. x.t1.1, x.t2.1, x.t1.2, x.t2.2, y.t2.1



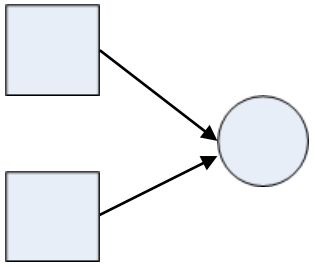
  - Dynamische Anzahl von Task-Instanzen
    - Modellierung z.B. über Counter

Beispiel Synchronisation

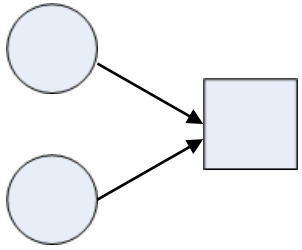


Synchronisation

- Parallele, optionale Teilprozesse
- Fragestellung der Synchronisation dieser Teilprozesse („Synchronising Merge“):
  - Keine Synchronisation (XOR-join)
  - Partielle Synchronisation (OR-join)
  - Synchronisation aller Teilprozesse (AND-join)
- Zur Modellierungszeit nicht bekannt
- Bei High-Level Petrinetzen hat der Workflowdesigner folgende Alternativen:
  - Durchreichen von Synchronisationsinformationen
  - Aktivieren der Verzweigungen mit boolschen Werten
  - Spezifikation der Synchronisations-Schaltlogik



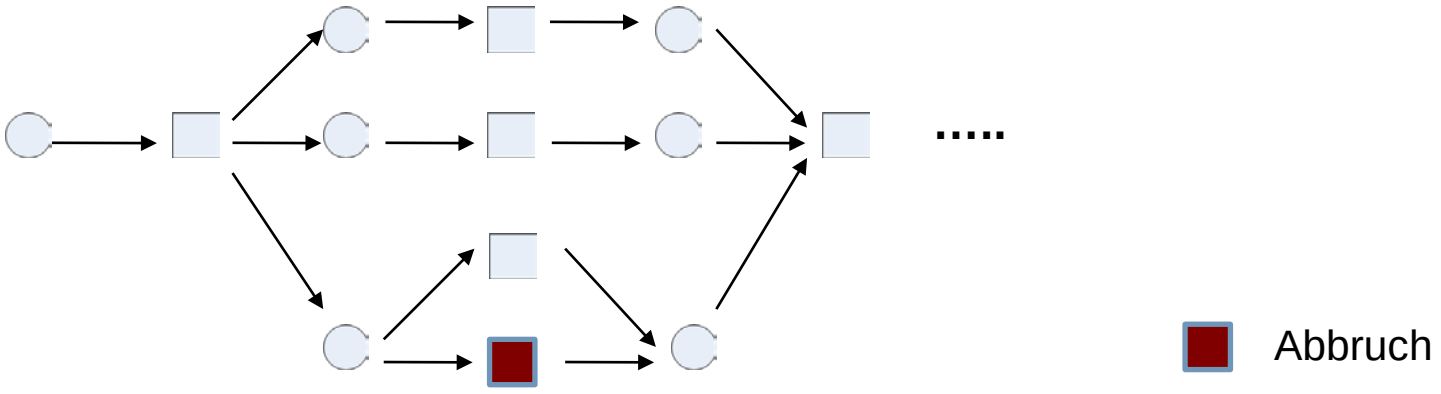
XOR Join



AND Join

Beispiel Nicht-lokales Schaltverhalten

- Modellierung von globalen Effekten
  - Beispiel „Abbruch“: Bei Aktivierung einer Task, die den Abbruch einer Workflow-Instanz spezifiziert, sollen auch andere, nebenläufige Teile des Workflows beendet werden



Nicht-lokales Schaltverhalten

- Darstellung von globalen Effekten mit Hilfe des lokalen Schaltverhaltens von Workflow Nets
  - Am Beispiel: Abbruch kann zu beliebigen Zeitpunkten eintreten, daher Berücksichtigung aller möglichen Markierungs-Kombinationen zum Entfernen von Token („vacuum cleaner“)
  - Impliziert sehr komplexe Netze

- Zusammenfassung: mit Workflow Nets können
  - Multiple Instanzen
  - Synchronisationskonstrukte (z.B. Instanzen, OR-Join)
  - Nicht-lokales Schaltverhalten

  ausgedrückt werden, allerdings führt dies zu sehr komplexen Modellen und liegt in der Verantwortung des Workflow Designers

Überblick YAWL

- Die Sprache YAWL (Yet Another Workflow Language) ist eine Erweiterung von Workflow Nets um
  - Multiple Instanzen
  - Oder-Verknüpfungen (OR-Joins)
  - Konstrukte zum Entfernen von Token („vacuum cleaner“)
  - Direkt verbundene Tasks

- Eine YAWL-Spezifikation ist eine Menge von Erweiterten Workflow Netzen (Extended Workflow Nets), die hierarchisch angeordnet sind (Baumstruktur)
- YAWL hat eigene Ausführungs-Semantik (Zustands-Übergangs-Systeme)
- Schwerpunkt Verhaltensaspekt, aber auch Datenaspekt, Organisations- und operationaler Aspekt

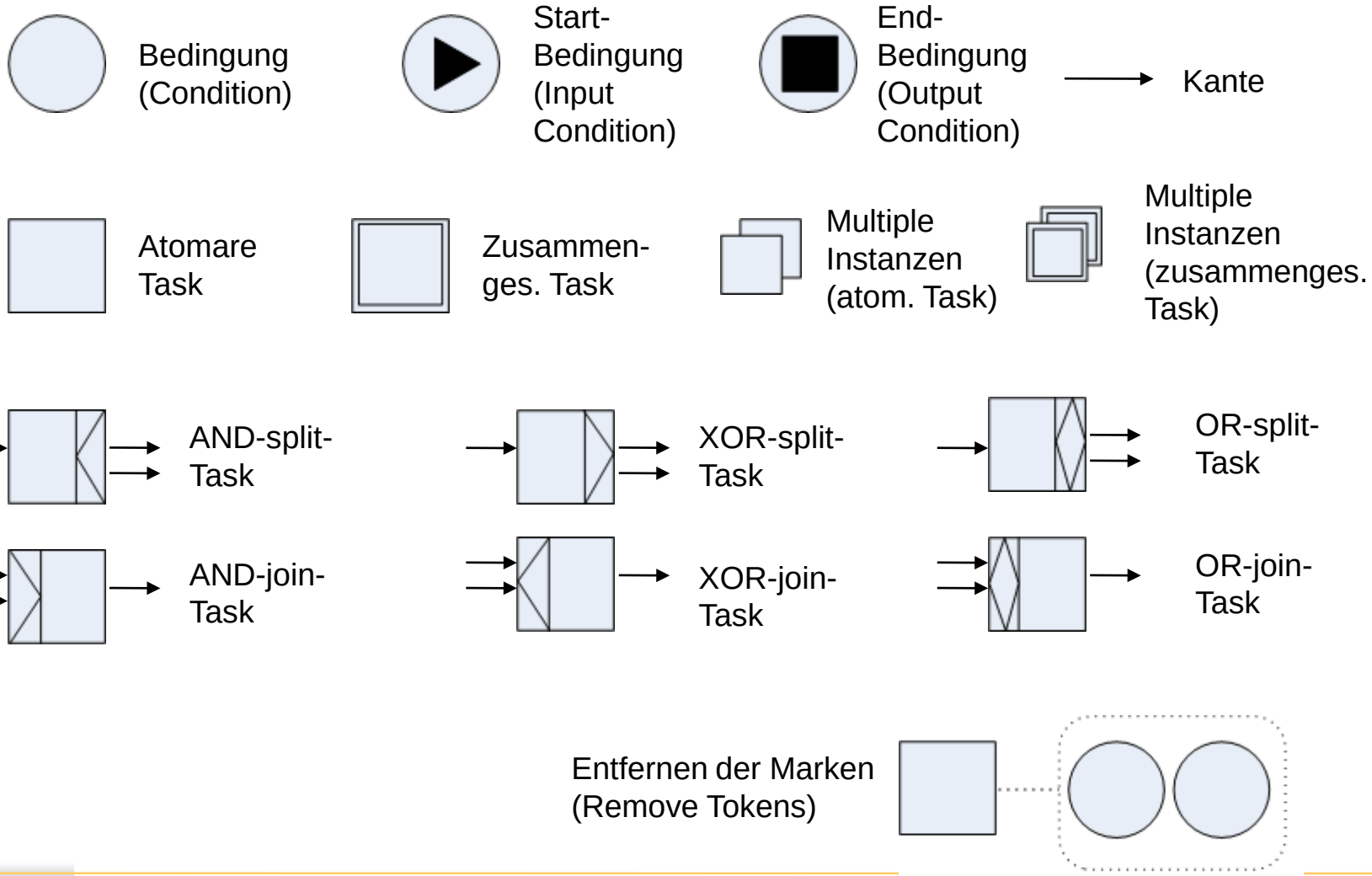
Definition Erweiterte Workflow Nets (I)

- Definition: Ein erweitertes Workflow Net (EWF) ist ein Tupel (C, i, o, T, F, *split*, *join*, *rem*, *nofi*):
  - C ist die Menge der Bedingungen („Conditions“)
  - i $\in$ C ist die Anfangsbedingung („initial condition“), o $\in$ C ist die Endbedingung („final condition“)
  - T ist die Menge von Tasks (Aktivitäten), C und T sind disjunkt
  - $F \subseteq (C - \{o\} \times T) \cup (T \times C - \{i\}) \cup (T \times T)$ ist die Fluß-Relation
  - Jeder Knoten des Graphen C $\cup$ T liegt auf einem Pfad von i nach o.

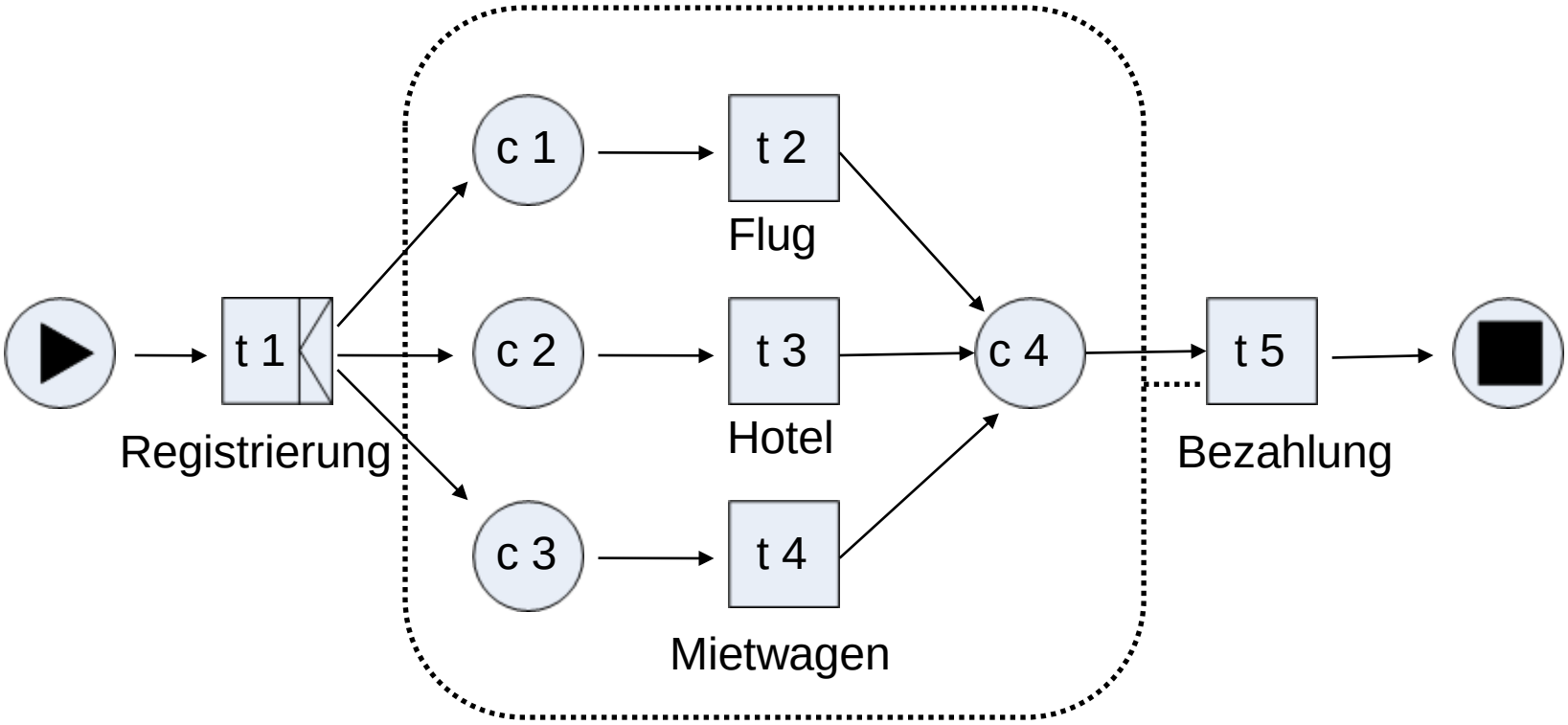
Definition Erweiterte Workflow Nets (II)

- Fortsetzung Definition: Ein erweitertes Workflow Net (EWF) ist ein Tupel (C, i, o, T, F, *split*, *join*, *rem*, *nofi*):
  - ....
  - *split*: T $\rightarrow \{AND, XOR, OR\}$ spezifiziert das Verzweigungs-Verhalten (split) einer Task
  - *join*: T $\rightarrow \{AND, XOR, OR\}$ gibt das Join-Verhalten einer Task an
  - *rem:* $T \rightarrow (T \cup C \setminus \{i, o\})$ bestimmt das Subnetz, dessen Token beim Ausführen einer Task entfernt werden
  - *nofi:* $T \rightarrow \mathrm{N} \times N^{\inf} \times N^{\inf} \times \{dynamic, static\}$ spezifiziert die Anzahl der Instanzen („number of instances“) einer Task (min, max, Schwellwert, dynamische oder statische Erzeugung von Instanzen)

Modellierungselemente von YAWL



Beispiel Reisebuchung in YAWL

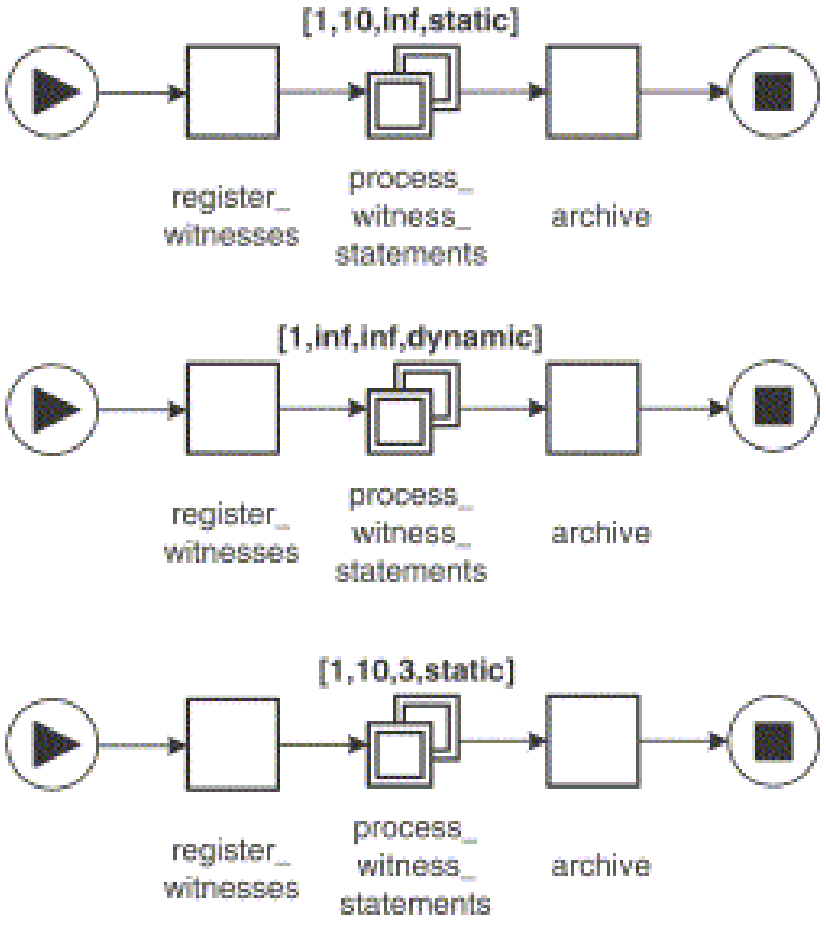


Beispiel: Spezifikation der Reisebuchung als EWF

EWF= (C, i, o, T, F, *split*, *join*, *rem*, *nofi*)

- C= {c1, c2, c3, c4}
- T= {t1, t2, t3, t4, t5}
- F = {(i,t1), (t1,c1), (t1,c2), (t1,c3), (c1,t2), (c2,t3), (c3,t4), (t2,c4), (t3,c4), (t4,c4), (c4,t5), (t5,o)}
- *split* (t1) = AND $split(t) \perp \forall t \in \{t2, t3, t4, t5\}$
- $join(t) \perp \forall t \in \{t1, t2, t3, t4, t5\}$
- *rem* (t5) = (c1, c2, c3, c4, t2, t3, t4) $rem(t) \perp \forall t \in \{t1, t2, t3, t4\}$
- $nofi(t) \perp \forall t \in \{t1, t2, t3, t4, t5\}$

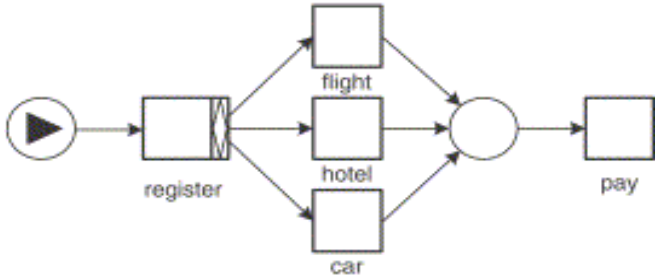
Beispiele Multiple Instanzen



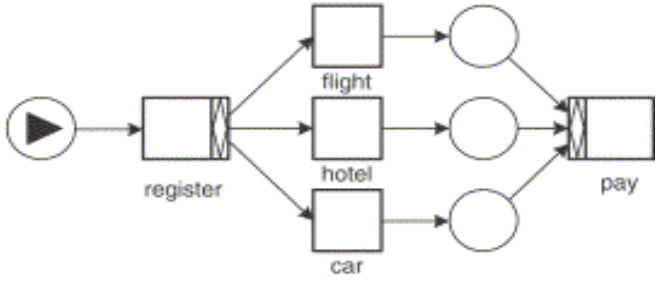
nofi:
Anzahl der Instanzen (number of instances) einer Task (min, max, Schwellwert, statisch/ dynamisch)

- Ohne Schwellwert: Task terminiert, wenn alle Instanzen ausgeführt sind;
- Mit Schwellwert: Task terminiert, wenn <*Schwellwert*> Instanzen beendet sind
- Statisch: alle Instanzen werden zu Beginn erzeugt
- Dynamisch: Instanzen können erzeugt werden, während andere Instanzen bereits ausgeführt werden

Beispiele Synchronisation (I)

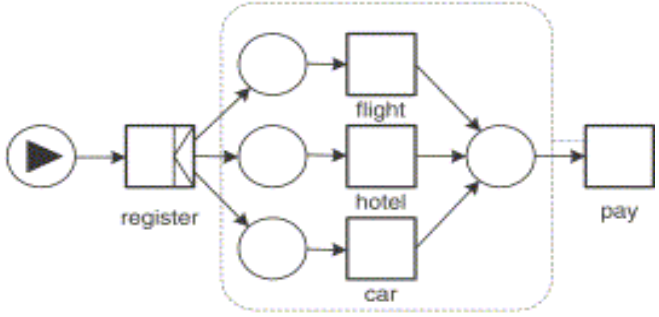


Oder-Verzweigung
Wann wird Task „Pay“ ausgeführt?



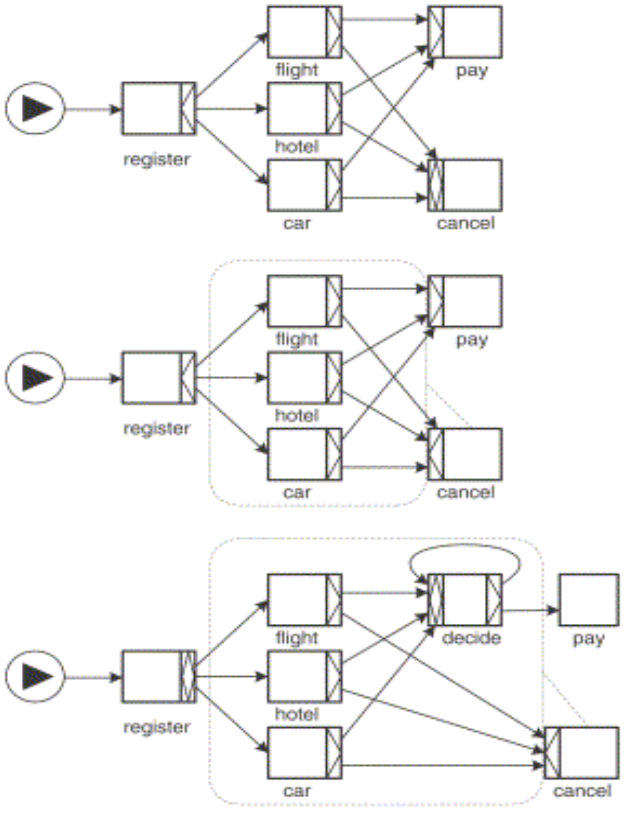
Oder-Verzweigung, Oder-Verknüpfung
- Oder-Verknüpfung wartet auf weitere (eingehende) Bedingungen, die derzeit nicht erfüllt sind, aber in der Zukunft

Wann wird Task „Pay“ ausgeführt?



UND-Verzweigung
Wann wird Task „Pay“ ausgeführt?

Beispiele Synchronisation (II)

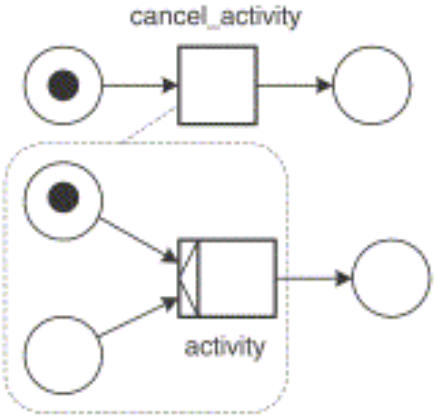


UND-Verzweigung, XOR-Verzweigung, UND-Verknüpfung, ODER-Verknüpfung
Wann wird Task „Pay“ ausgeführt?

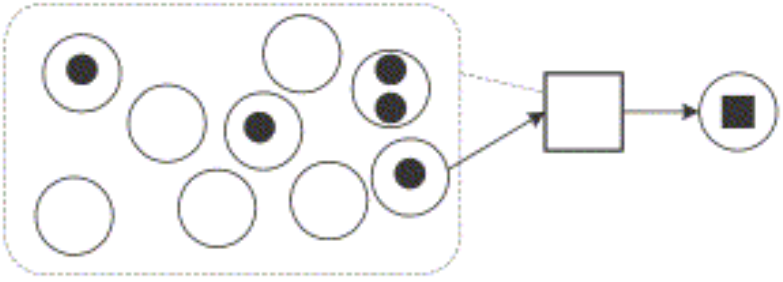
UND-Verzweigung, XOR-Verzweigung, UND-Verknüpfung, XOR-Verknüpfung
Wann wird Task „Pay“ ausgeführt?
Was passiert, wenn Task „Cancel“ ausgeführt wird?

ODER-Verzweigung, XOR-Verzweigung
ODER-Verknüpfung (Task „Decide“)
Wann werden die Tasks „Cancel“ und „Pay“ ausgeführt?

Beispiele Entfernen von Token



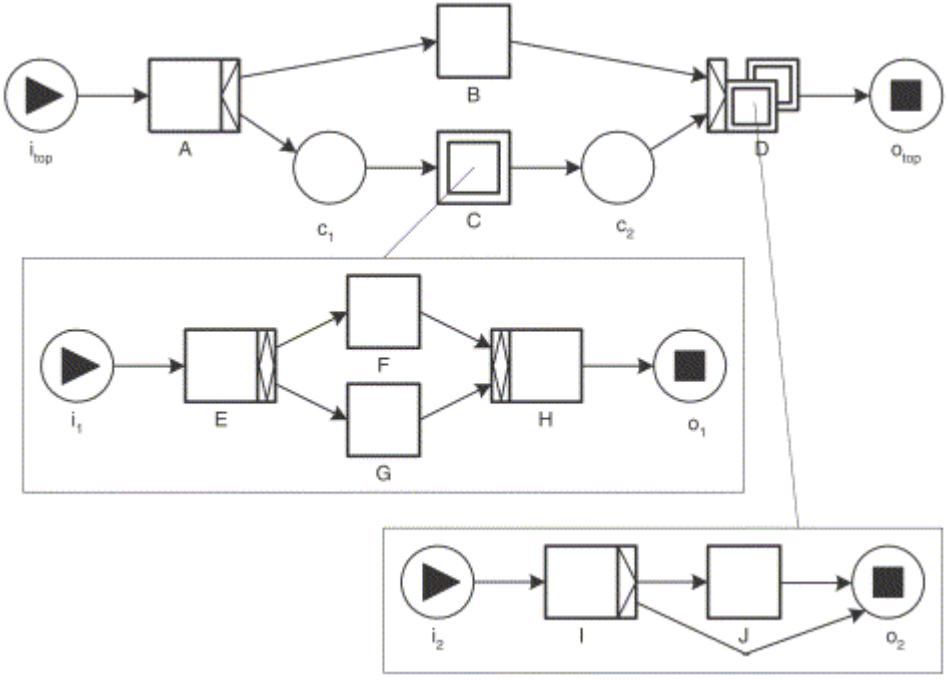
Entfernen der Token



Zusätzlich zum Entfernen der Token wird ein Token übernommen

YAWL Workflow Spezifikation

- Bestandteile
  - Menge von Extended Workflow Nets (EWFs) Spezifikationen
  - Beschreibung der Zusammenhänge (Hierarchie) dieser EWFs
    - Einbindung von Subprozessen (EWFs): Kopplung an zusammengesetzte Tasks



Definition YAWL Workflow Spezifikation

- Eine Workflow Spezifikation *S* ist ein Tupel $(Q, top, T^{\diamond}, map)$ mit:

Q Menge von EWF Nets

$top \in Q$ Top level Workflow, Wurzelknoten

$T^{\diamond} = \cup_{N \in Q} T_N$ Menge von Tasks (atomar oder zusammengesetzt)

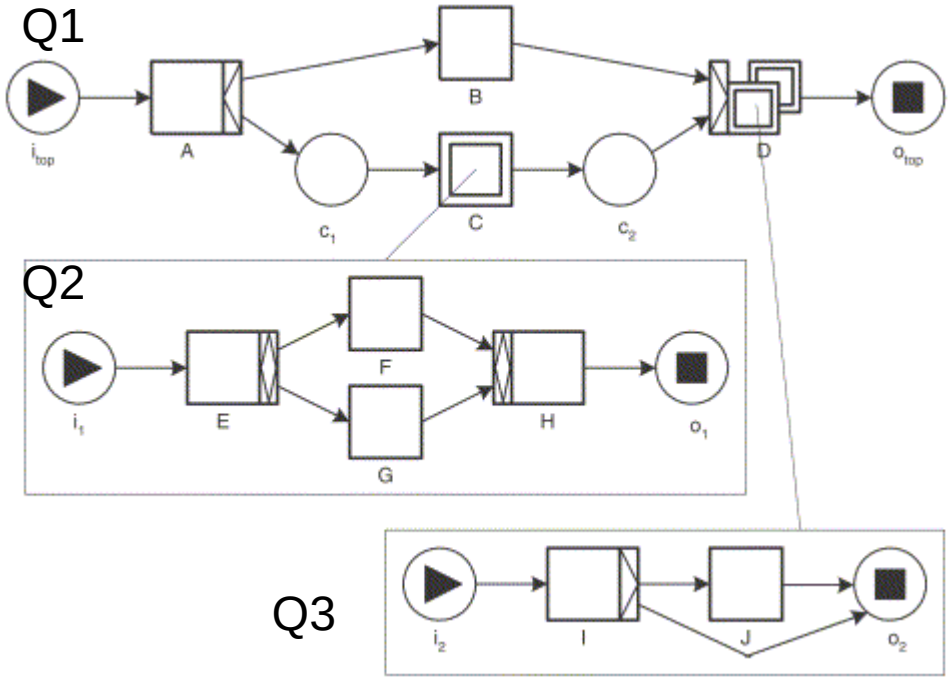
$\forall N_1 N_2 \in Q \quad N_1 \neq N_2 \Rightarrow (C_{N_1} \cup T_{N_1}) \cap (C_{N_2} \cup T_{N_2}) = \{\}$

d.h. keine Namensüberlappungen

$map : T^{\diamond} \rightarrow Q \setminus \{top\}$ Abbildung aller zusammengesetzten Tasks auf ein EWF Net (Sub-Prozess)

Die Relation $\{(N_1, N_2) \in Q \times Q \mid \exists_{t \in dom(map_{N_1})} map_{N_1}(t) = N_2\}$ ist ein Baum

Beispiel für eine Workflow Spezifikation in YAWL



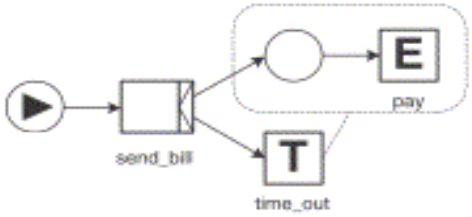
S= $(Q, top, T^{\diamond}, map)$

- Q={Q1,Q2,Q3}
- top=Q1
- $T^{\diamond}$= {A, B, C, D, E, F, G, H, I, J}
- *map*(C)=Q2; *map*(D)=Q3;
- $\forall T \in T^{\diamond} \setminus \{C, D\}$: $map(T) \perp$

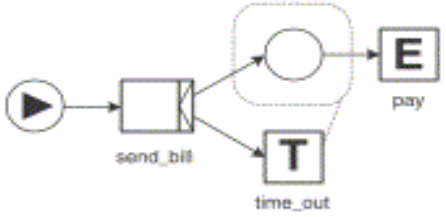
Beispiele zur Modellierung von Zeit in YAWL

Darstellung von Zeit und von Ereignissen

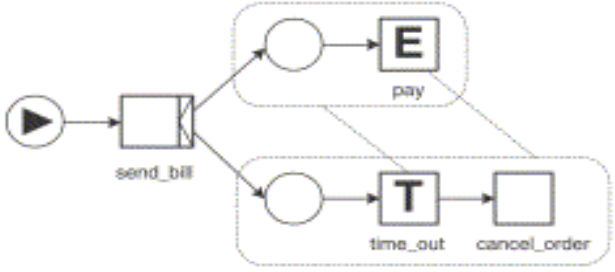
- Keine expliziten Sprachkonstrukte
- Einsatz von Tasks („Event Task“, „Timed Task“)



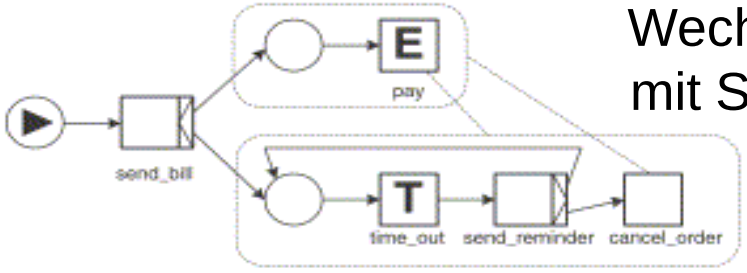
Abbruch von „pay“ (ggf. bereits gestartet)



Abbruch von „pay“ (noch nicht gestartet)



Wechselseitiger Abbruch (weitere Task durchzuführen)

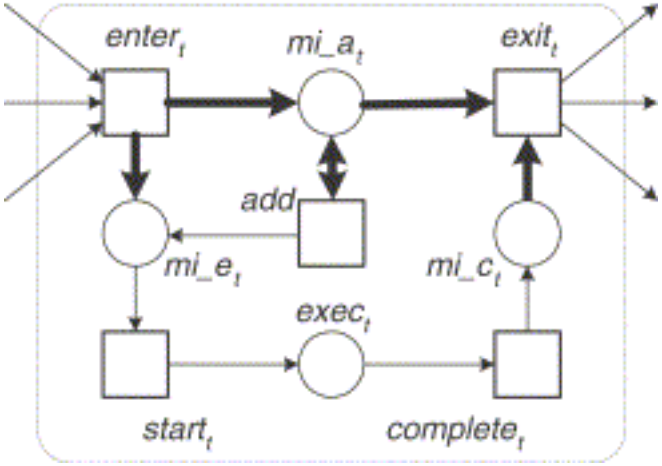


Wechsels. Abbruch mit Schleife

Ausführungssemantik von YAWL

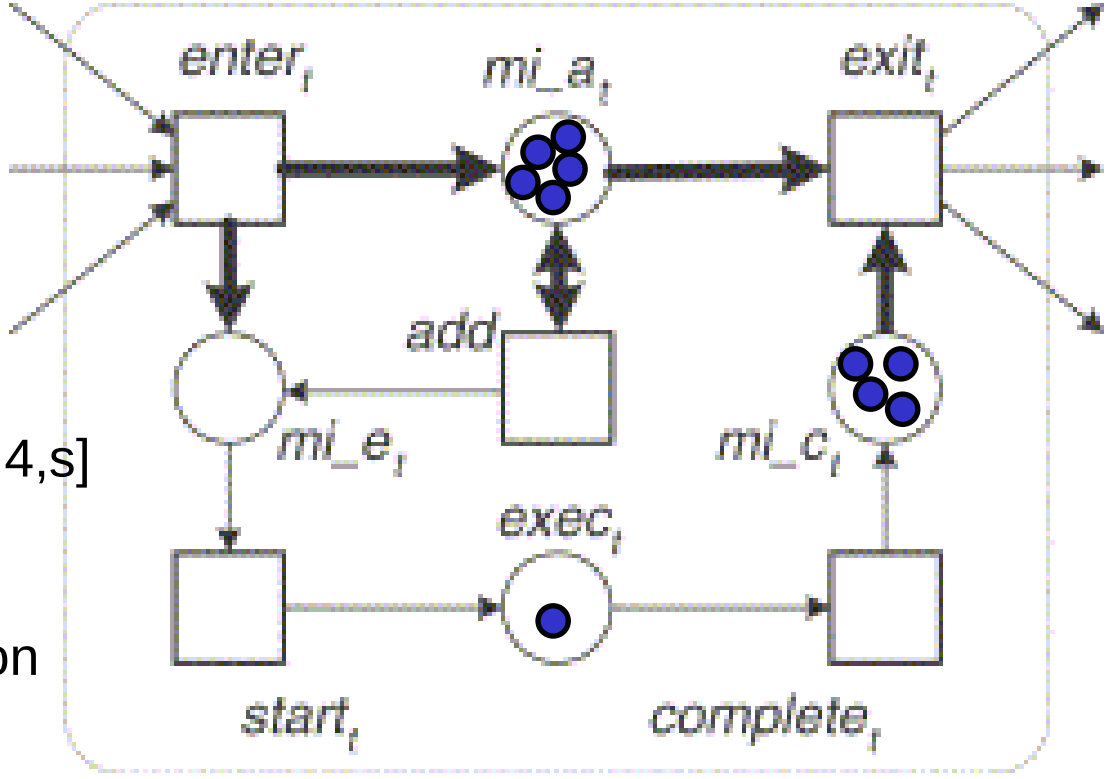
Task-Instanzen

- Generelle Idee: jede Task wird als Zustands-Übergangssystem („state/transition system“) dargestellt
- Zustände von Tasks
  - mi_a_t (active): Task-Instanz ist aktiv
  - mi_e_t (enabled): Instanz ist bereit
  - mi_c_t (completed): Instanz ist fertig
  - $exec_t$ (exec): Instanz führt aus
- Übergänge von Tasks
  - enter, add (multiple Instanzen), start, complete, exit



Beispiel Ausführungssemantik von Tasks

- Task mit multiplen Instanzen; nofi: [4,6,4,s]
- Erzeugung von 5 Instanzen (mi_a_t)
- Wann kann Transition „exit“ schalten?



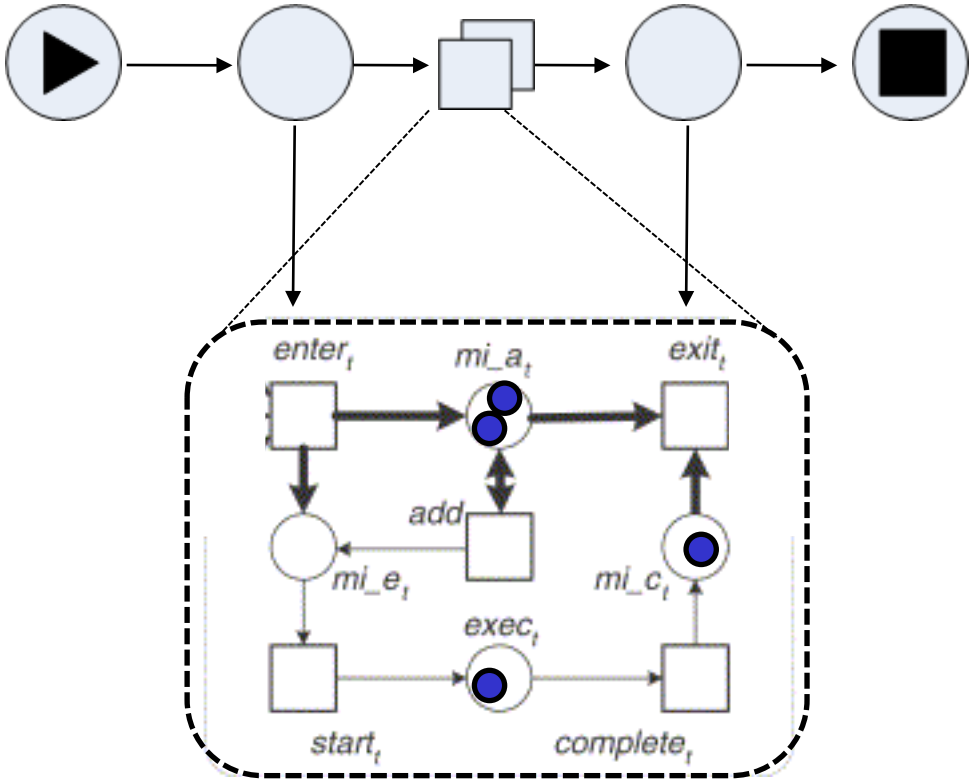
Ausführungssemantik in YAWL (II)

Prozess-Instanzen

- Eindeutige Identifikation von
  - Prozess-Instanzen
  - Task-Instanzen
- Vater-Kind Beziehungen zwischen Prozess-Instanzen und Task-Instanzen
  - Ermöglicht Zuordnung von Task-Instanzen zu Prozess-Instanzen
- Spezifikation des Workflow Zustandes („state s“)
  - Einfügen von künstlichen Bedingungen, sofern nötig
  - Verwaltung der Zustände (states) der einzelnen Tasks (exec, active, enabled, completed)

Beispiel Ausführungssemantik in YAWL

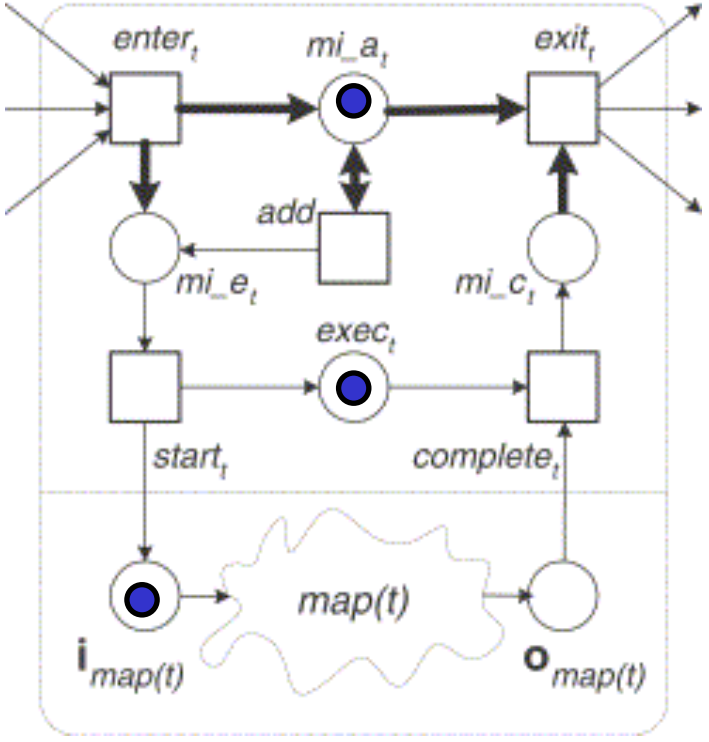
Task-Instanzen einer Prozessinstanz



Ausführungssemantik in YAWL (III)

Verschachtelte Prozesse (zusammengesetzte Tasks)

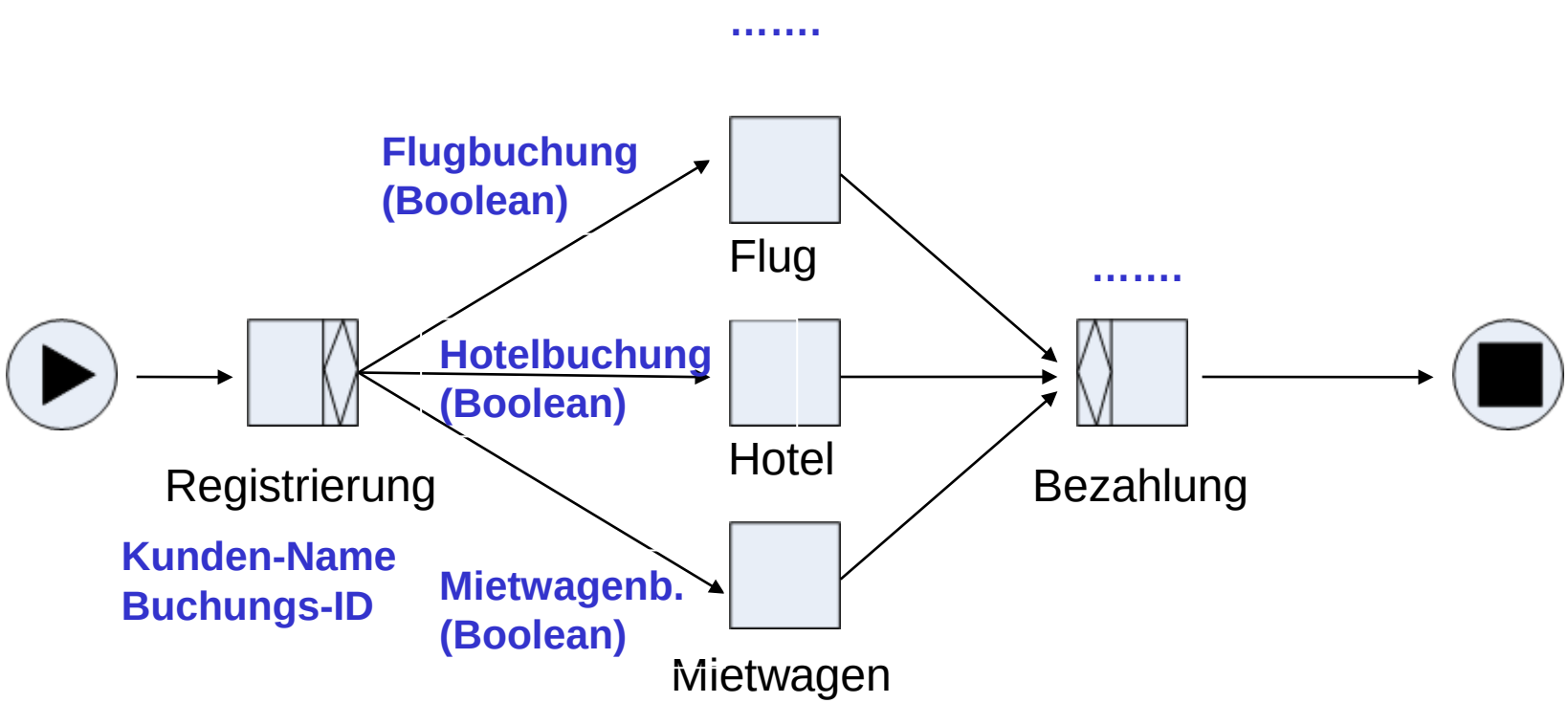
- bei zusammengesetzten Tasks zwei Token-Bereiche
- $i_{map(t)}$ und $o_{map(t)}$ initiale und letzte Bedingung des Sub-Prozesses



Datenaspekt in YAWL

- Verwaltung von
  - Produktionsdaten („Stammdaten“ der Workflow-Instanz), z.B. Kundendaten bei Reisebuchung, Kunden-ID
  - Steuerungsdaten (Management der Workflow-Instanz), z.B. Kunde möchte Flug und Hotel buchen, Prozess- und Task-Instanzen, Zustandsvariablen
- XML-basiert
  - Nutzung der Mechanismen von XML (z.B. XPath, XQuery)
  - Spezifikation von (benutzerdefinierten) Datentypen
  - Flexibles Austauschformat
- Ansatz:
  - Lokale Variablen
  - Austausch von Variablen mit Subprozessen

Beispiel Datenaspekt Reisebuchung

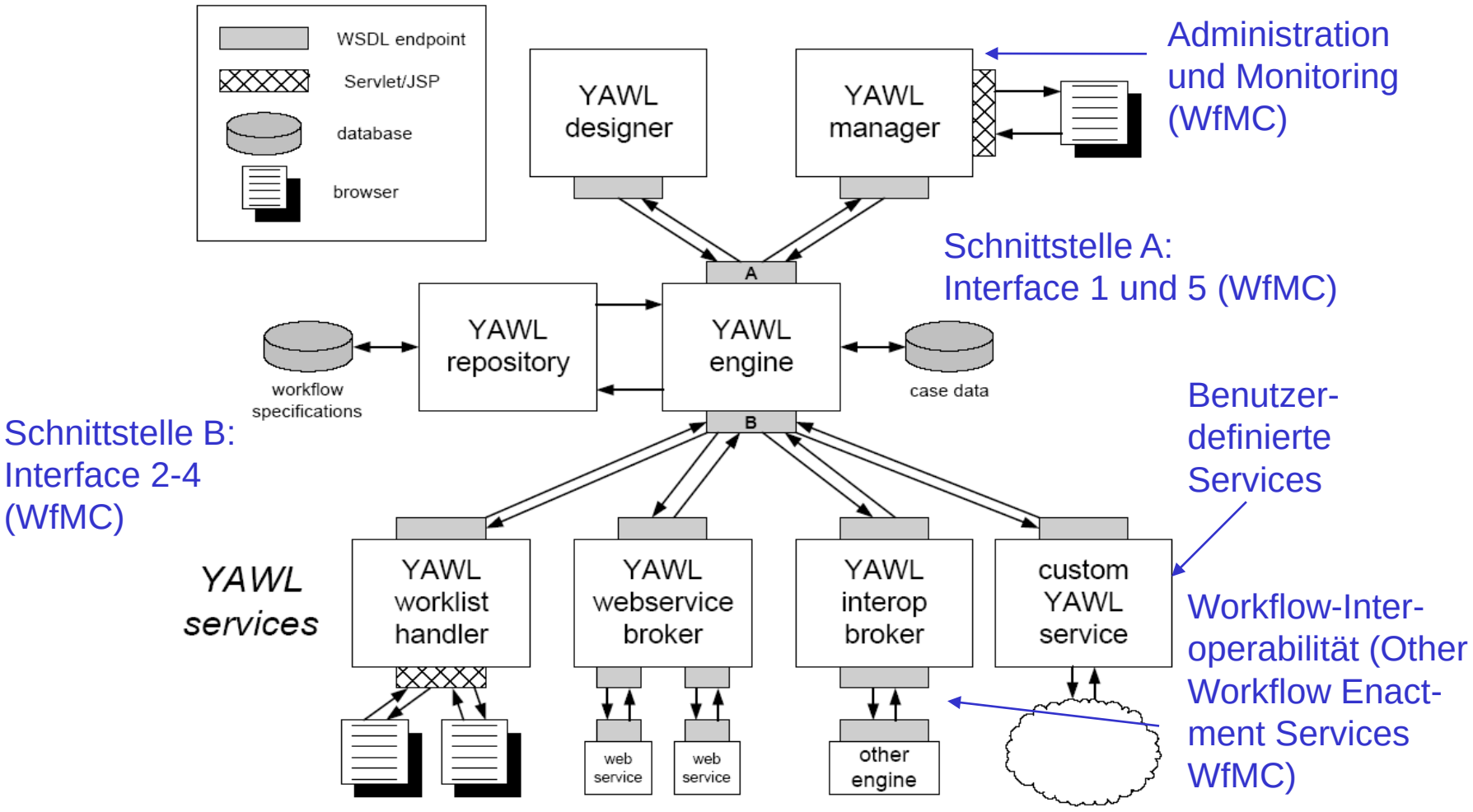


Beispiel Deklaration von lokalen Variablen

```
<rootNet id="make_trip">
<localVariable name="customer">
<type>xs:string</type>
<initialValue>Type name of customer</initialValue> </localVariable>
<localVariable name="payment_account_number">
<type>xs:string</type> </localVariable>
 ...
<localVariable name= "want_flight">
 <type>xs:boolean</type> </localVariable>
<localVariable name= "want_hotel">
<type>xs:boolean</type> </localVariable>
<localVariable name= "want_car">
<type>xs:boolean</type></localVariable>

<localVariable name="flightDetails">
<type>xs:string</type></localVariable>
...
```

YAWL System


WSDL endpoint
Servlet/JSP
database
browser
YAWL designer
YAWL manager
Administration und Monitoring (WfMC)
Schnittstelle A: Interface 1 und 5 (WfMC)
A
YAWL repository
YAWL engine
workflow specifications
case data
B
Schnittstelle B: Interface 2-4 (WfMC)
Benutzer-definierte Services
YAWL services
YAWL worklist handler
YAWL webservice broker
YAWL interop broker
custom YAWL service
web service
web service
other engine
Workflow-Inter-operabilität (Other Workflow Enact-ment Services WfMC)

Bewertung YAWL

- Zusammenfassung
  - Graphische Repräsentationsform, ähnlich zu Workflow Nets
  - Präzise Ausführungssemantik
  - Formale Analyse von Workflows (z.B. Soundness)
  - Zeitverhalten von Tasks als Zustands/ Übergangssysteme (nicht zeitlos)
  - Ausdrucksmöglichkeit für nicht-lokales Verhalten
  - Vielseitige Unterstützung von multiplen Instanzen
  - Akademischer Ansatz
  - Werkzeugunterstützung
  - newYAWL: weitere Kontrollfluß-Elemente, detaillierter Ressourcenaspekt

Exemplarische Fragen zu Kapitel 5

- Erläutern Sie die Vor- und die Nachteile von (einfachen) Workflow Nets.
- Was ist YAWL? Wie unterscheidet sich YAWL von einfachen Workflow Nets?
- Beschreiben Sie die funktionalen Besonderheiten von YAWL.
- Erläutern Sie die Synchronisationskonstrukte XOR-Join, OR-Join und AND-Join in YAWL.
- Spezifizieren Sie ein Erweitertes Worflow Net.
- Geben Sie eine YAWL-Workflow-Spezifikation anhand eines einfachen Beispiels (mindestens ein Sub-Workflow) an.
- Beschreiben Sie die Parameter, die YAWL zum Ausführen von multiplen Instanzen zur Verfügung stellt und erläutern Sie die Ausführungssemantik von multiplen Task-Instanzen in YAWL am Beispiel nofi(2,8,6,dynamic).

Ergänzende Literatur zu Kapitel 5

- **YAWL: Yet Another Workflow Language:** W.M.P. van der Aalst und A.H.M. ter Hofstede. Information Systems, 30(4):245-275, 2005
- **Design and Implementation of the YAWL system:** W.M.P. van der Aalst, L. Aldred, M. Dumas, and A.H.M. ter Hofstede. Proceedings of the 16th International Conference on Advanced Information Systems Engineering (CAiSE 04), Riga, Latvia, Juni 2004, Springer.
- http://www.yawlfoundation.org