

# 1 Übungsaufgaben mit Lösungen – YAWL-System

## 1.1 Beispiel für das YAWL System (Reisebuchungsszenario)

Um den Umgang mit YAWL und insbesondere mit dem YAWL-editor zu illustrieren soll ein Beispiel bearbeitet werden. Im Folgenden wird das Szenario einer Reisebuchung in einem Reisebüro betrachtet. Eine Buchung soll wie folgt ablaufen:

- Der Kunde wird zunächst mit seinen persönlichen Daten von einem Bearbeiter registriert.
- Eine Buchung besteht aus der Buchung eines Fluges, eines Hotels und eines Mietwagens. Flug, Hotel und Mietwagen können unabhängig voneinander gewählt werden. Es soll einerseits möglich sein z. B. nur ein Hotel zu buchen. Andererseits sollen auch Kombinationen von Flug-, Hotel- und Mietwagenbuchungen möglich sein.
- Sind alle Einzelbuchungen abgeschlossen, so müssen alle Buchungsdaten zusammengefügt werden und der Auftrag bezahlt werden.

Aufeinander aufbauend soll das Szenario in mehreren Aufgaben modelliert werden. In den folgenden Aufgaben wird insbesondere auf den Funktions-, Verhaltens- und den Datenaspekt eingegangen. Die Aufgaben können unter zu Hilfe nahme des YAWL-worklist-handlers getestet werden. Der Organisationsaspekt, der in diesem Beispiel nicht behandelt wird, wird ausführlich in dem Beispiel für MQ Workflow behandelt werden.

### 1.1.1 Aufgabe 1

#### **Aufgabenstellung:**

In Aufgabe 1 sollen die Teilaufgaben (tasks) einer Reisebuchung modelliert werden. Die folgenden Tasks sind zu erstellen:

- Registrierung des Kunden (Task Registrieren)
- Buchung eines Fluges (Task Flug)
- Buchung eines Hotels (Task Hotel)
- Buchung eines Mietwagens (Task Mietwagen)

In einem nächsten Schritt ist der Kontrollfluss zu modellieren. Es soll möglich sein Flug, Hotel oder Mietwagen einzeln zu buchen. Zudem soll es die Möglichkeit geben Kombinationen von allen zu buchen.

#### **Lösung:**

Abbildung 7 zeigt die Lösung von Aufgabe 1. Task Registrieren enthält einen OR-split. Task Bezahlen führt den Kontrollfluss durch einen OR-join zusammen.

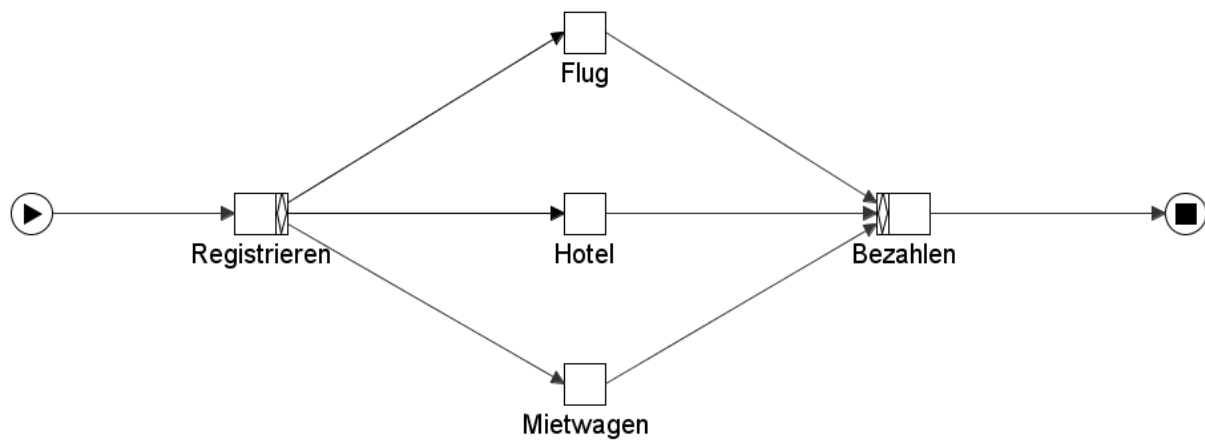


Abbildung 7 : Lösung Aufgabe 1

## 1.1.2 Aufgabe 2

### Aufgabenstellung:

Aufgabe 1 ist so zu erweitern, dass der Benutzer während der Task Registrieren entscheiden kann, ob er einen Flug, ein Hotel und/oder einen Mietwagen buchen will. Abhängig von seiner Entscheidung sollen ihm nur die Teilaufgaben angeboten werden, die er gewählt hat.

### Lösung:

Um den Kontrollfluss zu steuern bietet YAWL die Möglichkeit Kontrollflusspfeile mit Prädikate über Netzvariablen zu versehen. Das Prädikat wird ausgewertet und je nach Ergebnis wird ein Zweig eingeschlagen oder fallen gelassen.

Es bietet sich an boolesche Taskvariablen (Fluggewünscht, Hotelgewünscht, Mietwagengewünscht) in der Task Registrieren zu erstellen, die durch den Besitzer gesetzt werden können. Diese werden auf ebenfalls zu erstellende Netzvariablen gemappt. Das Verzweigungsprädikat fragt den Inhalt der Netzvariablen ab. Für die Abfrage ob ein Flug gebucht werden soll ergibt sich so das Prädikat (`/Aufgabe2/Fluggewuenscht/text()='true'`).

## 1.1.3 Aufgabe 3

### Aufgabenstellung

Im nächsten Schritt sollen nun Daten, den Kunden, den Flug, das Hotel und den Mietwagen betreffend aufgenommen werden. Dazu müssen zunächst entsprechende benutzerdefinierte Datentypen angelegt werden.

Von einem Kunden soll der Name und Vorname sowie sein Geburtsdatum aufgenommen werden.

Eine Flugbuchung soll den Abflugort und den Ankunftsart beinhalten. Zudem die Anzahl der Tickets und die Flugklasse.

Eine Mietwagenbuchung enthält lediglich die Typenbezeichnung des Wagens

In einer Hotelbuchung soll es möglich sein verschieden Zimmer mit unterschiedlicher Bettenzahl zu buchen. Zudem soll jeweils gewählt werden können ob ein Frühstück gewünscht ist.

### **Lösung:**

Die folgenden Datentypen sind zu erstellen:

```
<schema xmlns="http://www.w3.org/2001/XMLSchema">

<complexType name='Flugbuchung'>
<sequence>
  <element name="von" type="string"/>
  <element name="nach" type="string"/>
  <element name="AnzahlTickets" type="double"/>
  <element name="Flugklasse" type="string"/>
</sequence>
</complexType>

<complexType name="Hotelbuchung">
<sequence>
  <element name="Hotelname" type="string"/>
  <element name="Zimmer">
    <complexType>
      <sequence>
        <element name="AnzahlBetten" type="double"/>
        <element name="Fruehstueck" type="boolean"/>
      </sequence>
    </complexType>
  </element>
</sequence>
</complexType>

<complexType name='Mietwagenbuchung'>
<sequence>
  <element name="Typbezeichnung" type="string"/>
</sequence>
</complexType>

<complexType name='Kunde'>
<sequence>
  <element name="Name" type="string"/>
  <element name="Vorname" type="string"/>
  <element name="Geburtsdatum" type="date"/>
</sequence>
</complexType>
</schema>
```

Es müssen nun noch Taskvariablen der oben aufgeführten Typen in den jeweiligen Tasks erstellt werden. Dadurch wird modelliert, dass der Benutzer an dieser Stelle Daten einzugeben hat.

### 1.1.4 Aufgabe 4

#### Aufgabenbeschreibung:

In dieser Aufgabe soll der Umgang mit composite tasks erlernt werden. Es soll ein neues Netz erstellt werden, in dem die Mietwagenbuchung modelliert wird. Eine Mietwagenbuchung soll dabei wie folgt aussehen. Zunächst wird überprüft ob ein gültiger Führerschein vorliegt. Ist dies der Fall, so wird die Buchung durchgeführt. Anderenfalls passiert nichts.

#### Lösung:

Die Task Mietwagen wird durch die Composite Task SubNetMietwagen ersetzt. Die entsprechende Modellierung ist in den Abbildungen 8 und 9 ersichtlich.

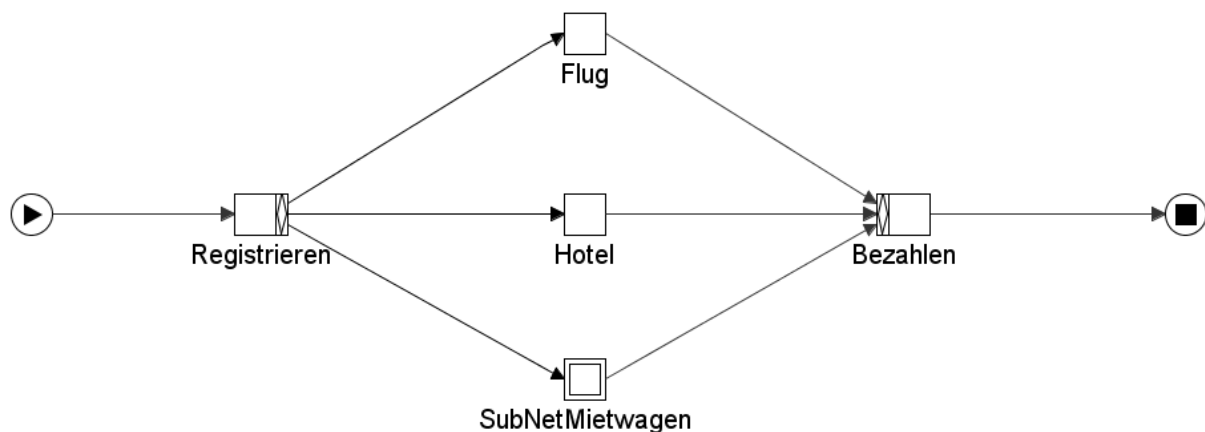


Abbildung 8: Lösung Aufgabe 4

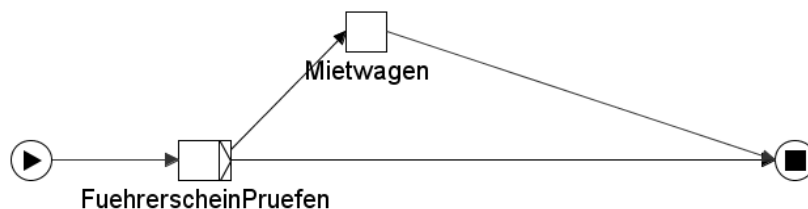


Abbildung 9: Lösung Aufgabe 4: SubNetMietwagen

## 1.1.5 Aufgabe 5

### Aufgabenbeschreibung:

Aufgabe 5 gliedert sich in zwei Teile:

Analog zu Aufgabe 4 soll eine weitere Composite Task für Task Hotel erstellt werden. Dabei soll eine Hotelbuchung nun folgendermaßen ablaufen. Zunächst werden alle gewünschten Zimmer gewählt. Anschließend sollen die Daten für jedes Zimmer noch einmal einzeln überprüft werden. Dabei soll eine Multiple Task zum Einsatz kommen.

Der zweite Teil der Aufgabe besteht darin sämtlich gesammelten Buchungsdaten in der Task Bezahlen zusammenzuführen und so eine Rechnungserstellung zu ermöglichen.

### Lösung:

Die Modellierung der Netze ist in den Abbildungen 10,11 und 12 ersichtlich.

Die Erstellung der Composite Task SubNetHotel geschieht analog zu Aufgabe 4. SubNet Hotel soll eine Hotelbuchung modellieren und besitzt somit eine Netzvariable Hotelbuchung vom Typ Hotelbuchung. Task ZimmerPruefen besitzt eine Taskvariable Zimmer vom Typ Zimmer. Es sind nun Anfragen für die Multiple Task ZimmerPruefen zu schreiben die jeder Taskvariable Zimmer ein Zimmer aus den in der Task „Hotelzimmer wählen“ ausgesuchten und in der Netzvariable Hotelbuchung gespeicherten Zimmern zuweist. Anschließend müssen die bearbeiteten Zimmer wieder in Hotelbuchung gespeichert werden. Die entsprechenden Abfragen sind in Abbildung 13 zu sehen.

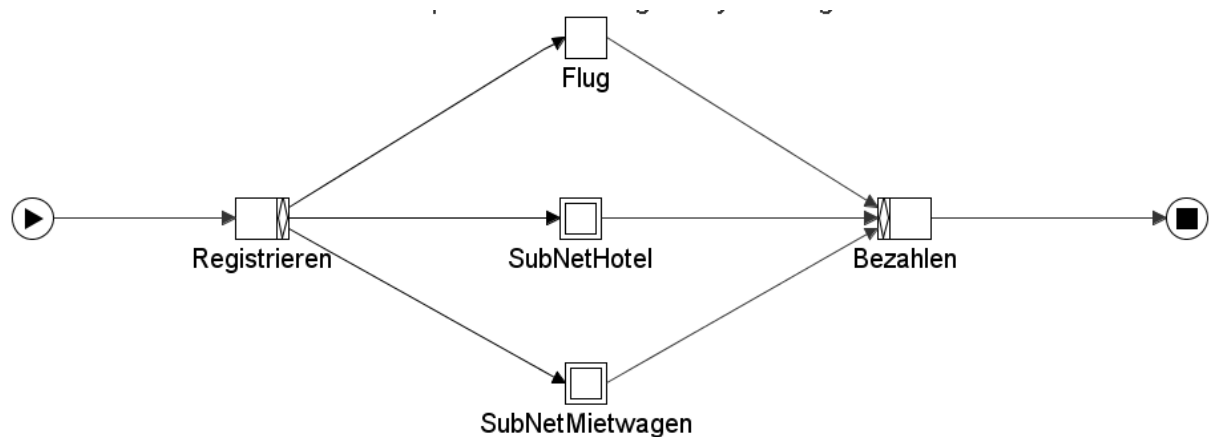


Abbildung 10

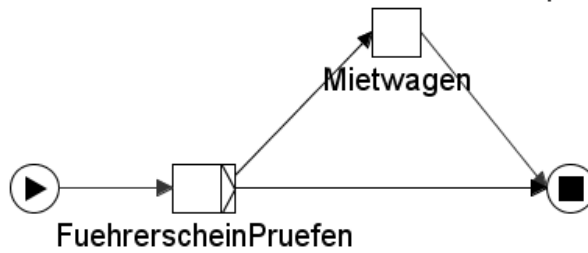


Abbildung 11

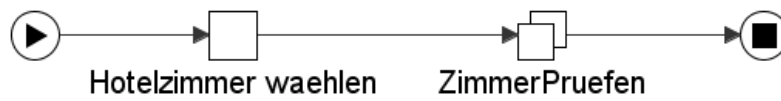


Abbildung 12

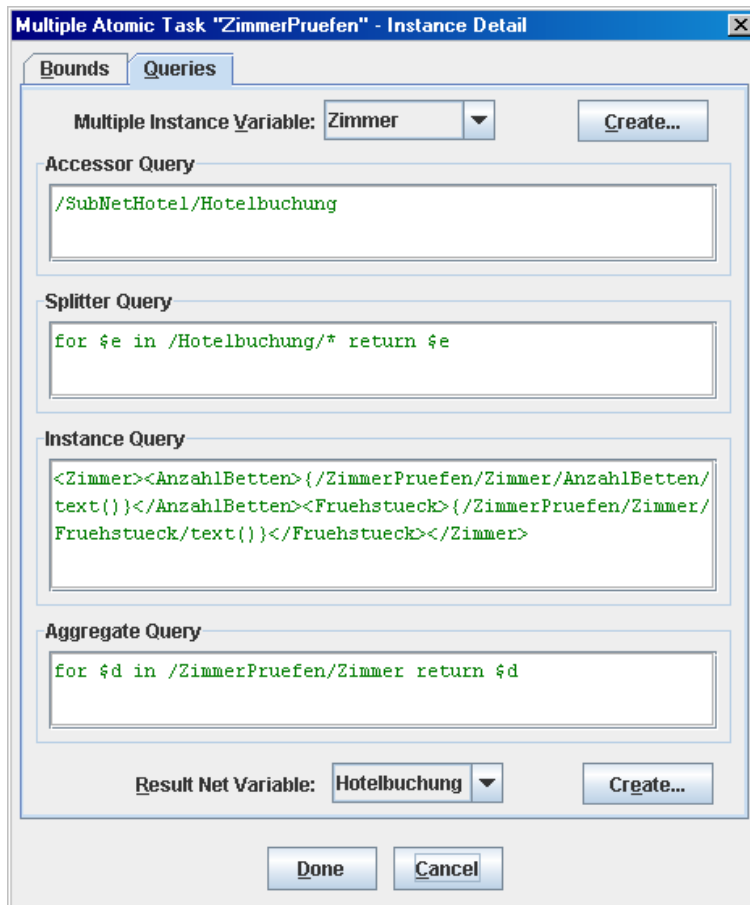


Abbildung 13

Für den zweiten Teil der Aufgabenstellung müssen Netzvariablen für alle Buchungen und die Kundendaten angelegt werden. Diese werden durch die entsprechenden Tasks durch Mappen ihrer Taskvariablen auf die Netzvariablen gesetzt. In der Task Bezahlen werden

Taskvariablen für die Buchungen und den Kunden angelegt, welche durch die Netzvariablen gesetzt werden.