

# 1 YAWL – Yet Another Workflow Language

Das YAWL Workflow-Management-System wurde von Wil van der Aalst und seinem Team an der Eindhoven University of Technology entwickelt. Das System ist in seiner jetzigen Version noch nicht in allen Modulen fertig implementiert, stellt aber grundlegende Funktionen bereit. Dem System liegt eine neuentwickelte Workflow-Sprache zu Grunde, die Mängel an bestehenden Workflow Sprachen ausgleichen soll. Ausgehend von Petri-Netzen wurde die YAWL Sprache um zusätzliche Mechanismen erweitert um die Anforderungen von WfMS zu erfüllen.

## ***1.1 Die Workflow Sprache YAWL***

Eine in YAWL geschriebene Workflow-Spezifikation besteht aus einer Menge von Prozessdefinitionen. Eine Prozessdefinition besteht aus tasks und conditions. Conditions korrespondieren zu Stellen in Petri-Netzen. Jede Prozessdefinition besitzt eine „Input condition“ und eine „Output Condition“, die Start und Ende eines workflow bzw. subworkflows kennzeichnen. Tasks entsprechen Transitionen in Petri-Netzen. Sie können entweder atomic tasks, die eine einzelne Aktivität darstellen, oder composite tasks sein. Eine composite task wird durch eine Prozessdefinition spezifiziert. Sowohl task als auch composite task können mehrfache Instanzen haben. YAWL besitzt eine Reihe von Verzweigungs- und Vereinigungstasks. Diese sind AND-split task, XOR-split task, OR-split task sowie AND-join task, XOR-join task, OR-join task. In Abbildung 4 sind die Modellierungselemente von YAWL dargestellt.

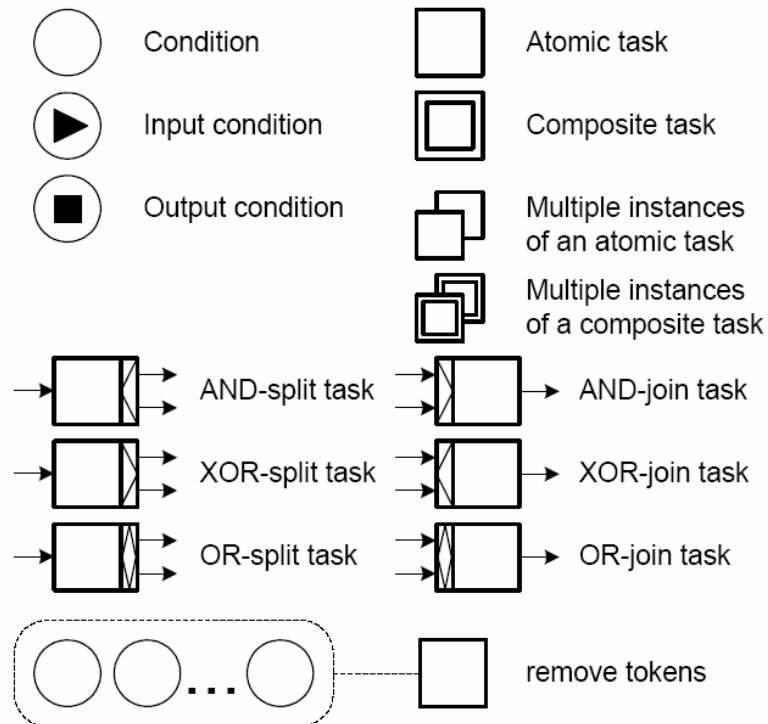


Abbildung 4: Modellierungselemente von YAWL [1]

Das Beispiel in Abbildung 5 soll die Modellierung mit YAWL illustrieren.

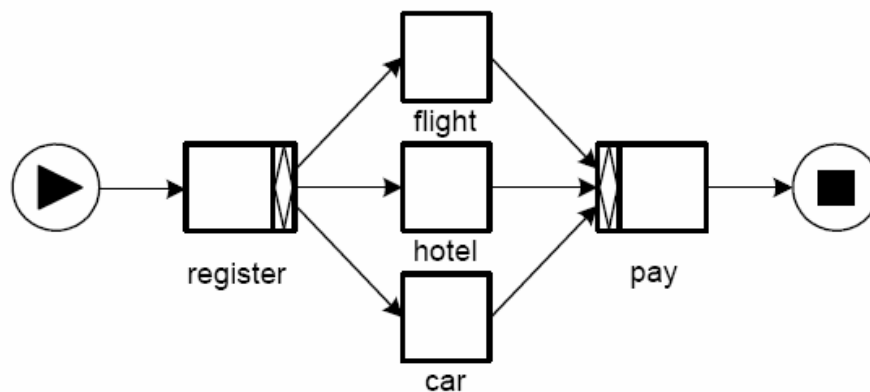


Abbildung 5: Beispiel einer Modellierung in YAWL [1]

Task register beinhaltet eine OR-Verzweigung (OR-split task) und task pay eine OR-Zusammenführung (OR-join task). In dem Beispiel muss sich ein Benutzer zunächst registrieren, bevor ein Flug, ein Hotel und/oder ein Auto gebucht werden kann. Es ist möglich, dass der Benutzer nur einen Flug, ein Hotel oder ein Auto bucht. Er kann aber auch Kombinationen buchen. Die OR-Zusammenführung (OR-join) synchronisiert die ausgewählten tasks in einem Bezahlvorgang.

## **1.2 Architektur des YAWL Systems**

Abbildung 6 zeigt die Architektur des YAWL Workflow-Management-Systems. Es ist modular aufgebaut und besteht aus den folgenden Komponenten:

### **YAWL-engine**

Im Zentrum der Architektur steht die YAWL-engine. Sie instanziiert Workflow Spezifikationen und kontrolliert ihre Ausführung. Es existieren definierte Schnittstellen zu den weiteren Komponenten des YAWL-Systems.

### **YAWL-repository**

Bestehende, lauffähige Spezifikationen werden durch das YAWL-repository verwaltet.

### **YAWL-designer**

Mit Hilfe des YAWL-designers können neue Spezifikationen erstellt werden. Es ist ein grafisches Tool bei dem die in 5.1 vorgestellten Modellierungskonstrukte zur Verfügung stehen. Neuerstellte Spezifikationen werden durch die YAWL-engine verifiziert und im YAWL-repository abgelegt.

### **YAWL-manager**

Die Administration von Workflow-Instanzen erfolgt mit Hilfe des YAWL-managers. Der YAWL-manager ist Teil der Administration and Monitoring Tools im Workflow-Referenz-Modell der Workflow-Management-Coalition(siehe oben).

### **YAWL-services**

Im Gegensatz zu anderen Workflow-Management-Systemen abstrahiert die YAWL-engine von Benutzern, Applikationen und Organisationen als Dienste. Sie kümmert sich nur darum was wann ausgeführt wird. Die Fragen wie und durch wen müssen durch entsprechende YAWL services beantwortet werden. Der YAWL-worklist-handler ist so ein Dienst. Er wird benützt um anstehende Aufgaben Benutzern zuzuweisen. Benutzer können Aufgaben annehmen und deren Fertigstellung anzeigen. Mit Hilfe des YAWL-webservice-brokers können web-services ausgeführt werden. Durch ihn ist es möglich Applikationen zu integrieren. Der YAWL-interopability-broker bildet die Verbindungsstelle zu weiteren workflow-engines. Durch ihn können workflows an andere engines delegiert werden. Neben den angesprochenen YAWL-services können durch den Benutzer weitere implementiert werden (custom-YAWL-services).

Im Referenz-Modell der WfMC repräsentieren der worklist-handler Workflow-Client-Applications und der YAWL-webservice-broker Invoked-Applications. Der YAWL-interopability-broker entspricht Workflow-Interoperability.

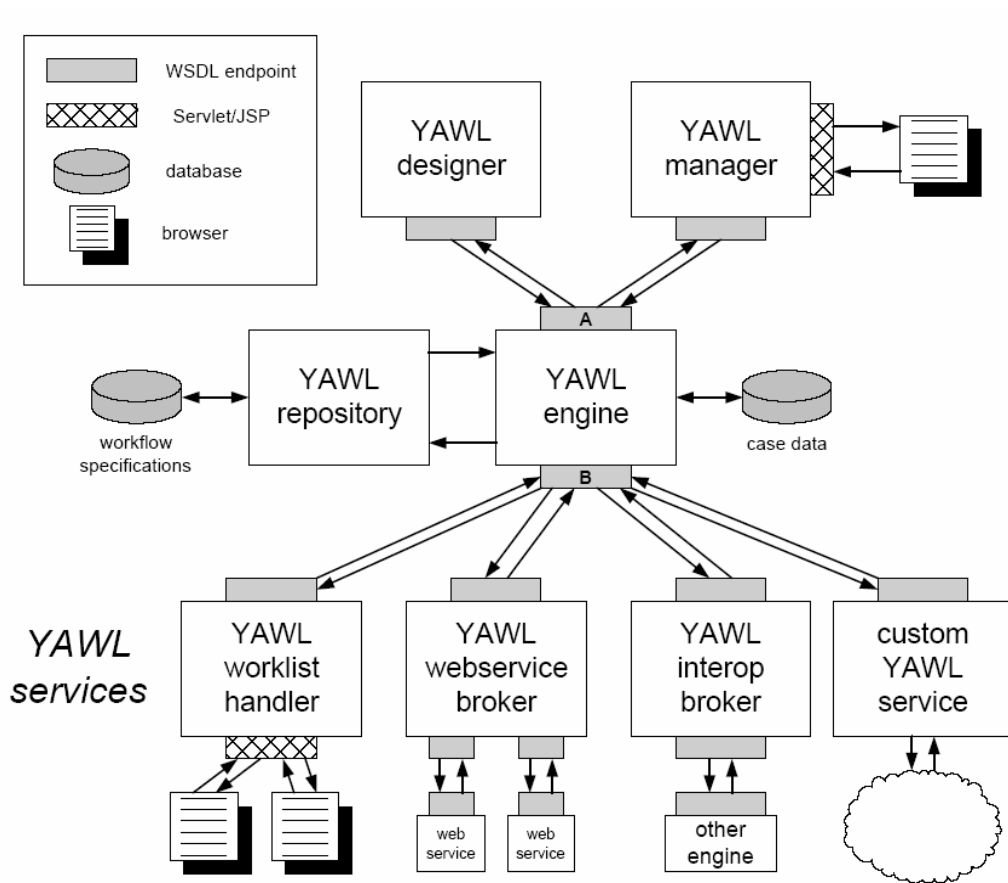


Abbildung 6: Architektur des YAWL Systems [1]

## Literaturangaben

- [1] W.M.P. van der Aalst, L. Aldred, M. Dumas, A.H.M. ter Hofstede; Design and implementation of the YAWL system; <http://www.yawl.fit.qut.edu.au/yawldocs/yawls.pdf>

### **1.3 Übungsaufgaben für das YAWL System (Reisebuchungs-szenario)**

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.

#### **1.3.1 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.

### **1.3.2 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.

### **1.3.3 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 Ankunftsort 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.

### **1.3.4 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.

### **1.3.5 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.