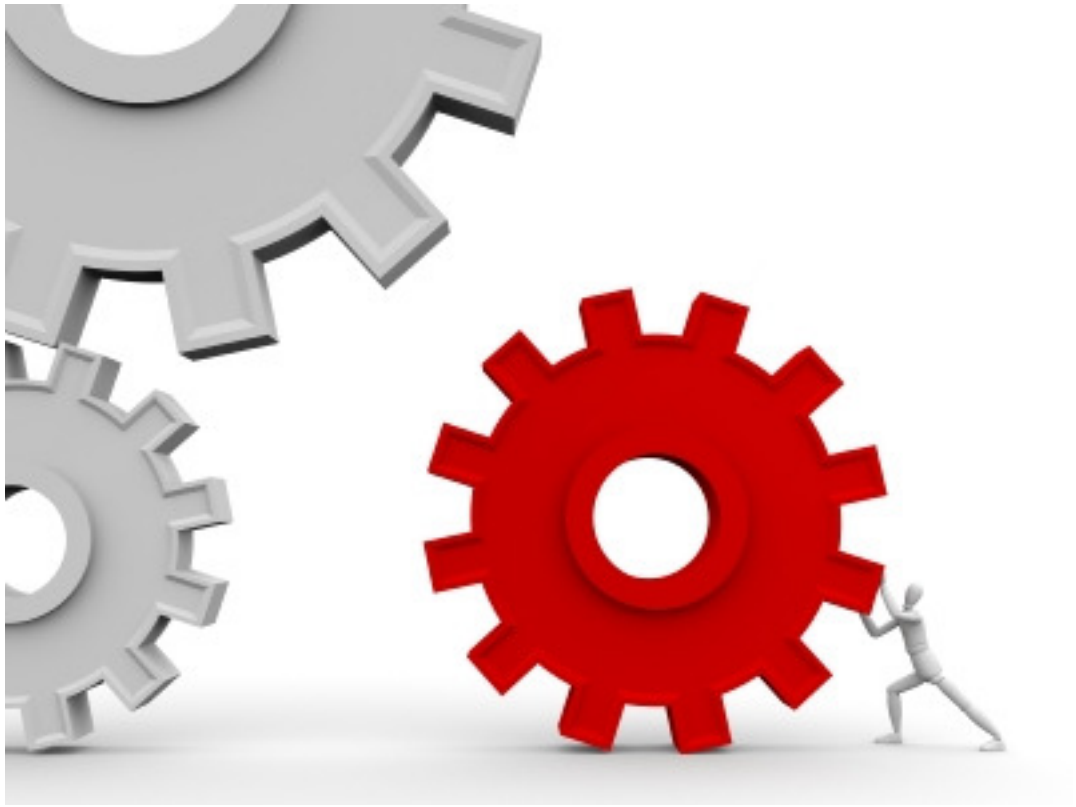


camunda

The Business Process Company

Geschäftsprozesse und Regeln mit jBPM und Drools

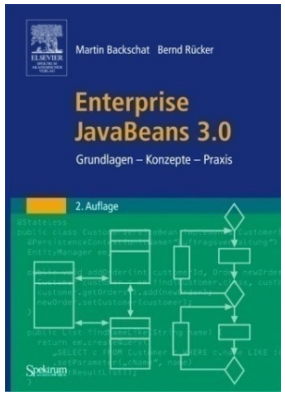
ein unschlagbares Team

Bernd Rücker

camunda
The Business Process Company

Wer bin ich?

- Berater, Trainer, Coach
- Softwareentwickler
- Committer im JBoss jBPM-Projekt
- Themen: BPM, SOA, Process Execution (jBPM, BPEL, XPDL, …), Java EE
- Eigene Trainings zu Process Execution, BPMN, BPM-Software, …



Agenda

camunda
The Business Process Company

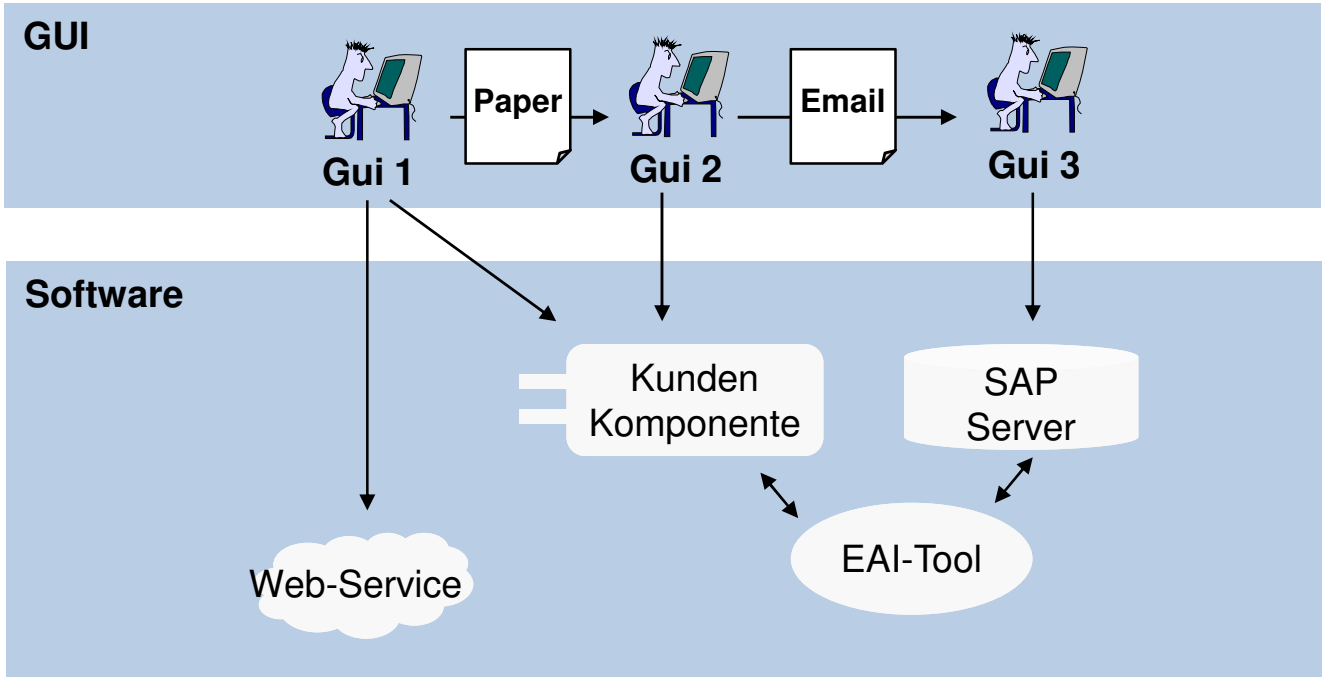
Geschäftsprozess und Regeln

- Business Process Management (BPM)
- JBoss jBPM
- Business Rules Management (BRM)
- JBoss Drools
- Einsatzgebiete, Abgrenzung und Kombinationsmöglichkeiten

Geschäftsprozesse & Software

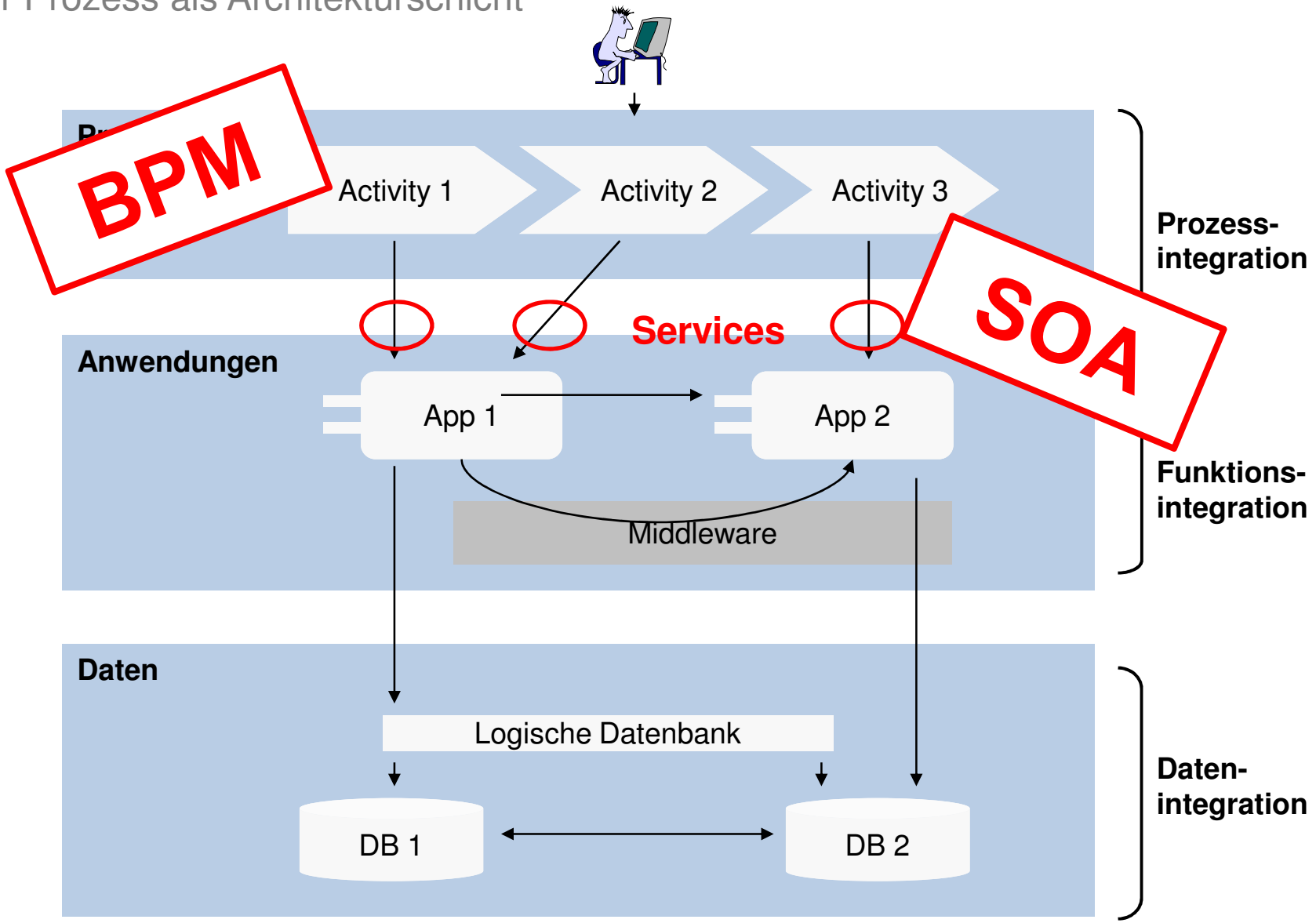
camunda
The Business Process Company

Der Status Quo in manchen Unternehmen



Softwareintegration

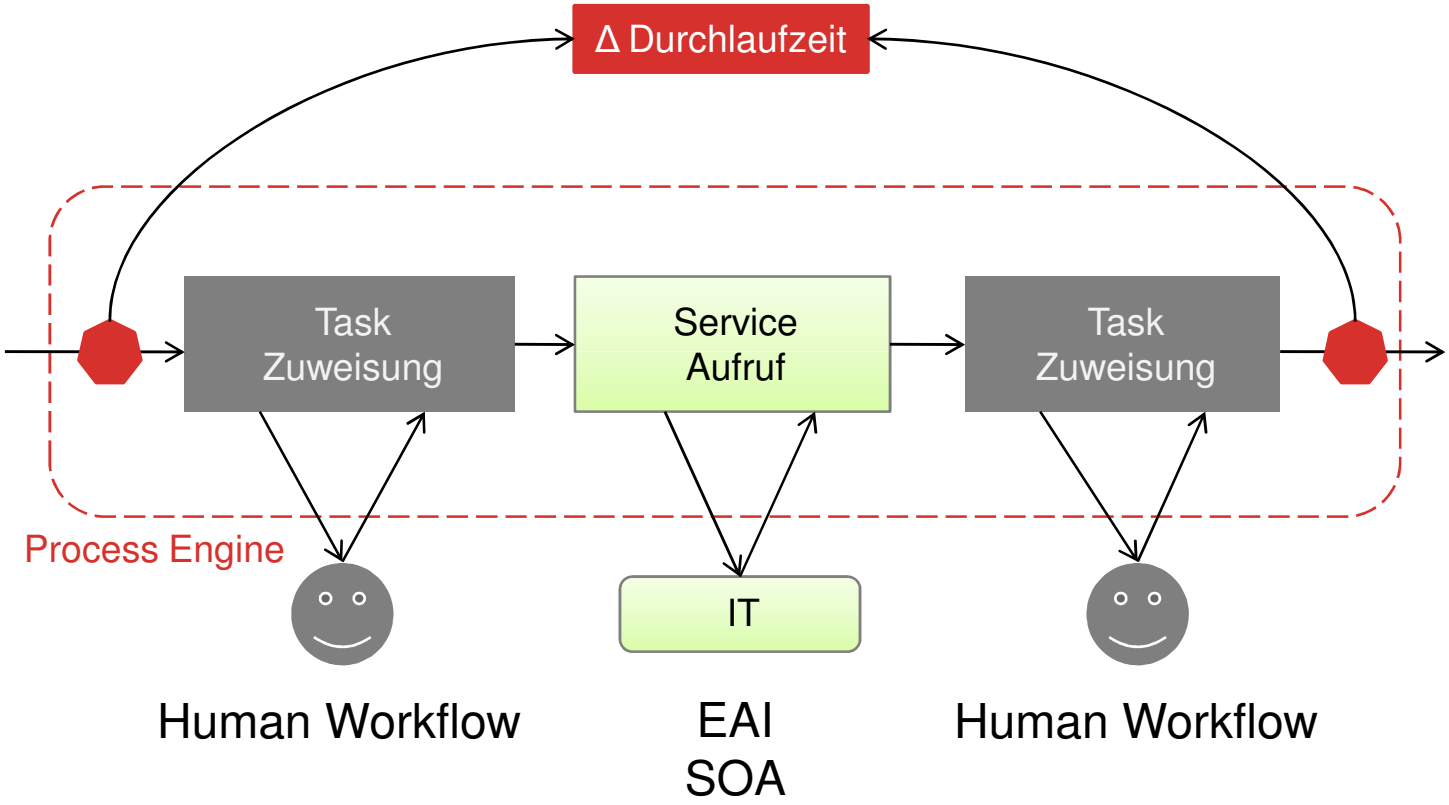
Der Prozess als Architekturschicht



Ein „digitaler“ Prozess

mit Business Process Engine

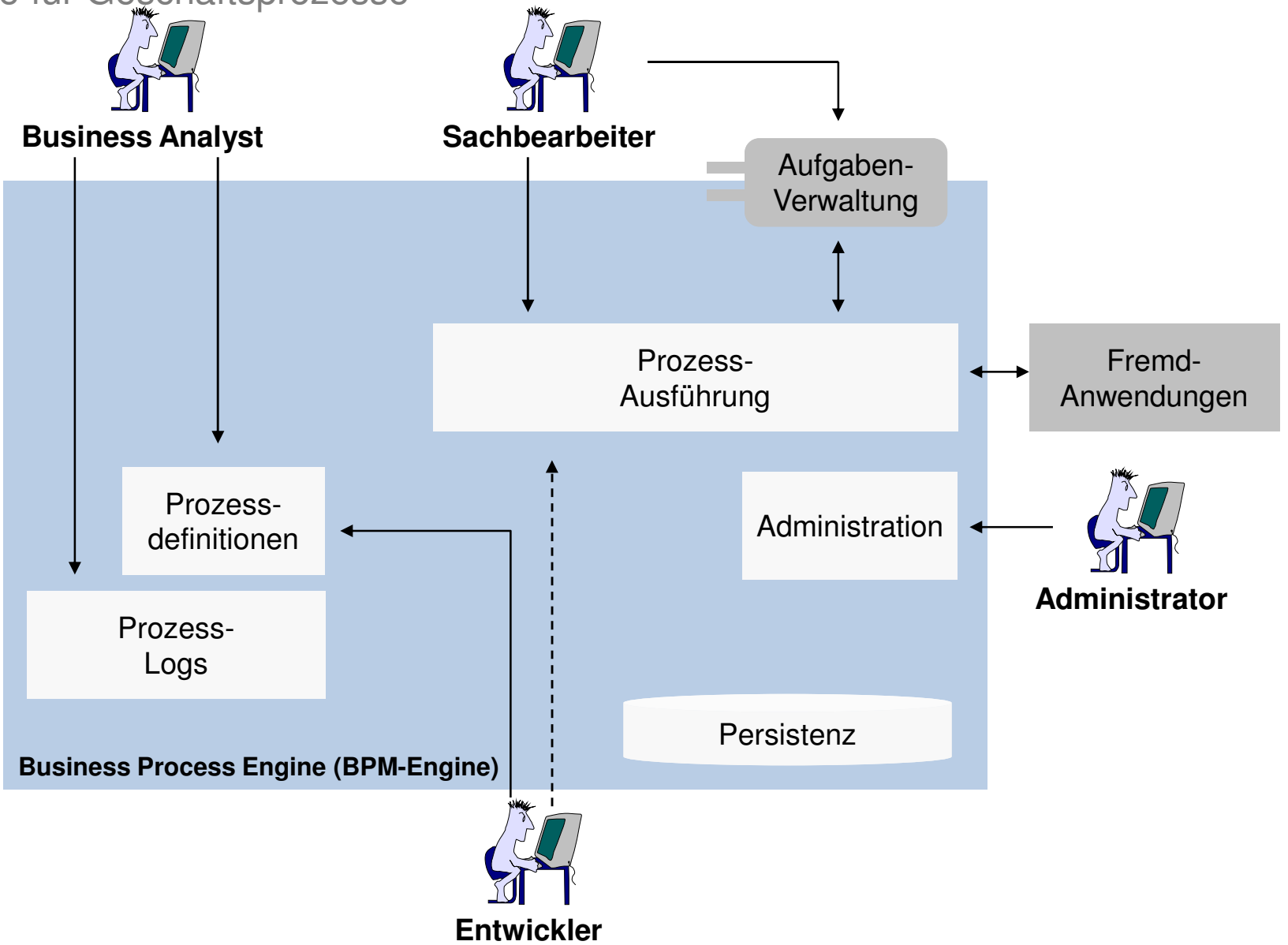
camunda
The Business Process Company



Business Process Engine

camunda
The Business Process Company

Middleware für Geschäftsprozesse



Was leistet die Business Process Engine

camunda
The Business Process Company

Features

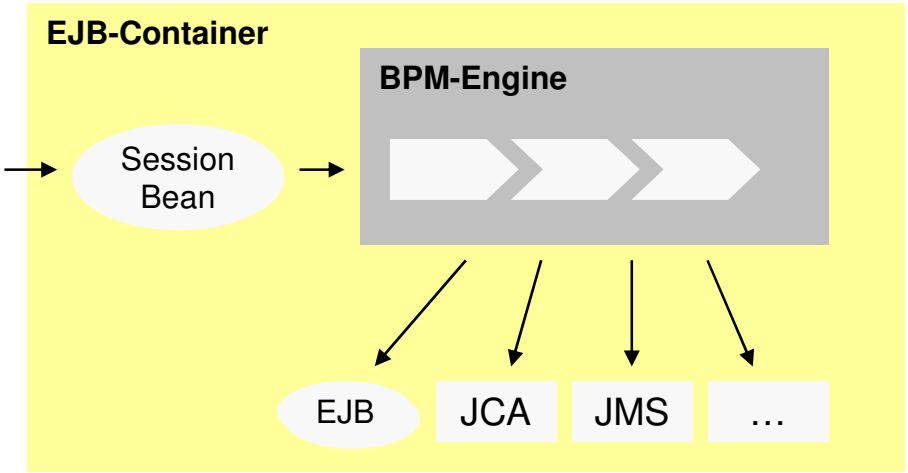
- Versionierung, Persistenz & Interpretation von Prozessmodellen
- Steuerung & Persistenz von Prozessinstanzen
- Task-Management & Wait-States
- Prozesskontext (Variablen zu Prozess speichern)
- Einbindung externer Services
- Verwalten von Ereignissen (wie Timeouts, …)

Business Process Engine in Java

camunda
The Business Process Company

Architektur

- Process Engine ist eigene Architekturschicht
- Domänenobjekte oder Referenzen als Prozessvariablen
- Ansteuerung ext. Services



JBoss jBPM

camunda
The Business Process Company

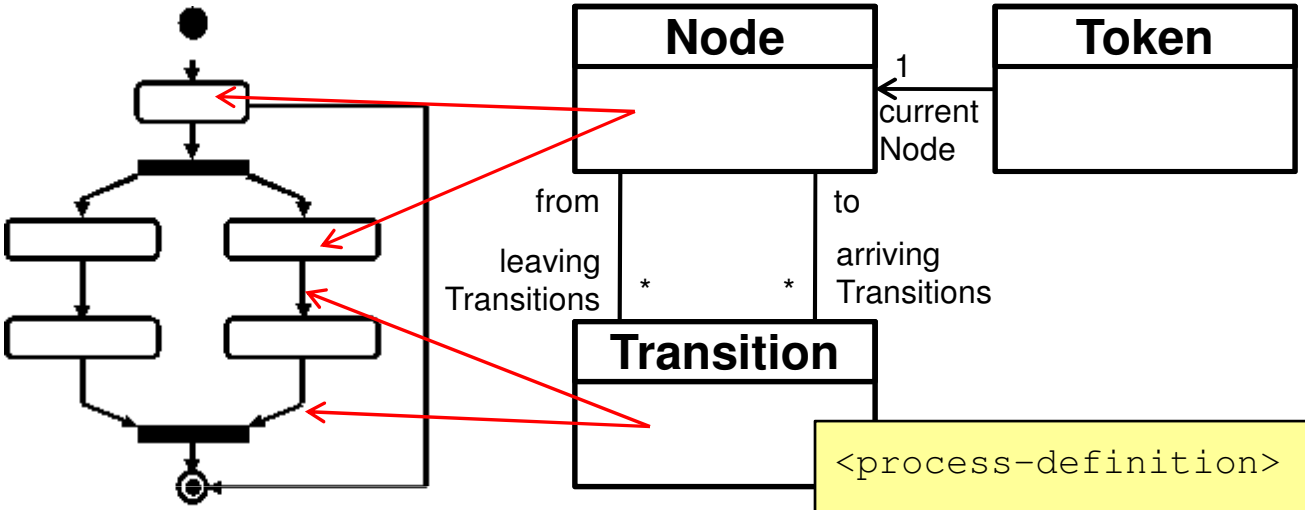
Open Source Process Execution

- Business Process Engine
- POJO-Kern: Interne Prozessrepräsentation durch Java-Modelle
- Persistenz über Hibernate (DB-Unabhängigkeit)
- Lauffähig mit oder ohne Application-Server
- „Library“
- Klein und flexibel, leicht erweiterbar
- Aktuell Version 3.2, Version 4 in der Entwicklung
- Open Source (LGPL)

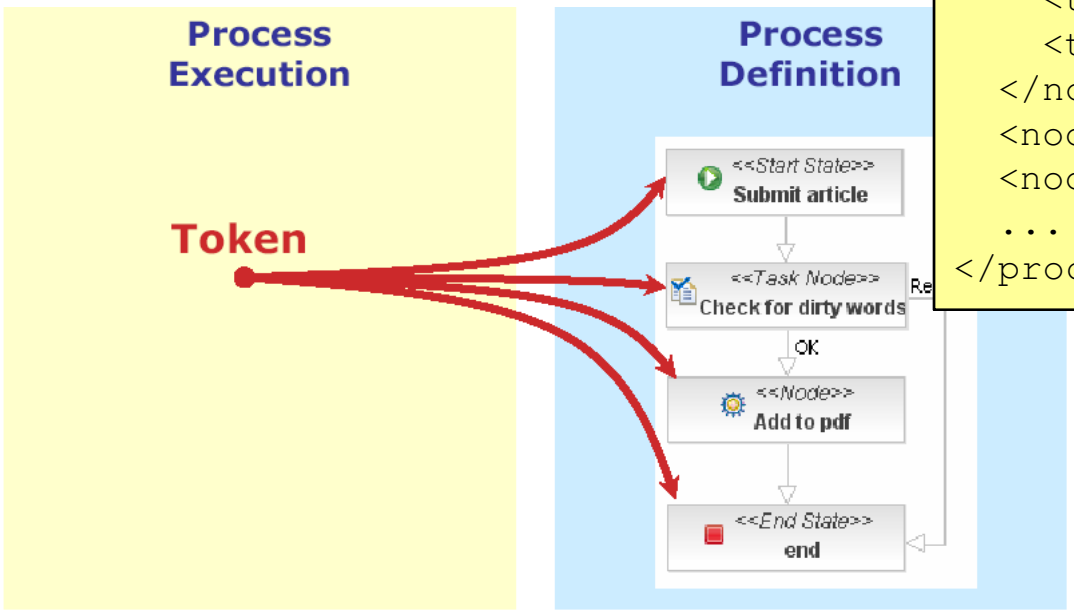
„Graph oriented programming“

camunda
The Business Process Company

jBPM in a nutshell



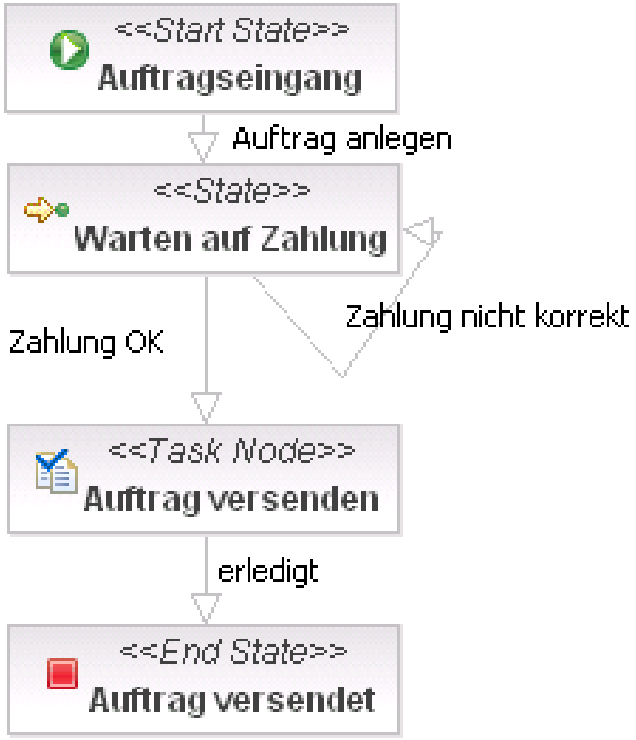

```
<process-definition>
  ...
  <node name="serve client">
    <transition name="ok" to="order" />
    <transition name="nok" to="joke" />
  </node>
  <node name="order" />
  <node name="joke" />
  ...
</process-definition>
```



Verschiedene Node-Typen

jBPM in a nutshell

- Task-Node: Human Tasks / Aufgaben
- State: Wait-States
- Fork / Join
- Decision: Automatische Entscheidung
- Start-State / End-State
- …
- Eigene Node-Typen mit Verhalten können implementiert werden



jBPM & Java

camunda
The Business Process Company

jBPM in a nutshell

- Einfache Java-API zur Steuerung der Engine
  - Prozessstart
  - Aufgabenliste
  - …
- Aufrufen von „User-Code“
  - definierte Stellen im Prozess
  - Interface & Java-Klassen

jBPM & Java

camunda
The Business Process Company

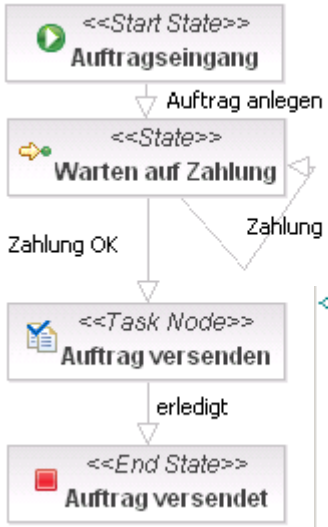
jBPM in a nutshell

```
JbpmConfiguration conf = JbpmConfiguration.getInstance();
JbpmContext context = conf. createJbpmContext();

ProcessInstance pi = context.getGraphSession().
          findLatestProcessDefinition("Ticket").createProcessInstance();
pi.getRootToken().signal();

List<TaskInstance> tasks = context.getTaskMgmtSession().
          findTaskInstances("Vertrieb");
tasks.get(0).end("Ticket schliessen");

context.close();
```

```
public class MyAction implements ActionHandler {
 public void execute(ExecutionContext ctx) {
  Object var = ctx.getVariable("var");
  result = service.doSomething(var);
  ctx.setVariable("result", result);
 }
}
```

Gemeinsame Sprache, Beispiel jBPM

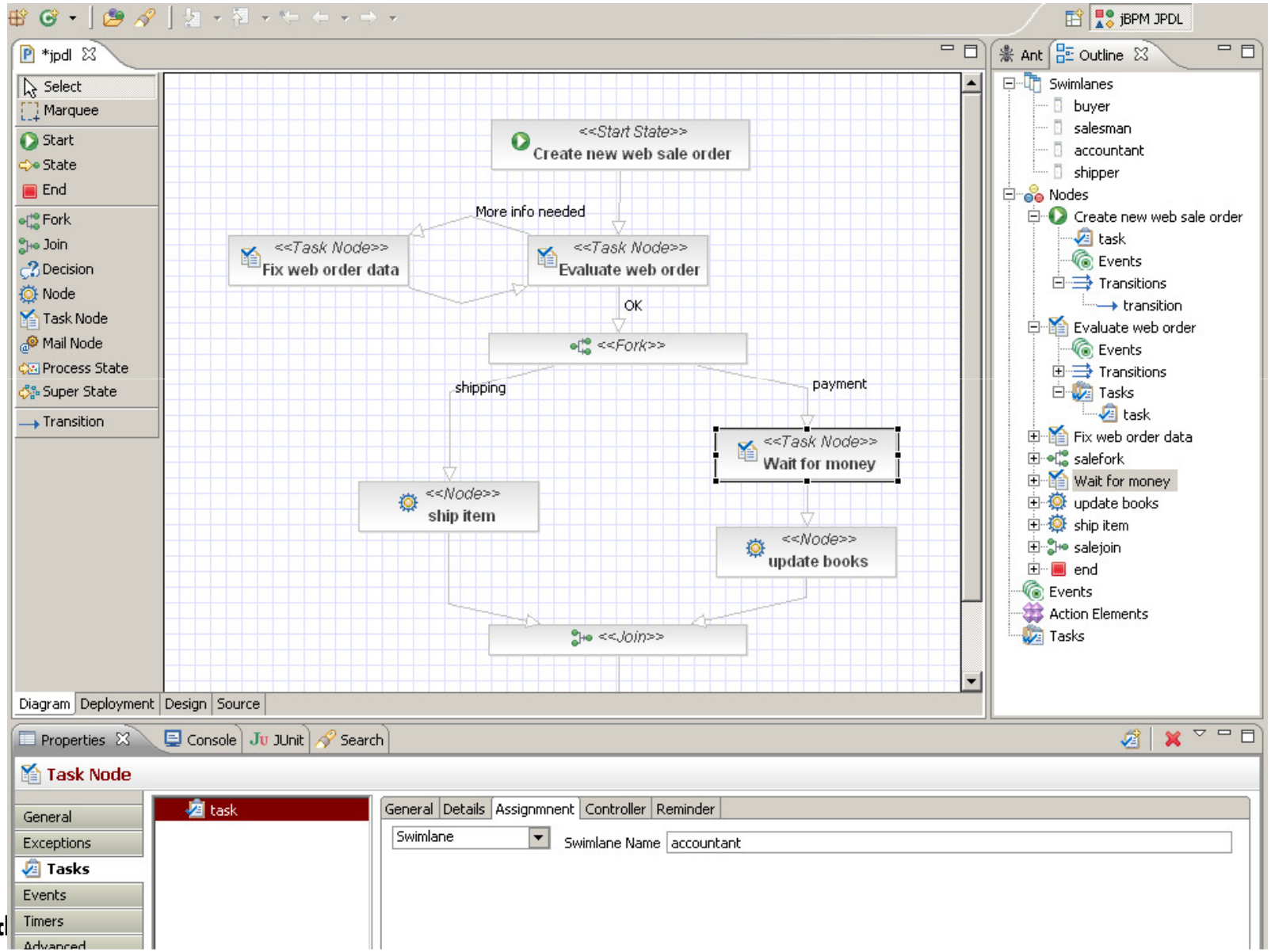
camunda
The Business Process Company




```
<process-definition name="SimpleOrder">
   <swimlane name="Lager">
      <assignment pooled-actors="Lager" />
   </swimlane>
   <start-state name="Auftragseingang">
      <event type="node-leave">
         <action name="create order in business logic" class="com.camunda.jmgui.actions
      </event>
      <transition name="Auftrag anlegen" to="Warten auf Zahlung"></transition>
   </start-state>
   <state name="Warten auf Zahlung">
      <transition name="Zahlung OK" to="Auftrag versenden"></transition>
      <transition name="Zahlung nicht korrekt" to="Warten auf Zahlung"></transition>
   </state>
   <task-node name="Auftrag versenden">
      <task name="Auftrag versenden" swimlane="Lager" />
      <event type="node-leave">
         <action name="set order shipped status in business logic" class="com.camunda.
      </event>
      <transition name="erledigt" to="Auftrag versendet"></transition>
   </task-node>
   <end-state name="Auftrag versendet" />
</process-definition>
```

Tooling: Eclipse

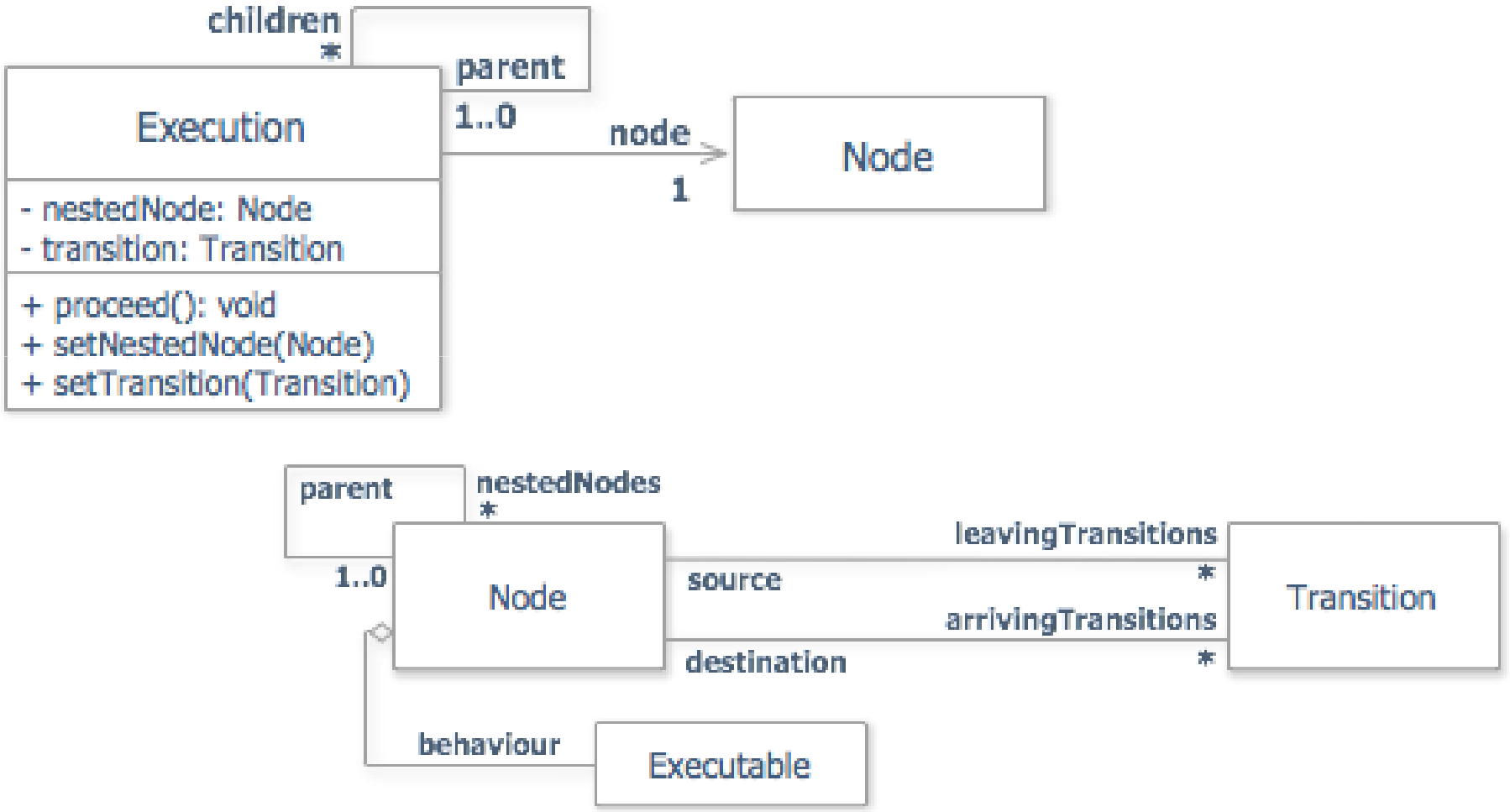
jBPM in a nutshell



Die Zukunft: Process Virtual Machine (PVM)

camunda
The Business Process Company

Sprachen werden aufbauend auf der PVM entwickelt



Was sind Regeln?

camunda
The Business Process Company

- „Wenn ich müde bin, dann gehe ich ins Bett!“

- „WENN .. DANN ..“-Struktur
- Bedingung und Konseqenz (Prämisse und Konklusion; Left-Hand-Side LHS und Right-Hand-Side RHS)
- Konsequenz wird häufig als „Aktion“ bezeichnet
- Bedingungen prüfen „Fakten“
- Regeln „feuern“, wenn deren Bedingung eintrifft

Wie werden Regeln umgesetzt?

camunda
The Business Process Company

Alternativen

- Direkte Programmierung im Quellcode:

```
if ( person.istMuede() == true ) {
  person.putzeZaehne();
  person.geheInsBett();
}
```

- Spezifische Lösungen (Codegenerierung, DSL, Speziallösungen, …)
- Regelmaschine / Rule Engine

Probleme programmierter Regeln

camunda
The Business Process Company

Wie werden Regeln umgesetzt?

- Wartbarkeit und Validierbarkeit nicht gegeben
- Regeln müssen durch Entwickler in Quellcode übersetzt werden
- Fachliche Regeln werden über verschiedene Klassen verteilt
- Keine Lesbarkeit der Regeln für den Fachbereich
- Konflikt-Lösung muss realisiert werden

Vorteile der Rule-Engine

camunda
The Business Process Company

Wie werden Regeln umgesetzt?

- Explizite Formulierung der Regeln als Regeln
- Deklarativ: Welche Regeln wann wie ausgeführt werden entscheidet die Regelmaschine
- Regeln für Fachbereich verständlich

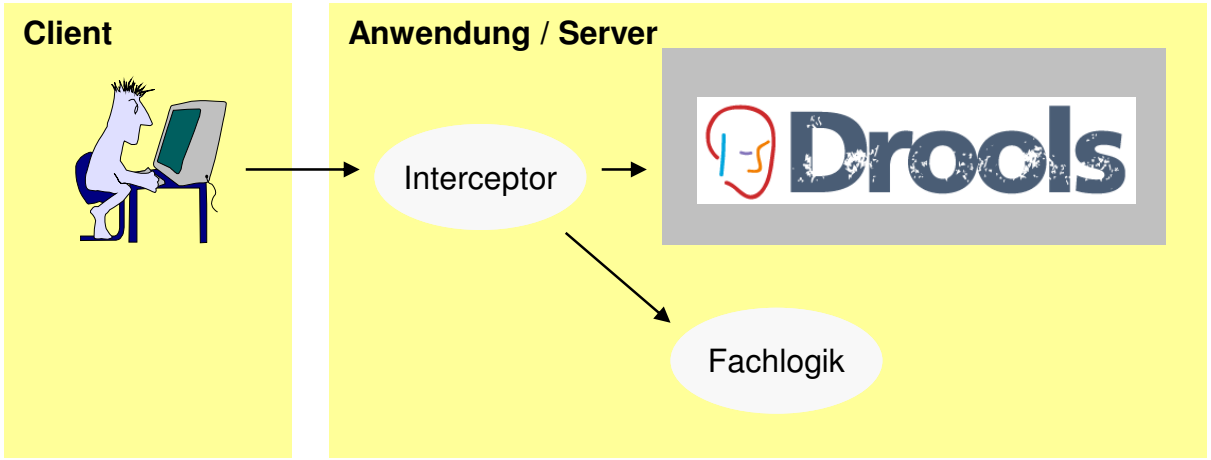
```
Bedingung:
  Person.muede = true

Konsequenz:
 person.putzeZaehne();
 person.geheInsBett();
```

Rule Engines in Java

camunda
The Business Process Company

Architektur

- Fakten (Wissen) = Domänenobjekte (POJOs)
- Rule Engine wird generisch in die Architektur integriert (Interceptoren, …)



JBoss Drools

camunda
The Business Process Company

Die Open Source Rule Engine

- Java Rule Engine (RETE-Implementierung)
- „JBoss Drools“ / „JBoss Rules“
- Lauffähig mit oder ohne Application-Server
- „Library“
- Business Rules Management System (BRMS) in der Entwicklung
- Aktuell Version 4.0
- Open Source (ASL)

Drools-Regeln

Beispiel

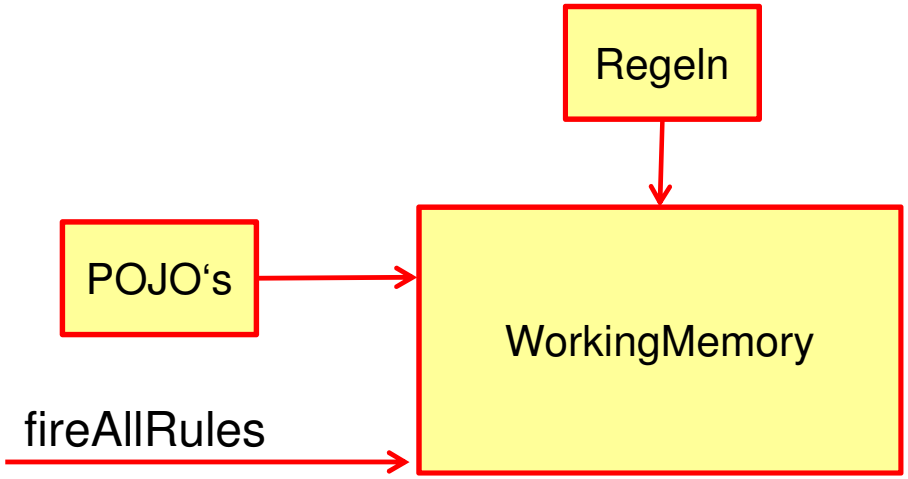
camunda
The Business Process Company

```
package com.camunda.demo

import demo.business.*;
import demo.infrastructure.ErrorList;

global ErrorList errors;

rule "Auftragsrabatt bei hohem Bestellwert"
when
   o: Order( value>5000 )
then
   o.setDiscount(0.05);
end

rule "Nachnahme nur bis 2500 € möglich"
when
   o: Order( value>2500, shippingType="COD" )
then
   errors.addError("Nachname nicht möglich bei Auftragswert " + o.getValue()
);
end
```

Drools im Einsatz

camunda
The Business Process Company

API



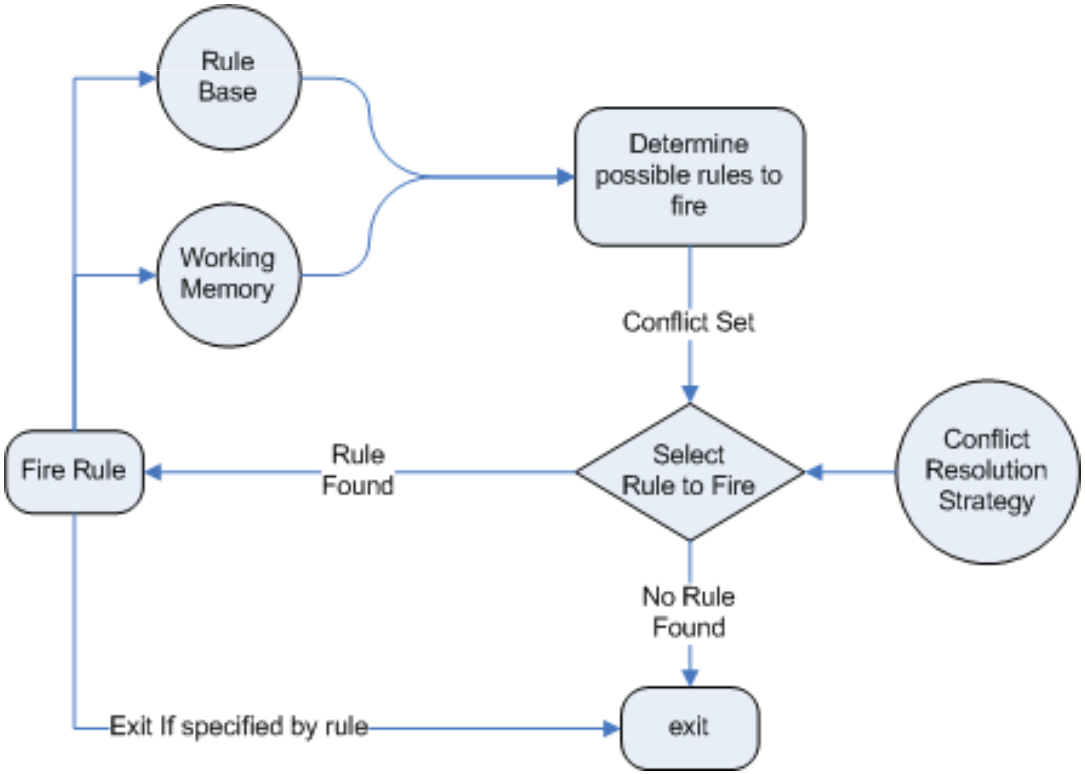

```
RuleBaseLoader loader = RuleBaseLoader.getInstance();
RuleBase ruleBase = loader.loadFromReader(
      new InputStreamReader(this.getClass().getResourceAsStream(
      "/demo.drl")));

WorkingMemory wm = ruleBase.newStatefulSession();

wm.insert(meldung);

wm.fireAllRules();
```

Drools im Einsatz

camunda
The Business Process Company

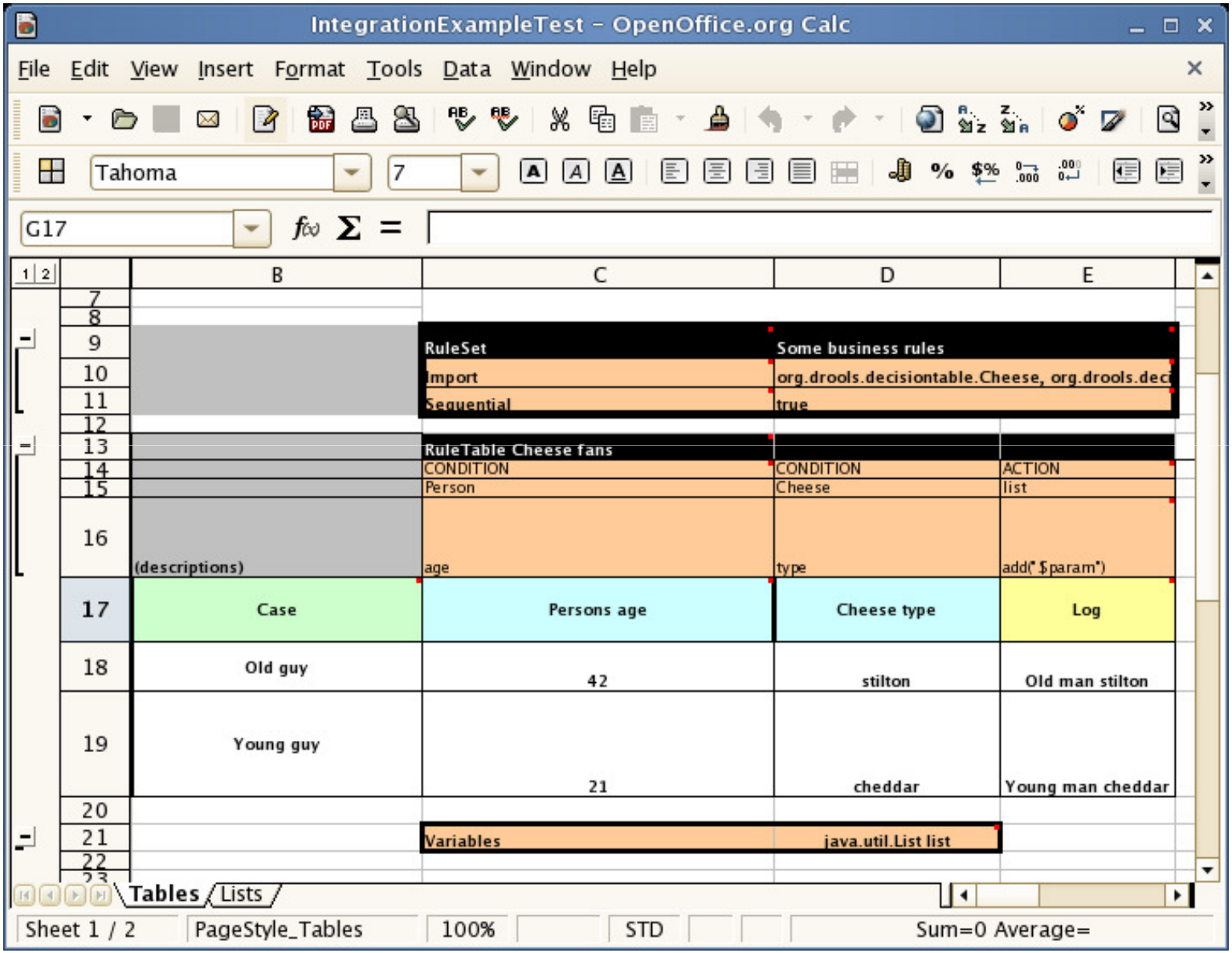
Wie funktioniert es intern?

- In-memory Knowledge-Repository
- Basiert auf Rete Algorithm
- Forward chaining
- Backward chaining ist geplant (Drools 5)



Für den Fachbereich: Decision Tables

camunda
The Business Process Company



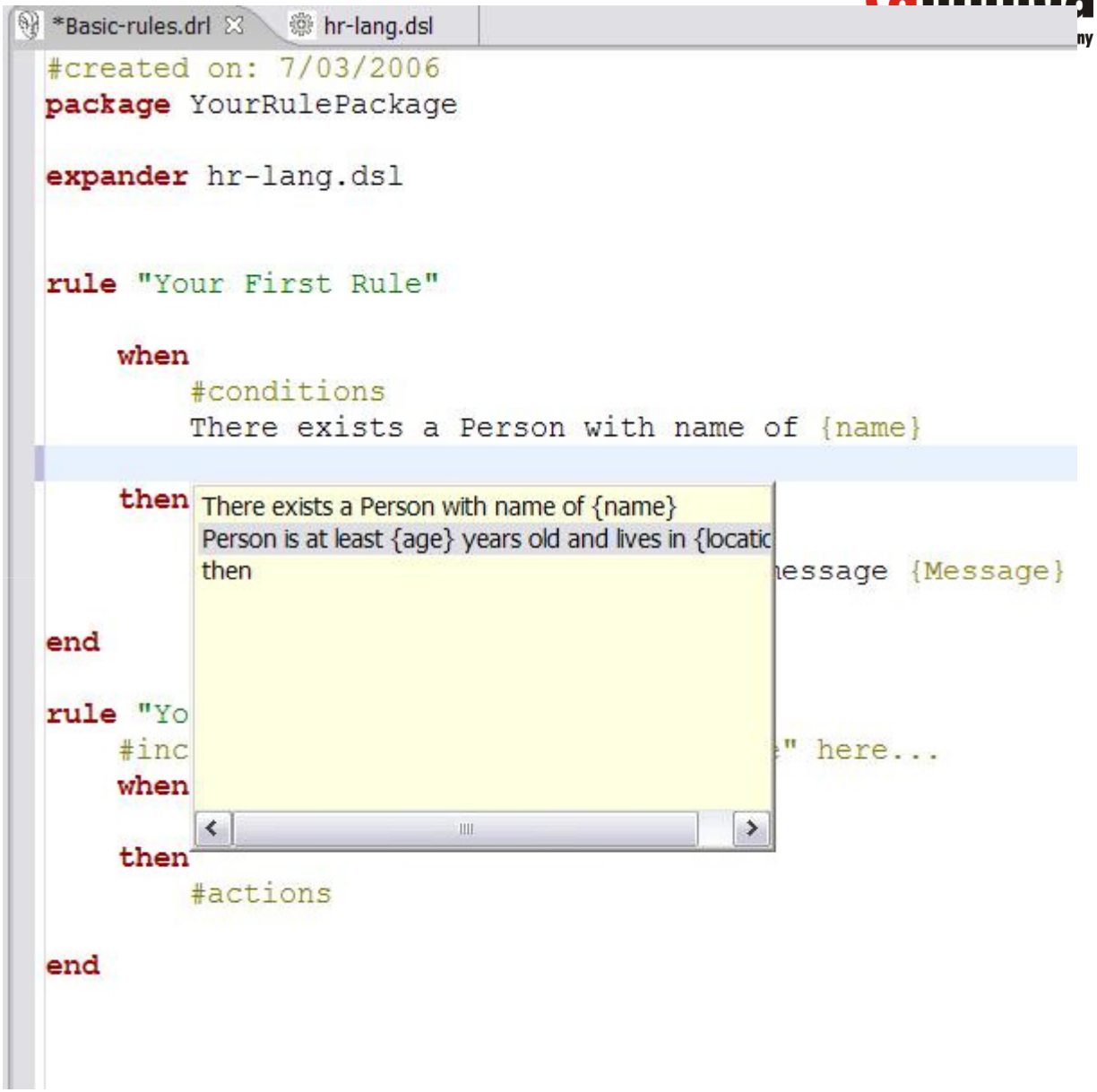
DSL-Support

camunda
The Business Process Company

```
[when]Versicherter unter {MindestAlter} Jahre alt=
    Person( age >= {MindestAlter} )
[when]Mehr als {Anzahl} Unfälle gebaut=
    Person( accidentCount > {Anzahl} )

...

[then]Police nicht möglich=
    errors.addError("Police kann nicht ausgestellt werden");
```

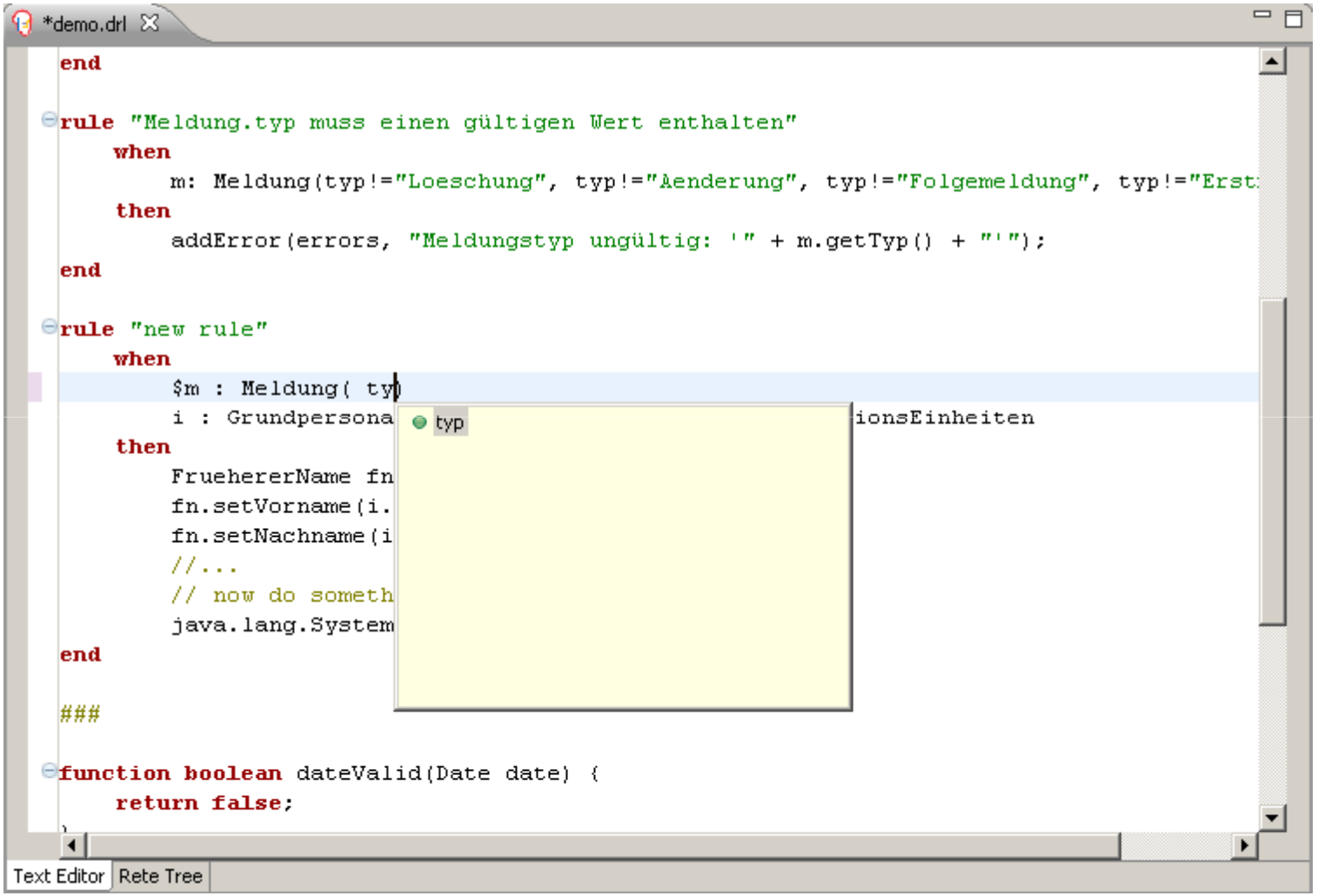
```
rule "Abmeldung - Check mit DSL"
when
  Versicherter unter 25 Jahre alt
  and
  Mehr als 3 Unfälle gebaut
then
  Police nicht möglich
end
```

DSL-Support


*Basic-rules.drl
hr-lang.dsl
#created on: 7/03/2006
package YourRulePackage
expander hr-lang.dsl
rule "Your First Rule"
when
#conditions
There exists a Person with name of {name}
then
There exists a Person with name of {name}
Person is at least {age} years old and lives in {locatic
then
essage {Message}
end
rule "Yo
#inc
" here...
when
then
#actions
end

Regeleditor ohne DSL

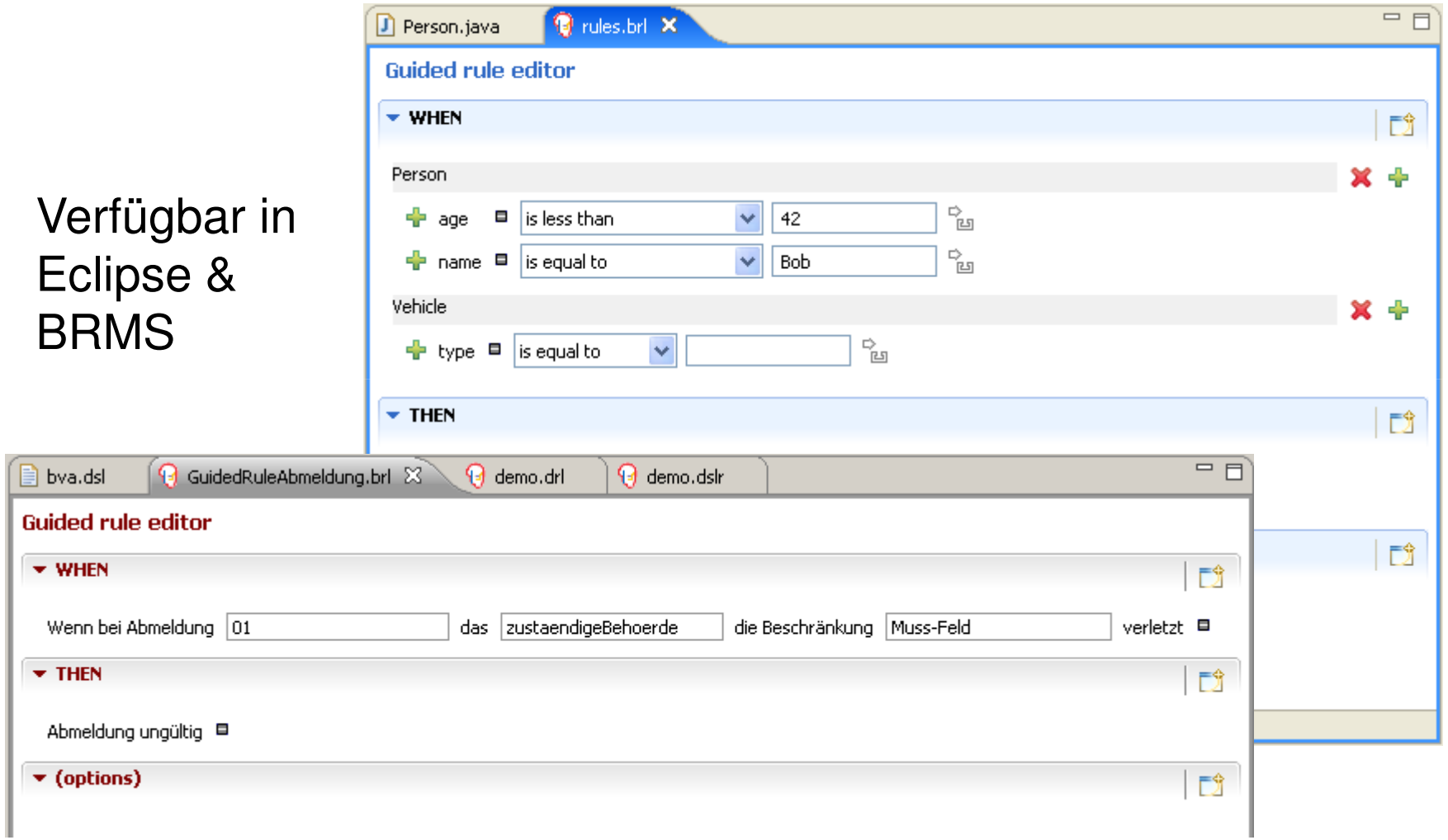
camunda
The Business Process Company



Guided Rule Editor

camunda
The Business Process Company

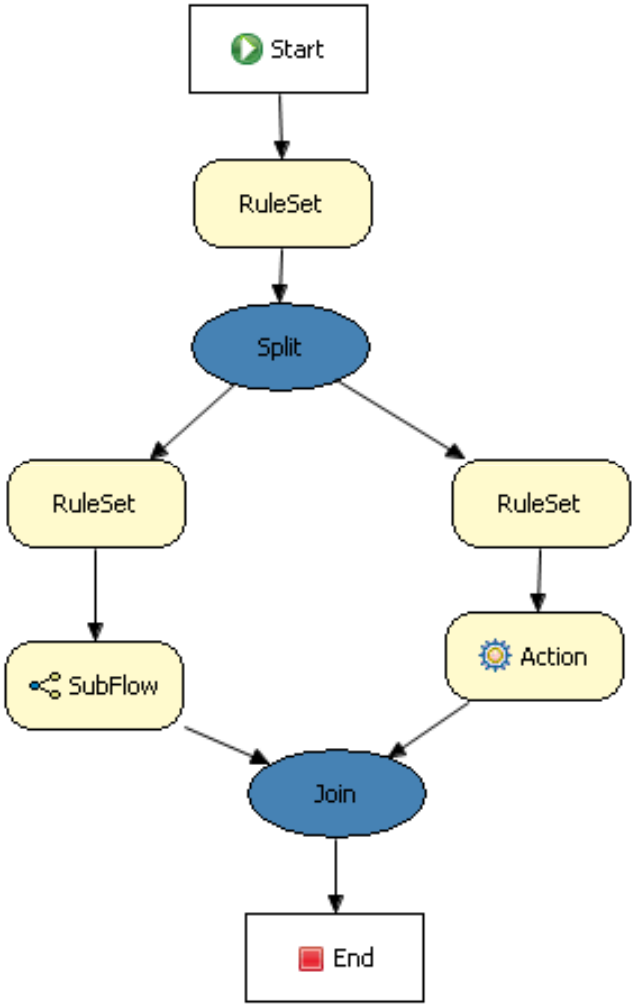
Verfügbar in Eclipse & BRMS



Rule Flow

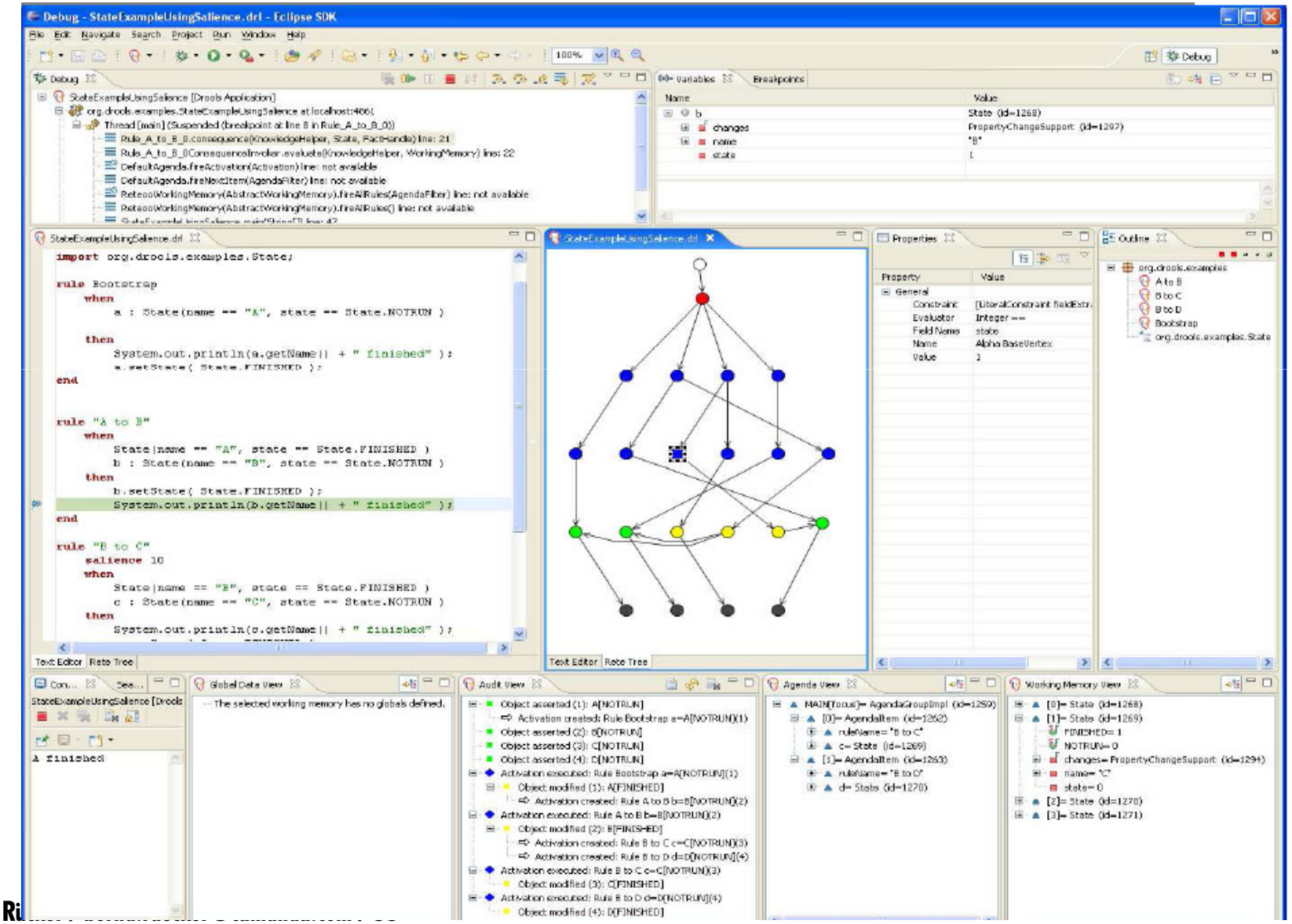
camunda
The Business Process Company

- Regel-Reihenfolge
- Parallelität
- Bedingungen ob Regeln ausgeführt werden



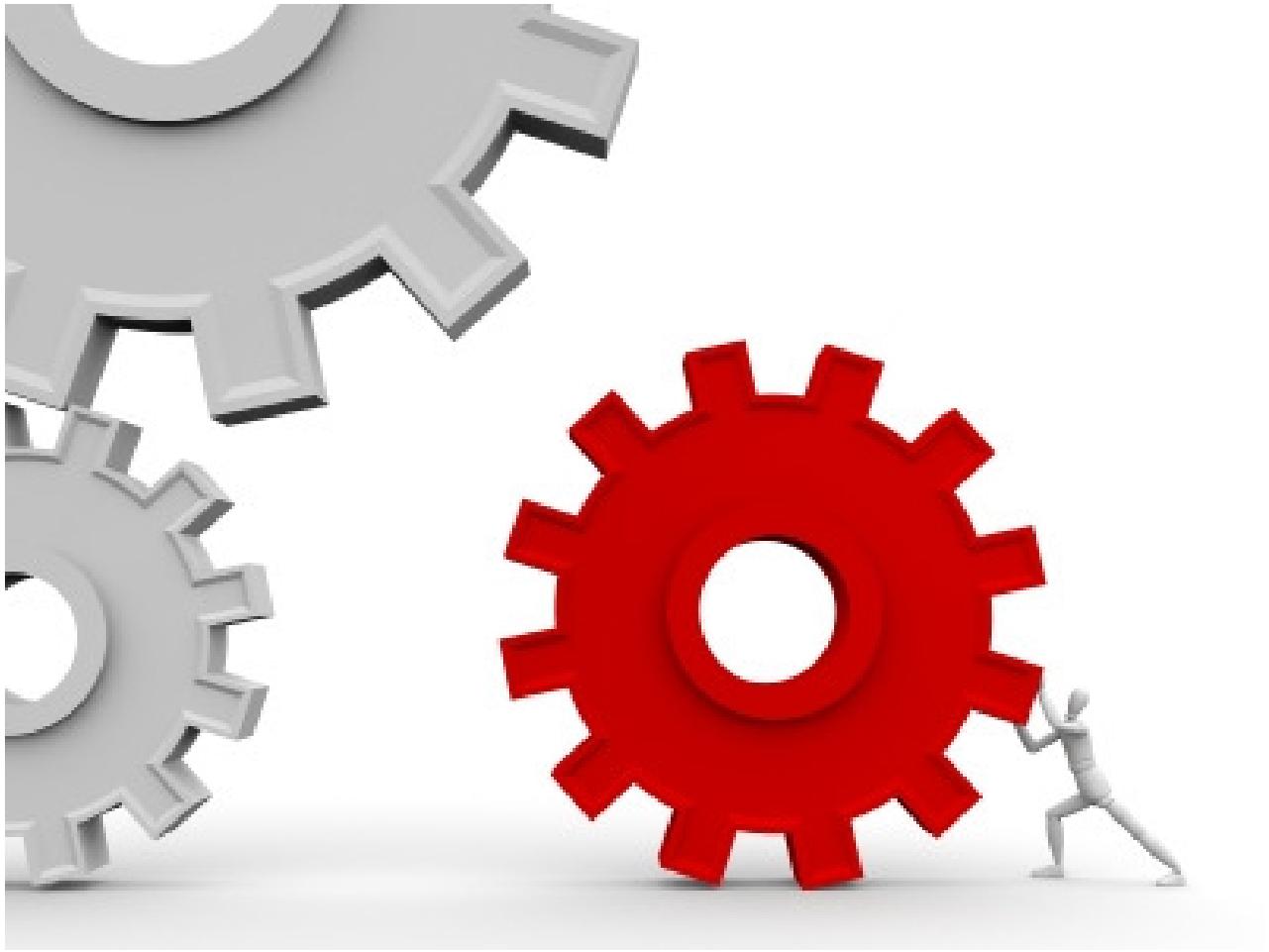
Tooling: Eclipse

camunda
The Business Process Company



BPM vs. BRM | BPM + BRM

camunda
The Business Process Company



Geschäftsprozesse mit Regeln umsetzen?

camunda
The Business Process Company

- Möglich!
- Vorteil: Maximale Flexibilität
- Nachteile
  - Keine Visualisierung / Modellierung des Prozessflusses
  - Nicht immer offensichtlich, warum was passiert
  - Keine „Grenzen“ durch Prozessstruktur
  - Evtl. Performance

Einsatzmöglichkeiten

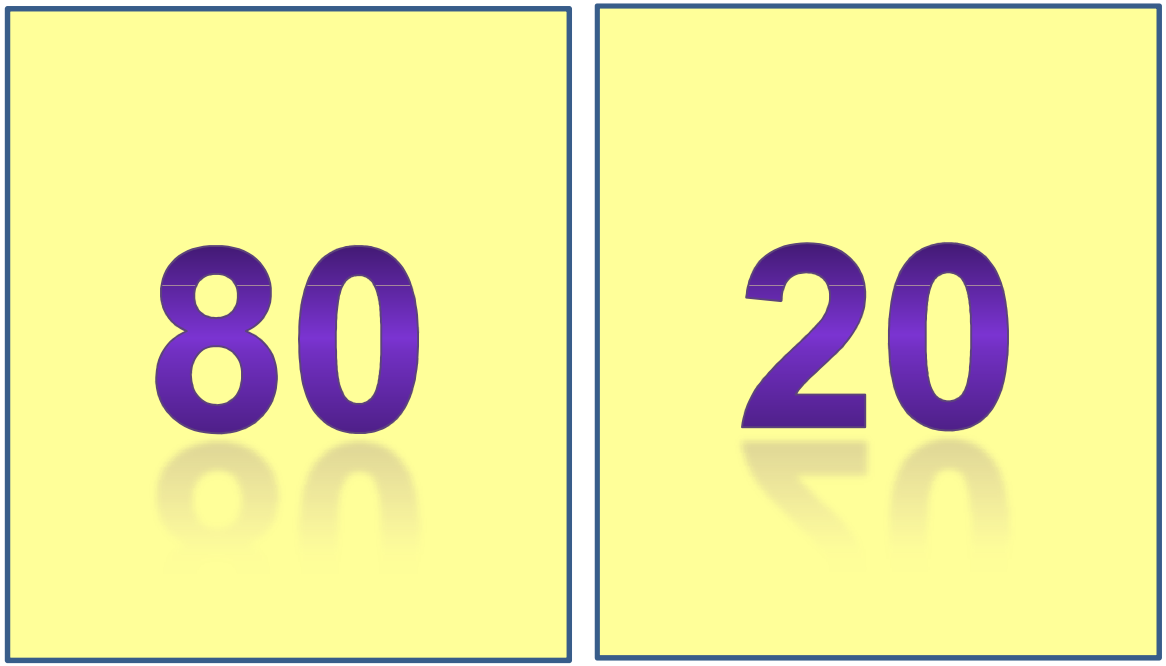
camunda
The Business Process Company

- BPM (Prozesse)
  - Geschäftsprozesse / Workflows
  - Fachliche Modellierung
  - Hoher Standardisierungs-/Wiederholungsgrad

- BRM (Regeln)
  - Prozessunabhängige Regeln
  - Punktuelle Integration in Prozesse
  - Beeinflussung des Prozessablaufs (80/20 Regel)

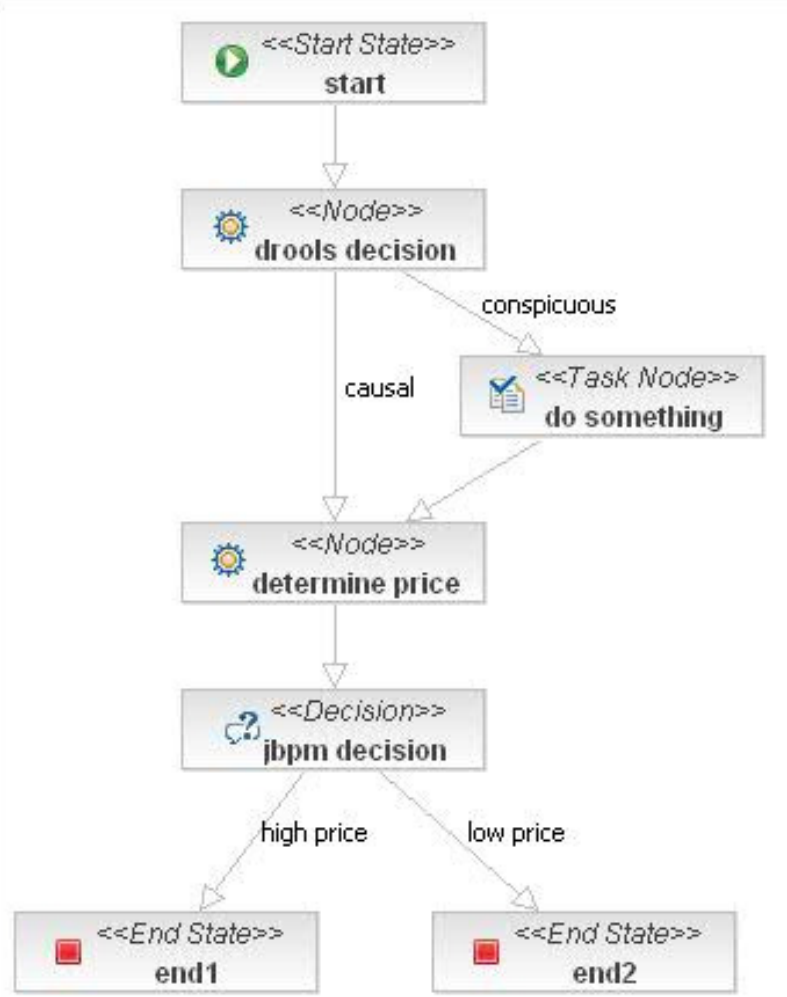
Wann benutze ich was?

camunda
The Business Process Company



Kombinationsmöglichkeiten

camunda
The Business Process Company



Regeln treffen Entscheidungen

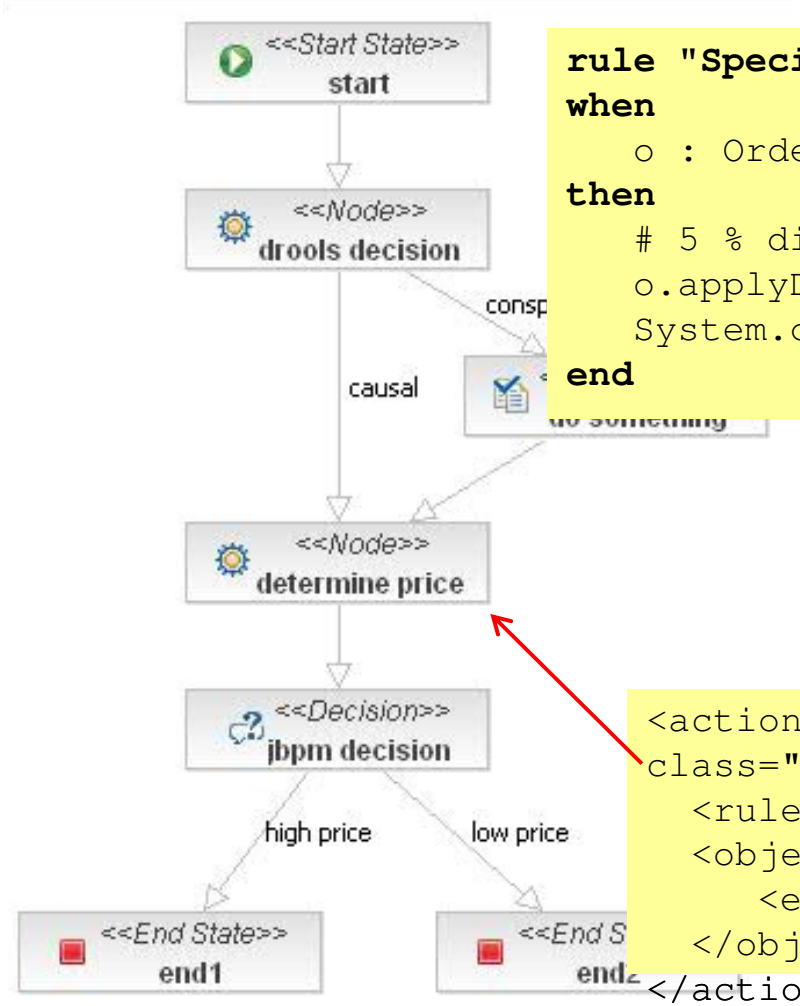
Regeln steuern Zuweisung von Aufgaben

Regeln schließen neues Wissen

Und: Beeinflussung des Prozessablaufs in Sonderfällen

Schließen von “neuem Wissen”

jBPM + Drools




```
rule "Special Discount"
when
   o : Order( customerName == "Bernd" )
then
   # 5 % discount
   o.applyDiscount( 0.95 );
   System.out.println("Special discount granted");
end
```

```
<action
class="org.jbpm.action.RulesActionHandler">
  <ruleFile>/...OrderPricingRules.drl</ruleFile>
  <objectNames>
     <element>order</element>
  </objectNames>
</action>
```

Entscheidungen

camunda
The Business Process Company

jBPM + Drools

```
<node name="drools decision">
  <action class="org.jbpm.action.RulesActionHandler">
    <ruleFile>/com/.../OrderDecisionRules.drl</ruleFile>
    <objectNames>
       <element>order</element>
    </objectNames>
    <signalToken>false</signalToken>
  </action>
  <transition name="conspicuous" to="do something" />
  <transition name="casual" to="determine discount" />
</node>
```

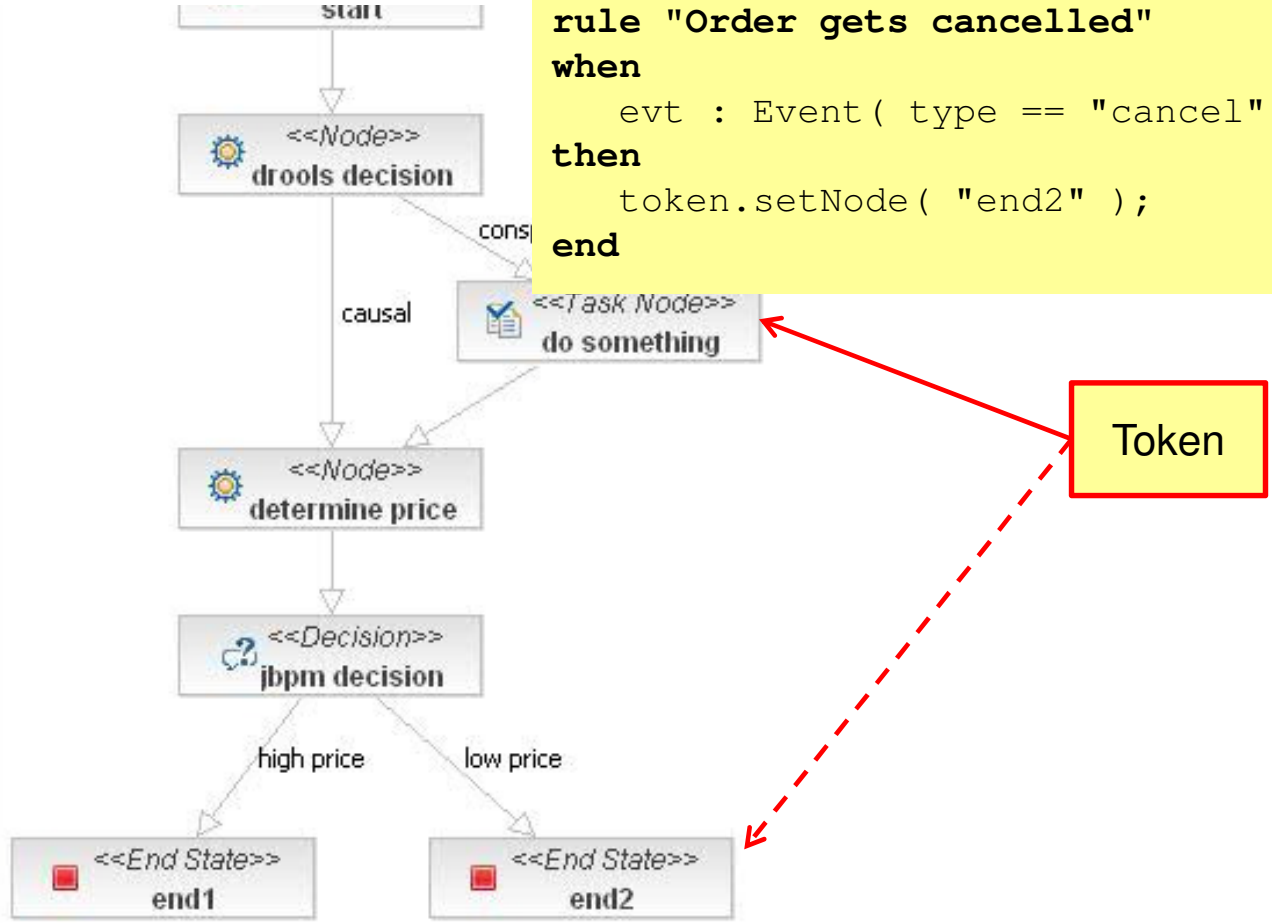
Besseres Design: Regeln schreiben Ergebnis in Prozessvariable und jBPM „Decision“ wertet diese aus

```
rule "Conspicuous Order"
when
  Order( price > 500)
then
  System.out.println("signal conspicuous order");
  executionContext.getNode().
                    leave(executionContext, "conspicuous");
end
```

Beeinflussung des Prozessablaufs

Event getrieben reagieren / Ausnahmen

```
rule "Order gets cancelled"
when
    evt : Event( type == "cancel" )
then
    token.setNode( "end2" );
end
```



Actor-Assignment

Wer ist zuständig?

```
<assignment class="org.jbpm.assignment.RulesAssignmentHandler">
  <group>underwriting</group>
  <ruleFile>/Assignment.drl</ruleFile>
  <objectNames><element>policy</element></objectNames>
</assignment>
```


```
rule "Determine Junior Role"
when
  Policy( basePrice < 500)
then
  insert(new Role("junior"));
end

rule "Determine Actor"
salience -100
when
  Role($roleName : roleName)
  $a : Assignable()
  $group : Group()
  Membership( group == $group, role == $roleName,
              $user : user )
then
  $a.setActorId($user.getName());
end
```

Fazit

camunda
The Business Process Company

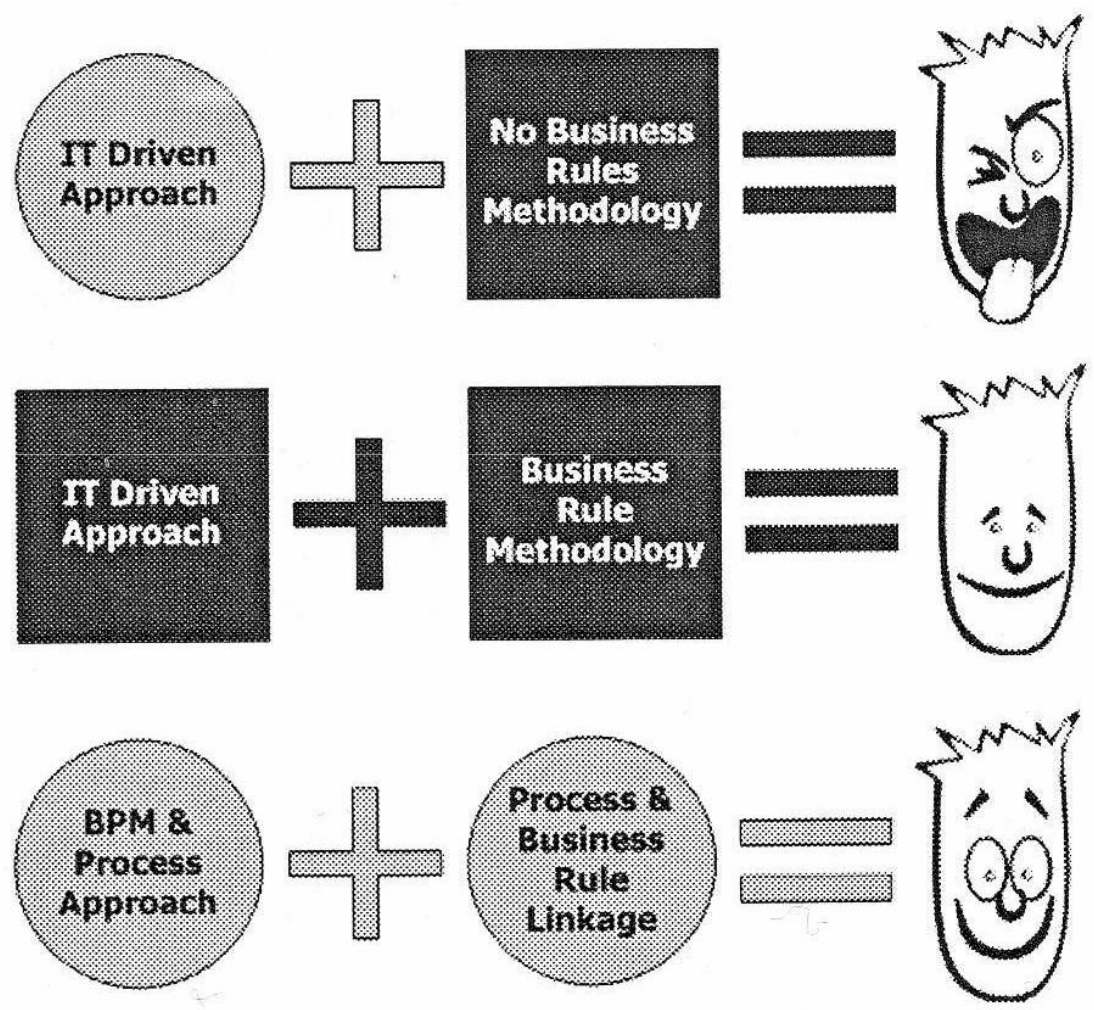
- jBPM & Drools integrieren sehr gut
- Benutzen Sie das richtige Tool für den Job
- Kombinieren Sie!
- jBPM ist eine kleine, flexible Process Engine. Bewährt auch in großen Projekten
- Drools ist cool ☺
- Drools steht teuren Rule Engines in nichts nach
- BPM & BRM sind Themen der Zukunft!

- Ausblick: Integration in den JBoss ESB

Fazit

camunda
The Business Process Company

Geschäftsprozesse und Regeln mit jBPM und Drools



Quelle:
Lary Ward &
Jordan Masanga

Fragen & Antworten

camunda
The Business Process Company

Bernd Rücker
Geschäftsführer
Berater, Trainer & Coach
bernd.ruecker@camunda.com
+49 711 3278645

Unsere Themen

- Ganzheitliches BPM
- Prozessautomatisierung
- SOA, BPEL, XPDL, jBPM, Drools, ESB
- BPMN
- BPM-Toolauswahl

Unsere Leistungen

- Beratung
- Seminare
- Process as a Service (Hosting)