

TRAVEL.CH BPMN

Simon Lang

12. Juni 2016

Version 1.0.0

STUDIENGANG	Informatik 5 Ba 2012
SEMESTERARBEIT	Travel.ch BPMN
DOZENT	Daniel Liebhart
SCHULE	ZHAW - School of Engineering

Kurzfassung

In der Softwareentwicklung wird traditionell nach dem Wasserfall-Prinzip¹ vorgegangen. Viele Teams wird dieses alte Vorgehensmodel durch neue, agile Methoden abgelöst. In kurzen abständen (drei bis vier Wochen) wird dem Klient eine lauffähige Version vorgestellt. Kunden werden dabei stärker in den Entwicklungsprozess eingebunden, wodurch neue Probleme entstehen. In jedem Zyklus ist eine Spezifikations-, eine Umsetzungs- und eine Test-Phase enthalten. Bei umfangreicher Software kann der letzte Abschnitt sehr umfangreich werden und wird deshalb gerne vernachlässigt.

Um diesem Problem entgegenzutreten, kann das Testen der Software Automatisiert werden. Diesem Thema nimmt sich diese Arbeit an. Sie versucht am Beispiel der travel.ch aufzuzeigen, wie ein Automated Testing geplant und umgesetzt werden kann, und wann sich dieses lohnt und wann nicht.

Schlagwörter: Integration Tests, Selenium, Testing

¹ Wasserfallmodell.

Inhaltsverzeichnis

1	Einleitung	1
1.1	Aufgabenstellung	1
1.1.1	Ausgangslage	1
1.1.2	Ziele der Arbeit	1
1.1.3	Aufgabenstellung	1
1.1.4	Erwartete Resultate	1
1.2	Dieses Dokument	1
1.2.1	Recherche	1
1.2.2	Anforderungen und Analyse	1
1.2.3	Konzept	1
1.2.4	Umsetzung	1
1.2.5	Summary	1
2	Recherche	2
2.1	BPMN	2
2.1.1	Notation	2
2.2	Rahmenbedingungen	3
2.2.1	Travelwindow AG Prozesse	3
2.2.2	Travelwindow AG API	4
2.2.3	Betriebssystem	4
2.3	Programme	4
2.3.1	Anforderungen	4
2.3.2	Programmanalyse	5
3	Anforderungen und Analyse	6
3.1	Analyse der travelwindow AG API	6
3.2	Bonita	6
3.2.1	Interface	6
3.2.2	Business Data Model	7
3.2.3	Connectors	9
3.2.4	Forms, Contracts und Operations	12
4	Konzept	15

5	Umsetzung	16
6	Zusammenfassung & Fazit	17
A	Testübersicht	18
	Quellenverzeichnis	19

Abbildungsverzeichnis

2.1 Flow Objects	2
2.2 Events	2
2.3 Pool and Swimlanes	3
2.4 Example	3
2.5 Hotel plus Flight BPMN Model	4
3.1 Bonita Interface	7
3.2 Bonita BPM	8
3.3 Bonita Pool Variables	9
3.4 Bonita Connectors	10
3.5 Bonita Connector Implementation	11
3.6 Bonita Form Contract	12
3.7 Bonita Form Operations	13
3.8 Bonita Form erstellung aus Contract und Operations	14

Tabellenverzeichnis

2.1 Programme - Feature Gegenüberstellung	5
---	---

Akronyme

Bezeichnung	Beschreibung
BDM	Business Data Model

Glossar

API

API steht für Application Programming Interface. Dabei handelt es sich um eine Programmierschnittstelle, die den Austausch von Daten oder das starten von Prozessen erlaubt.

HAL

HAL steht für Hypertext Application Language. HAL ist eine Art wie APIs aufgebaut werden können. Das Ziel ist eine vereinfachung der API sowie diese frei erkundbar zu machen. Dies ermöglicht HAL über Links zwischen den einzelnen Ressourcen. Für weitere Informationen: http://stateless.co/hal_specification.html

JSON

JSON steht für JavaScript Object Notation. Dies ist eine definierte Schreibweise um Daten auszutauschen.

KAPITEL 1

Einleitung

1.1 Aufgabenstellung

In diesem Abschnitt wird die Aufgabenstellung gemäss Eingabe im Einschreibe und Bewertungssystem (EBS) der ZHAW aufgeführt.

1.1.1 Ausgangslage

Bislang besteht keine Abbildung der Travel.ch Prozesse.

1.1.2 Ziele der Arbeit

Der Prozess soll abgebildet und ausgeführt werden können.

1.1.3 Aufgabenstellung

Eine Abbildung des Prozesses soll erstellt werden. Initiale Eingaben müssen definiert werden und Antworten der Services müssen mit XSLT transformiert werden.

1.1.4 Erwartete Resultate

Ein Visualisierter Prozess welcher mit einer "Prozess Engine" ausgeführt werden kann.

1.2 Dieses Dokument

Dieses Dokument dokumentiert die Vorbereitung sowie die Umsetzung. Dieser Abschnitt soll eine Übersicht bieten über die Struktur der Arbeit.

1.2.1 Recherche

1.2.2 Anforderungen und Analyse

1.2.3 Konzept

1.2.4 Umsetzung

1.2.5 Summary

KAPITEL 2

Recherche

In diesem Kapitel wird erklärt was BPMN ist, es werden die Rahmenbedingungen umrissen und die verschiedenen Programme analysiert welche für die Umsetzung in Frage kommen.

2.1 BPMN

BPMN steht für Business Process Model and Notation. Dabei handelt es sich um eine genormte Visualisierung für Business Processes. Entwickelt wurde der Standart im Jahre 2001 durch die Firma IBM.

2.1.1 Notation

In diesem Abschnitt soll die grundlegenden Objekte von BPMN vorgestellt werden.

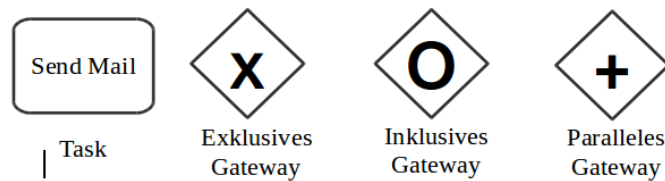


Abbildung 2.1: Flow Objects

Der Task ist ein Schritt eines Prozesses, welcher ausgeführt wird. Dieser kann entweder durch einen Menschen oder automatisiert durchgeführt werden. Die Rauten sind Gateways. Das Exklusive (XOR) Gateway, steht für einen Entscheidung. Es kann nur ein Pfad weitergeführt werden. Deshalb heisst es exklusiv. Das Inklusive (OR) Gateway ist wie das Exklusive Gateway eine Entscheidung, es können jedoch mehrere Wege parallel ausgeführt werden. Das Parallele (AND) Gateway führt alle Pfade parallel aus. Es kann auch dazu verwendet werden mehrere Pfade zu synchronisieren und als einzelner Weg weiterzuführen.



Abbildung 2.2: Events

Ein Event beschreibt, wenn etwas während eines Prozesses passiert. Oben aufgeführt sind die drei Grundevents. Der Start Event kennzeichnet der Beginn und der End Event das Ende eines Prozesses. Ein Intermediate Event kann irgendwo zwischen dem Start und dem Ende eines Prozesses stehen. Es gibt diverse Ausprägungen von Events, welche mit Icons in den Kreisen gekennzeichnet werden. Zum Beispiel ein Mail Event, Timer Event, Error Event, Cancel Event, Link Event, Signal Event und Terminate Event, um einige zu nennen.

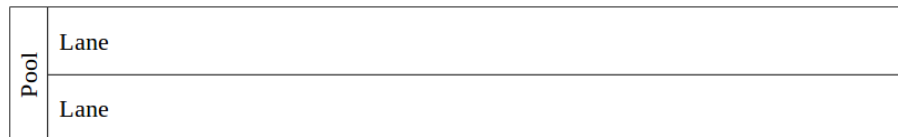


Abbildung 2.3: Pool and Swimlanes

Ein Pool kennzeichnet einen Participant, einen Benutzer oder Benutzerrolle in einem Prozess. Swimlanes ziehen sich über den gesamten Pool und werden dazu benutzt diesen weiter zu unterteilen.

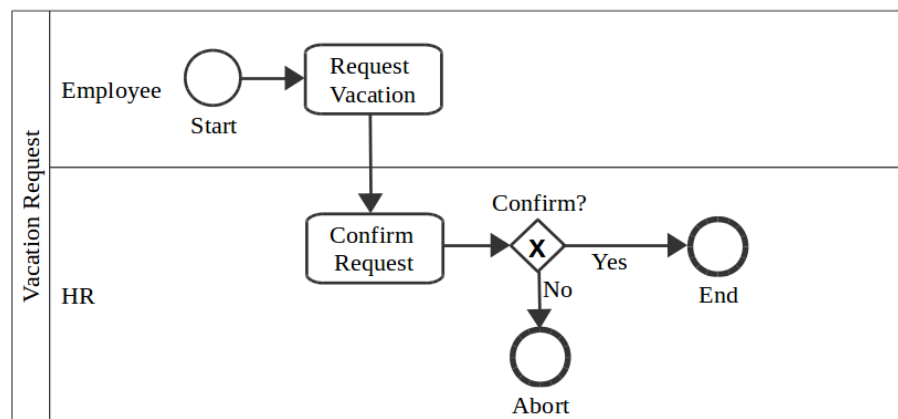


Abbildung 2.4: Example

Dies ist ein einfaches Beispiel eines Vacation Request Prozesses. Der Arbeitnehmer startet den Prozess und stellt seine Ferienanfrage. Die Anfrage wird von der Personalabteilung weiterverwendet und entscheidet, ob die Anfrage genehmigt wird oder nicht.

2.2 Rahmenbedingungen

Zu den Rahmenbedingungen gehört die Infrastruktur der travel.ch Webseite, das Entwicklungssystem des Programmierers sowie die etablierten Prozesse der travelwindow AG, welche die Webseite travel.ch betreibt.

2.2.1 Travelwindow AG Prozesse

Die Travelwindow AG ist der Betreiber der Seite travel.ch, auf welcher verschiedene Produkte gekauft werden können. Flüge, Hotel, Badeferien, Städtereisen sowie Hotel und Flug Kombinationen. Da der Buchungsprozess für Hotel und Flug Kombinationen der längste ist, wurde entschieden dass dieser in dieser Arbeit modelliert werden soll.

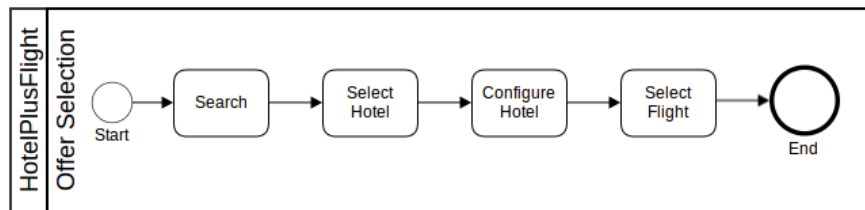


Abbildung 2.5: Hotel plus Flight BPMN Model

Der Prozess der Hotel und Flug Suche auf der travel.ch Webseite ist linear. Nach einer Produktsuche muss der Kunde ein Hotel wählen, welches er danach noch weiter Konfigurieren kann. Dabei kann er andere Zimmer- und Verpflegungstypen wählen. Zum Schluss erhält er eine Auswahl von Flügen bevor der Prozess mit einem End Event terminiert.

2.2.2 Travelwindow AG API

Travel.ch besteht aus einer Webseite und einer [API](#). Für die Ausführung der Prozesse via BPMN muss die API angesprochen werden. Diese liefert daten in [JSON](#) aus. Programme für das ausführen des travel.ch BPMN Prozesses muss demnach den Austausch von Daten über eine [API](#) mittels [JSON](#) ermöglichen.

2.2.3 Betriebssystem

Die travel.ch Webseite wird über eine [API](#) betrieben, welche mittels eines BPMN Programmes abgefragt werden soll. Diese [API](#) ist nur vom Firmennetzwerk aus erreichbar. In der Firma sind nur Rechner mit dem Windows Betriebssystem im Einsatz.

Das Entwicklungssystem des Programmierers ist eine Linux Mint Computer. Dabei handelt es sich um ein Unix basiertes Betriebssystem.

Es ist demnach zwingend Notwendig, dass das BPMN Programm auf Windows, sowie auf Unix basierten Betriebssystemen läuft.

2.3 Programme

In diesem Abschnitt werden verschiedene Programme vorgestellt und analysierten, ob sie für die Umsetzung dieses Projekte in Frage kommen. Dazu wird zuvor definiert, was die Anforderungen an das Programm sind.

2.3.1 Anforderungen

Eine zentrale Eigenschaft des Programmes ist, dass es eine Process Engine enthält. Dieses ermöglicht es, ein Prozess nicht nur in dem Tool zu zeichnen, sondern diesen auch auszuführen. Gemäss Aufgabenstellung muss das Tool auch die Datentransformation mittels XSLT unterstützen.

In den Abschnitten Abschnitt [2.2.1 Travelwindow AG Prozesse](#) und Abschnitt [2.2.2 Travelwindow AG API](#) wurde zudem beschrieben, dass das Programm auf den Betriebssystemen Windows und Unix lauffähig sein muss, sowie die Unterstützung von [JSON](#) basierten [APIs](#) beinhalten muss.

Zusätzlich sollte die Software gratis sein.

2.3.2 Programmanalyse

Die folgende Tabelle vergleicht verschiedene BPMN Tools auf die zuvor definierten Anforderungen (siehe Abschnitt 2.3.1 Anforderungen).

Tabelle 2.1: Programme - Feature Gegenüberstellung

Programm	Process Engine	Windows & Linux	API mittels JSON	XSLT	Gratis
Activiti ^a	x	x			x
Yaoqiang ^b		x			
jBPM ^c	x	x			x
Imixs ^d	x	x			x
IBM Business Process Manager ^e	x		x		
Camunda ^f	x	x			x
Drools ^g	x	x			x
Bonita ^h	x	x		x	x
BizTalk ⁱ	x		x	x	

^a *Activiti*. URL: <http://activiti.org/> (besucht am 12.06.2016).

^b *Yaoqiang BPMN Editor - an Open Source BPMN 2.0 Modeler*. URL: <http://bpmn.sourceforge.net/> (besucht am 12.06.2016).

^c *jBPM - Open Source Business Process Management - Process engine*. URL: <http://www.jbpm.org/> (besucht am 12.06.2016).

^d *Imixs Workflow Project - Open Source Java Workflow Engine for BPMN 2.0*. URL: <http://www.imixs.org/> (besucht am 12.06.2016).

^e *IBM - Software - IBM Business Process Manager*. URL: <http://www-03.ibm.com/software/products/en/business-process-manager-family> (besucht am 12.06.2016).

^f *BPMN Workflow Engine*. URL: <https://camunda.org/> (besucht am 12.06.2016).

^g *Drools - Drools - Business Rules Management System (Java™, Open Source)*. URL: <http://www.drools.org/> (besucht am 12.06.2016).

^h *Bonitasoft*. URL: <http://www.bonitasoft.com/> (besucht am 12.06.2016).

ⁱ *Microsoft BizTalk Integration | Microsoft*. URL: <https://www.microsoft.com/en-us/server-cloud/products/biztalk/> (besucht am 12.06.2016).

Von den Funktionen ist Microsoft's BizTalk die geeignetste Variante. Jedoch läuft dieses Programm nicht auf Linux und es ist nicht gratis. Die zweitbeste Wahl ist Bonita von Bonitasoft, welches keine API Anfragen mittels JSON unterstützt.

KAPITEL 3

Anforderungen und Analyse

3.1 Analyse der travelwindow AG API

Die API der travelwindow AG ist mit HAL aufgebaut. Sprich es gibt einen Einstiegspunkt, welcher unter „/“ erreichbar ist. Danach navigiert man sich über Links durch die Schnittstelle durch. Dies macht die API frei erkundbar, jedoch die Umsetzung mittels BPMN komplexer.

Nachfolgend werden die benötigten Anfragen an die API aufgeführt, welche für die Suche eines Hotel und Flug Angebotes benötigt werden (siehe Abschnitt 2.2.1 Travelwindow AG Prozesse). Die aufgeführten Antworten sind für eine bessere Übersichtlichkeit verkürzt.

```
1 {  
2   test: "haha",  
3   "blubb": 1  
4 }
```

3.2 Bonita

Also BPMN Programm wurde Bonita festgelegt (siehe Abschnitt 2.3.2 Programmanalyse). In diesem Abschnitt wird eine Einführung in das Tool gegeben.

Bonita besteht aus einem Programm, welches in Java geschrieben ist, sowie aus einer Webseite, welche standardmässig auf einem mitgelieferten Tomcat Webserver¹ läuft. Nachfolgend wird das Interface des Programmes und die verwendeten Funktionen der Software beschrieben.

3.2.1 Interface

Wenn man ein neues Projekt erstellt, sieht das Programm folgendermassen aus:

¹ Apache Tomcat® - Welcome! URL: <https://tomcat.apache.org/> (besucht am 12.06.2016).

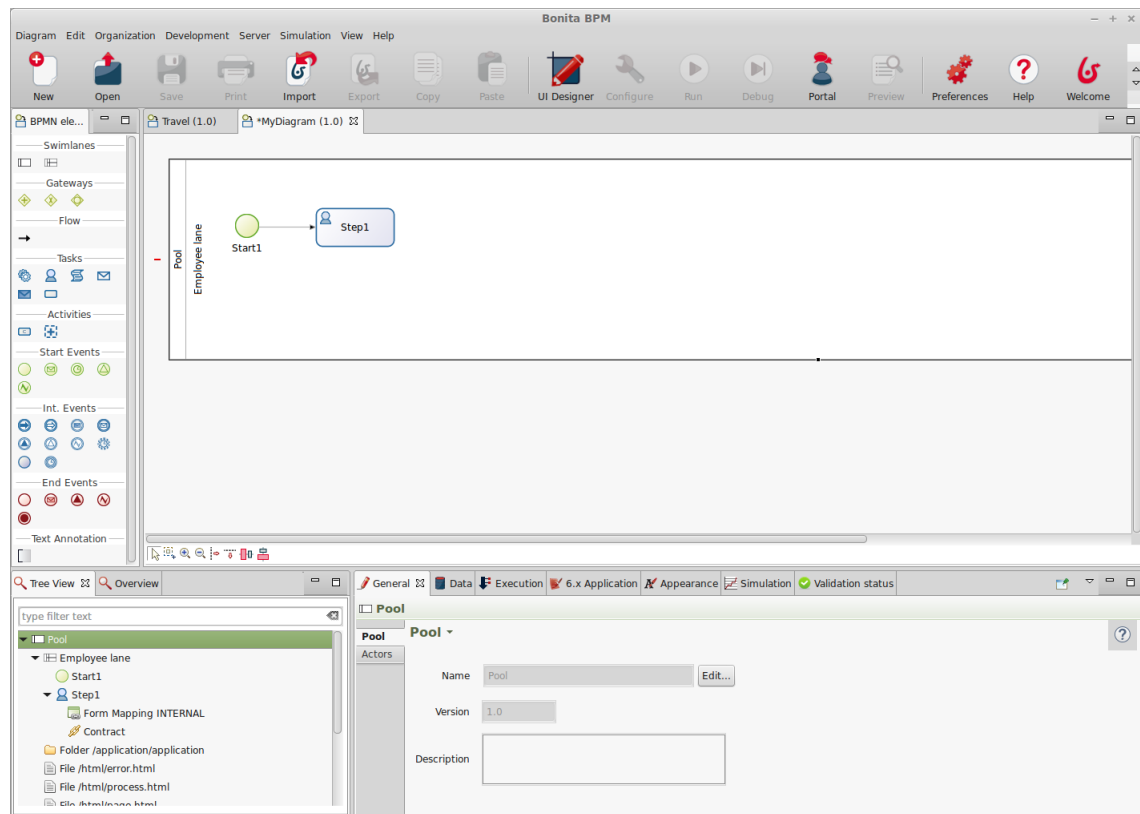


Abbildung 3.1: Bonita Interface

Zuerst ist die Toolbar. Darunter auf der linken Seite sind die BPMN Elemente (siehe Abschnitt 2.1.1 Notation) wie z.B. die Swimlanes, Gateways und Events. Daneben kann der Prozess definiert werden in der Process View. Bei einem neuen Projekt hat es einen Start Event (mit dem Namen „Start1“) und ein Task „Step1“. Bei dem Task hat es links oben ein Benutzer Icon. Dieses sagt aus, dass es sich um einen Human Task handelt, bei welchem eine Benutzerinteraktion benötigt wird.

Zuunterst links ist eine Übersicht über alle Elemente im Projekt (Tree View) und daneben können die BPMN Elemente im Projekt konfiguriert werden (Configuration View).

Bonita ist mit Java implementiert. Auch die Erweiterungen (siehe Abschnitt 3.2.3 Connectors) werden in dieser Sprache entwickelt. Für das Verständnis wird vorausgesetzt, dass man grundlegendes Wissen für Java Begriffe besitzt (z.B. Namespace, Class, Implementation, Interface, etc.).

3.2.2 Business Data Model

Business Data Model (BDM) werden für die Modellierung der Daten benötigt. Connectoren können später diese Daten entgegen nehmen oder zurückgeben. Intern handelt es sich dabei um Java Classes.

Ein **BDM** hat einen Namen und Felder. Felder haben wiederum einen Namen und einen Java Datentypen (string, int oder ein weiteres **BDM**). Ein Feld kann zusätzlich noch die zwei Attribute Multiple oder Mandatory haben. Ist Multiple gesetzt, so handelt es sich bei

dem Feld um eine Liste, und Mandatory macht es zwingend, dass bei der Erstellung ein Wert zugewiesen wird.

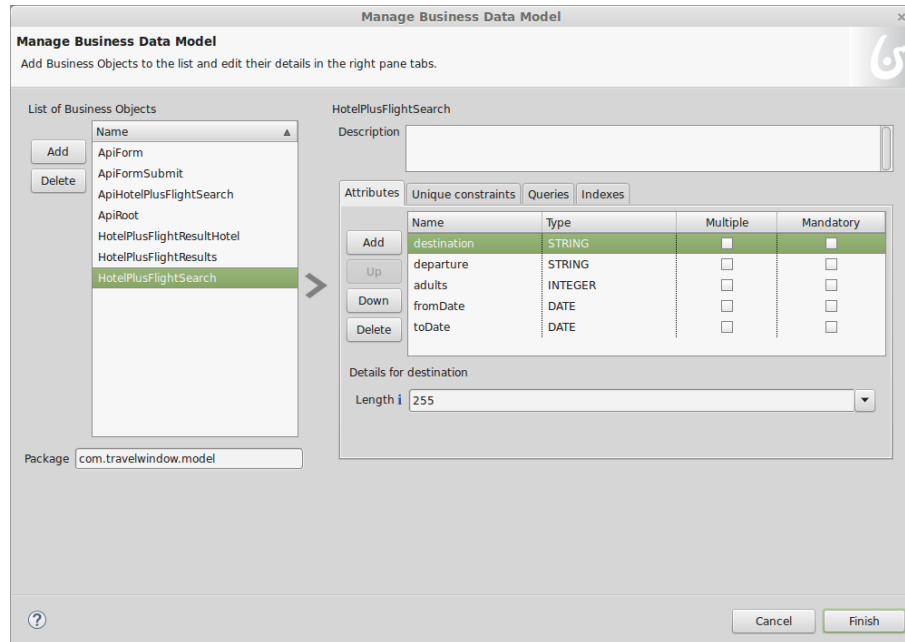


Abbildung 3.2: Bonita BPM

Pool Variables

Ist in der Process View ein Pool angewählt, so kann in der Configuration View unter „Data -> Pool Variables“ neue Pool Variablen definiert werden. Diese sind für alle Tasks in diesem Pool und Swimlanes sichtbar und können von den Connectors verwendet werden. Dabei handelt es sich um Instanzen von **BDMs**. Eine Pool Variable besteht aus einem Namen, einem **BDM** und einem Initialisierungs-Skript, um die Mandatory Fields abzufüllen.

Edit Business variable

Edit an existing Business variable

Name * hotelPlusFlightResultHotel

Description

Business Object * HotelPlusFlightResultHotel [Create a new Business Object...](#)

☐ Is multiple

Default value

Cancel OK

Abbildung 3.3: Bonita Pool Variables

3.2.3 Connectors

Connectors werden in Bonita dazu verwendet, um eine Aktion durchzuführen.

Bonita wird mit folgenden Connectors ausgeliefert:

- CMS
- CRM
- Calendar
- Database
- ERP
- LDAP
- Messaging
- Reporting
- SOAP Web Services
- Script
- Social
- Talend

Connectors können einem Task angehängt werden. Klickt man einen Task in der Process View an, kann man in der Configuration View unter Execution Connectors hinzufügen.

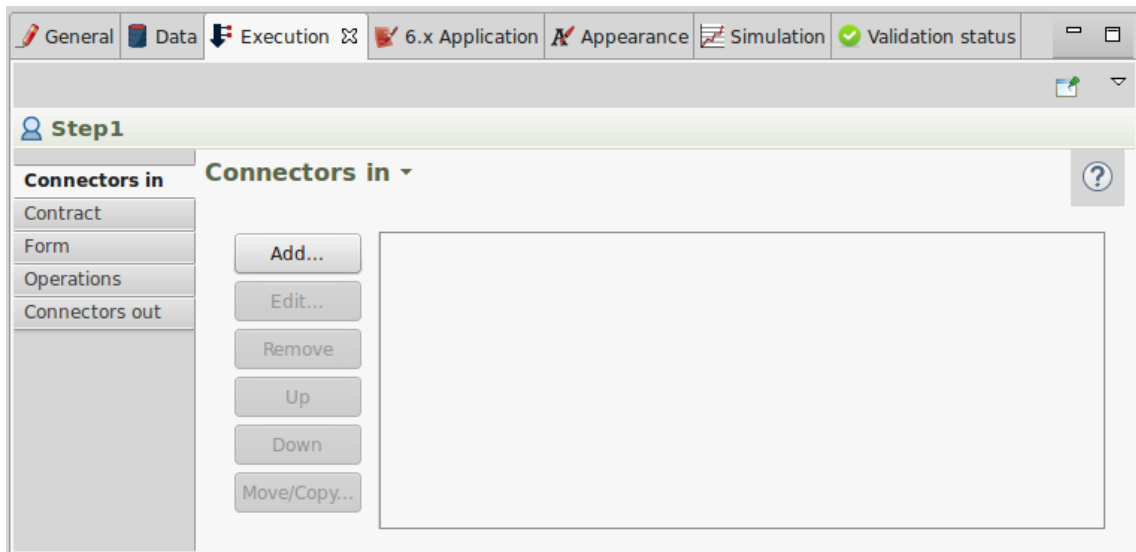


Abbildung 3.4: Bonita Connectors

Es gibt in und out Connectors. Diese werden entweder vor dem Task, oder nach dem Task ausgeführt. Bei einem Human Task kann so definiert werden, ob der Connector vor der Benutzerinteraktion ausgeführt wird oder danach.

Ein Connector besteht aus einer Definition und einer Implementation, welche nachfolgend beschrieben sind. In der Definition ist spezifiziert, was die Eingaben und Ausgaben sind. Wird ein Connector einem Task angehängt, so müssen die Eingaben definiert sowie ein Mapping der Ausgabeparameter gemacht werden.

Definition

Die Definition eines Connectors besteht aus Beschreibungsdaten (Name, Namespace, Kategorie, etc.), Input-Parameter, Wizard Pages und Output-Parameter. Als Input- und Output-Parameter können Java Classes oder [BDMs](#) definiert werden. Wizard Pages werden dazu benötigt, dass wenn ein Connector einem Task hinzugefügt wird die entsprechenden Input-Parameter definiert werden können.

Implementation

Um eine Implementation eines Connectors anzulegen, muss zuerst eine Definition gewählt und Beschreibungsdaten angegeben werden.

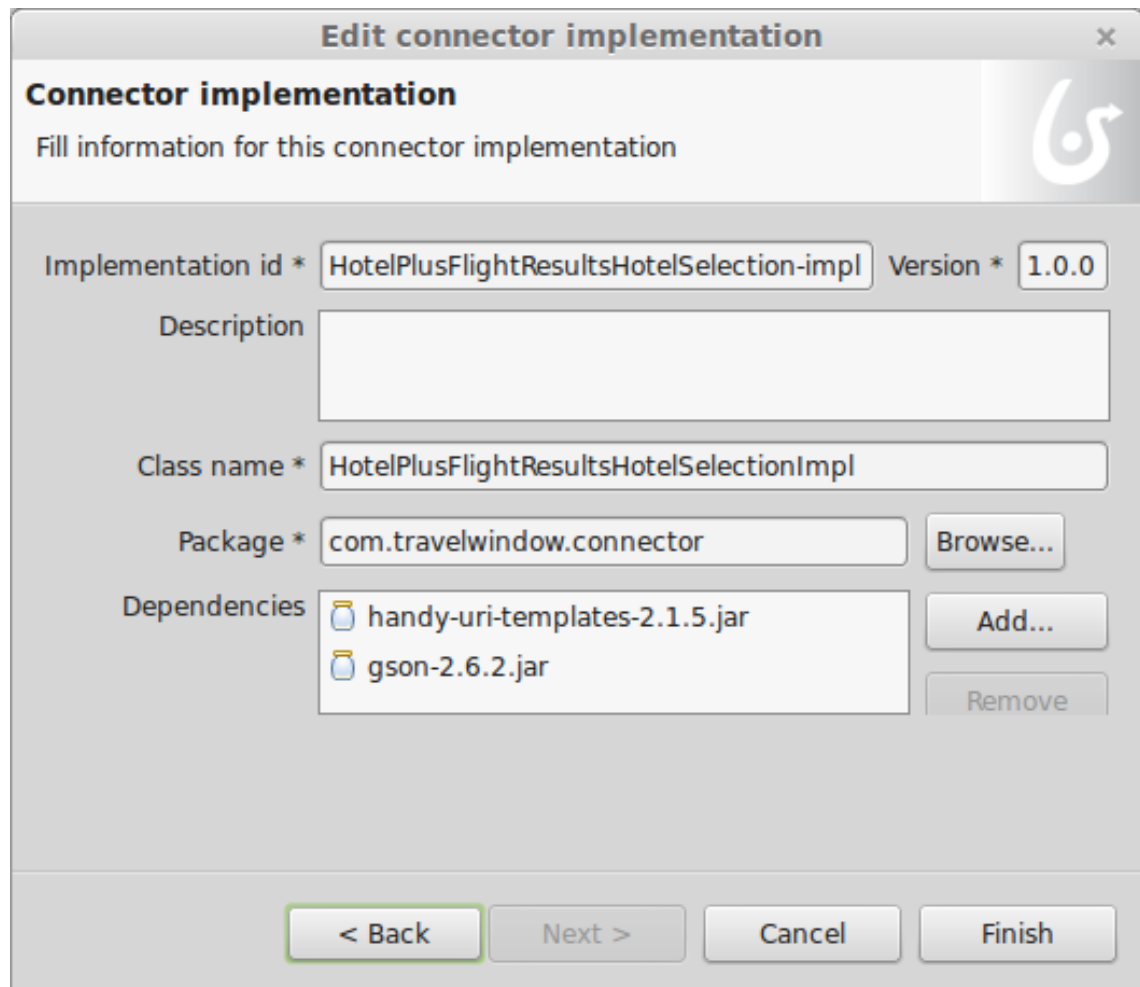


Abbildung 3.5: Bonita Connector Implementation

Die Implementation id kann frei gewählt werden, muss im Projekt jedoch eindeutig sein. Der Class name ist der Name der Java Klasse die erstellt wird. Dependencies sind JAR Files welche von innerhalb der Java Klasse benötigt werden.

Bestätigt man den obigen Dialog, so wird eine Java Klasse und abstrakte Klasse erzeugt. Nach dem obigen Beispiel werden die beiden Classes `HotelPlusFlightResultsHotelSelectionImpl` und `AbstractHotelPlusFlightResultsHotelSelectionImpl` erstellt. Die Hauptmethode der ersteren ist die Methode `executeBusinessLogic`. In der Definition dieses Connectors wurde definiert, dass ein [BDM](#) `HotelPlusFlightResultHotel` zurückgegeben wird, welches den namen „selection“ hat. Deshalb muss die Methode `executeBusinessLogic` die Methode „setSelection“ aufrufen und dieser eine Klasse vom Typen `HotelPlusFlightResultHotel` übergeben, um der Definition zu genügen. Die Methode `setSelection` ist dabei in der `AbstractHotelPlusFlightResultsHotelSelectionImpl` definiert.

Die Funktionalität eines Connectors kann somit frei implementiert werden. Zwingend ist nur das die Methode `setSelection` aufgerufen wird, sonst die die Ausführung des Connectors

fehlerhaft.

3.2.4 Forms, Contracts und Operations

Bei einem Human Task muss ein Form angehängt werden, um dem Benutzer eine Interaktion in dem Prozess zu ermöglichen. In einem Task kann direkt ein neues Formular erstellt werden. Es empfiehlt sich jedoch, zuerst den Contract zu definieren, da die dort angegebenen Informationen automatisch bei der Erstellung des Forms und den Operations mitberücksichtigt werden. Weitere Informationen dazu werden den folgenden beiden Abschnitten gegeben.

Contract

Ein Contract definiert die Daten, die vom Formular erwartet werden. Sie sind dafür verantwortlich, welche Informationen aus dem Process an das Form übergeben werden. Die dort definierten Werte können Java Klassen oder BDMs sein. Beim Task für die Auswahl eines Hotels in der Hotel und Flug suche, könnte dies die Liste der Hotelresultate sein. Dazu wird im Contract eine Liste von Hotels angegeben.

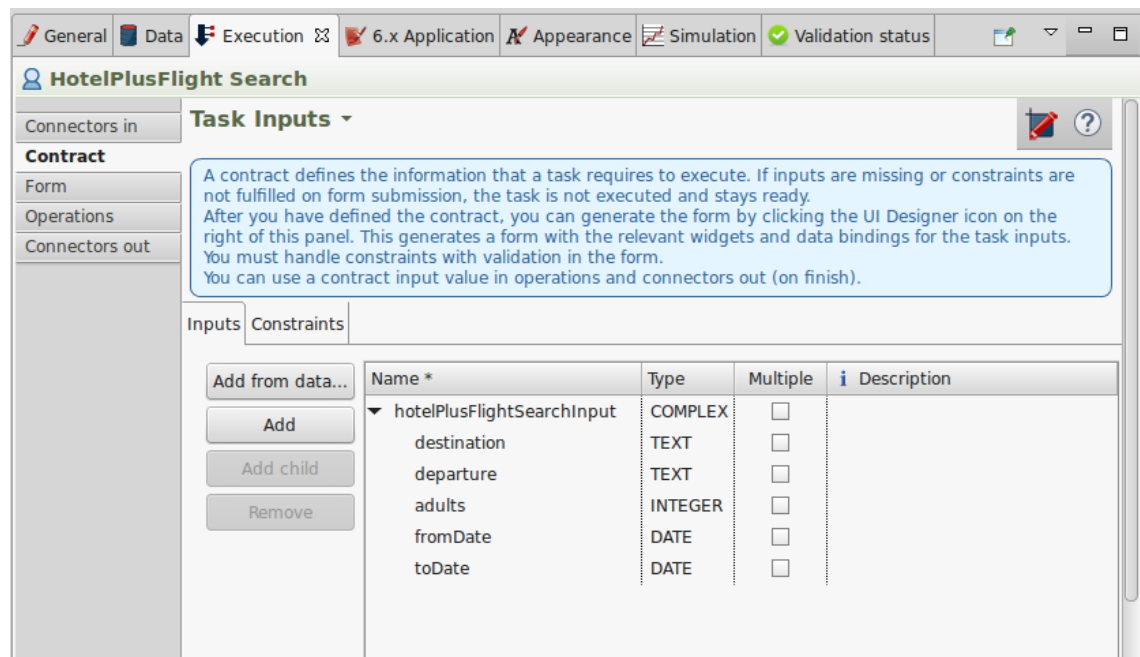


Abbildung 3.6: Bonita Form Contract

Operations

Operations werden nach der Ausführung des Formulars ausgeführt. Sie definieren, wie die Informationen aus dem Formular zurück in den Prozess fließen.

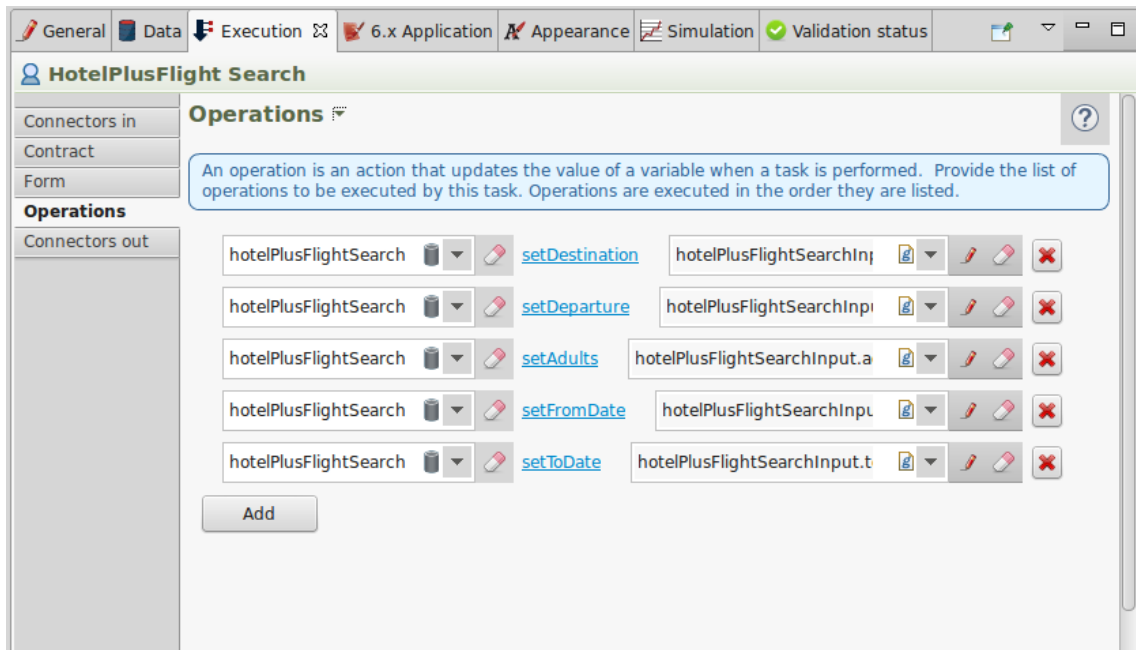


Abbildung 3.7: Bonita Form Operations

Im obigen Beispiel wird definiert, dass auf der Pool Variable `hotelPlusFlightSearch` die Java Methode `setDestination` aufgerufen wird. Der Methode wird der Wert `hotelPlusFlightSearchInput.destination` übergeben. Bei `hotelPlusFlightSearchInput` handelt es sich um eine Variable, welche auf dem Formular definiert wurde (siehe [Forms](#)).

Forms

Ein Form ermöglicht die Interaktion eines Benutzers. Die Daten die an das Form übergeben werden und wie die bearbeiteten Daten zurück in den Process fließen, ist in dem Contract und den Operations definiert (siehe [Contract](#) und [Operations](#)). Es ist zu empfehlen, dass zuerst diese beiden Informationen definiert werden, da sie bei der Erstellung des Forms direkt weiterverwendet werden. Definiert man bei dem Contract dass eine Liste von Hotels übergeben wird, so wird bei der Erstellung des Formulars automatisch eine Liste generiert welche diese Daten anzeigt.

Um ein Formular aus den Daten des Contracts und der Operations zu erstellen, muss dieses in der Configuration View erstellt werden.

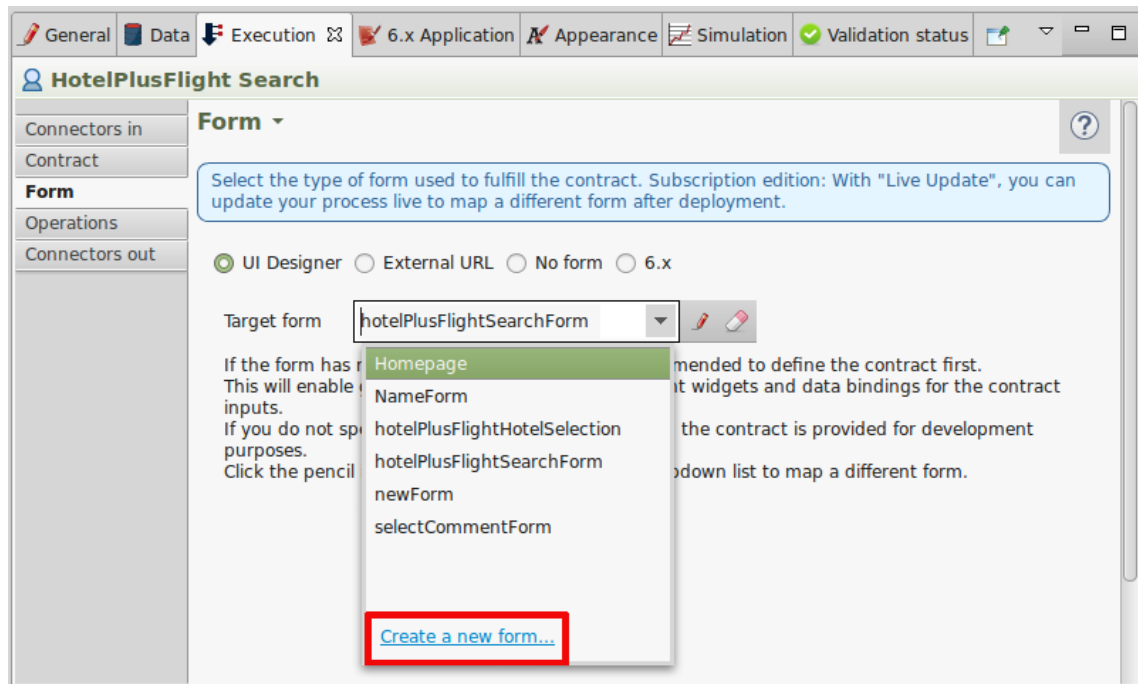


Abbildung 3.8: Bonita Form erstellung aus Contract und Operations

Danach öffnet sich der UI Designer. Dies ist ein Webinterface, welches auf dem mitgelieferten Tomcat Webserver läuft. Dort kann das Formular angepasst werden. Die Funktionalität des UI Designers ist gross. Einen guten Überblick bietet ein Video Tutorial, welches von Bonitasoft bereit gestellt wird und auf folgender URL eingesehen werden kann: <http://www.bonitasoft.com/resources/videos/getting-started-tutorial>

KAPITEL 4

Konzept

KAPITEL 5

Umsetzung

KAPITEL 6

Zusammenfassung & Fazit

ANHANG A

Testübersicht

Quellenverzeichnis

- [1] *Activiti*. URL: <http://activiti.org/> (besucht am 12.06.2016) (siehe S. 5).
- [2] *Apache Tomcat® - Welcome!* URL: <https://tomcat.apache.org/> (besucht am 12.06.2016) (siehe S. 6).
- [3] *Bonitasoft*. URL: <http://www.bonitasoft.com/> (besucht am 12.06.2016) (siehe S. 5).
- [4] *BPMN Workflow Engine*. URL: <https://camunda.org/> (besucht am 12.06.2016) (siehe S. 5).
- [5] *Drools - Drools - Business Rules Management System (Java™, Open Source)*. URL: <http://www.drools.org/> (besucht am 12.06.2016) (siehe S. 5).
- [6] *IBM - Software - IBM Business Process Manager*. URL: <http://www-03.ibm.com/software/products/en/business-process-manager-family> (besucht am 12.06.2016) (siehe S. 5).
- [7] *Imixs Workflow Project - Open Source Java Workflow Engine for BPMN 2.0*. URL: <http://www.imixs.org/> (besucht am 12.06.2016) (siehe S. 5).
- [8] *jBPM - Open Source Business Process Management - Process engine*. URL: <http://www.jbpm.org/> (besucht am 12.06.2016) (siehe S. 5).
- [9] *Microsoft BizTalk Integration | Microsoft*. URL: <https://www.microsoft.com/en-us/server-cloud/products/biztalk/> (besucht am 12.06.2016) (siehe S. 5).
- [10] *Yaoqiang BPMN Editor - an Open Source BPMN 2.0 Modeler*. URL: <http://bpmn.sourceforge.net/> (besucht am 12.06.2016) (siehe S. 5).