Part I – The Consolidated Past: Drools 5.x

- **Drools Expert**
  - Object-Oriented, Production Rule engine
- **Drools Flow**
  - Complex Event Processing engine
- **JBPM**
  - BPMNv2-compliant process engine
- **Decision Tables**
  - Waiting for DMN
- **Drools Planner** (new! Rebranded and released as OptaPlanner)
  - Metaheuristics Constraint Optimization
- **Guvnor**
  - Repository and Authoring BRMS
Part II – The Upcoming Future : Drools 6.x

- **KIE**
  - Knowledge Integration and Execution Environment
- **Drools – PMML**
  - Predictive Analytics
- **Drools Chance**
  - Fuzzy and other “non-boolean” reasoning styles
- **Drools Shapes**
  - Semantic Web Technology integration
- **Drools MAS**
  - FIPA-compliant agent implementation
About me

- Davide Sottara, PhD
  - Department of Biomedical Informatics
    Arizona State University, Scottsdale (AZ)
  - Background:
    - AI
    - Hybrid Systems
    - Decision Support Systems
  - Drools community member, developer and contributor since 2006

- Community Member - not a JBoss employee
  - The opinions presented here are personal and may not reflect the actual intentions of the Drools development team, or JBoss, inc. their employer
  - Thanks to the members of the Drools team for providing part of the material presented!
Resources

- Main Web Sites
  - www.jboss.org/drools
  - http://www.jboss.org/jbpm/
  - http://www.optaplanner.org/

- Documentation & Demos
  - www.jboss.org/drools/documentation

- More introductory material
  - http://www.jboss.org/drools/presentations.html

- License
  - Open source, ASL 2.0

- Source code
  - www.github.com/droolsjbpm
Drools Expert
Drools Expert

- Heir of the “golden age” of RETE Production Rule Systems:
  - OPSx
  - CLIPS
  - Jess
  - ...

Started as a “friday afternoon” project, sometimes considered just the open source “clone” of Jess, but...
Drools Expert

- Traditional “forward chaining” architecture
Drools Expert

Premise (LHS)

Bound variables

Rule "foo"
when
Pattern /* conditions */
$var : AnotherPattern()
then
/* actions */
end

Conclusion (RHS)

Logical Conditions

Joined Patterns

- Working Memory Actions
  - Insert
  - Retract
  - Modify
- "Side effects"

Premise (LHS)

Bound variables

Rule "foo"
when
Pattern /* conditions */
$var : AnotherPattern()
then
/* actions */
end

Conclusion (RHS)

Logical Conditions

Joined Patterns

- Working Memory Actions
  - Insert
  - Retract
  - Modify
- "Side effects"
Drools Expert

- **Native Java implementation**
  - Object-Oriented engine
  - Object-Oriented rules and facts
    - (Java) Classes, *Interfaces* and *properties* can be used directly
      - As facts
      - As patterns

- **Declarative model**
  - Runtime code generation and compilation

- **Model Import**
  - *XSD* (via JaxB / xjc)
  - *UML* (missing.. anyone?)
  - *OWL*

```java
declare Person
    name : String
    age : int
end

declare Patient extends Person
    mrn : String
end
```
Drools Expert - Advanced features

- **Complex operators**
  - **Quantifiers**: exists, forall, not*
  - **Aggregators**: sum, avg, collect, ...
- **Truth maintenance**
  - Deep retraction
- "In-place" **modifications**
  - Property reactivity (limited refraction)
- **Queries** → backward chaining
  - Limited unification
- **Free-form constraints**
  - Use expressions to filter patterns
Drools Expert - Demo

• Some basic examples...
  Demo from https://github.com/droolsjbpm/drools/tree/master/drools-examples

• Wumpus world!
BRMS: Guvnor

- Centralized Knowledge Repository
  - Web Application
    - Access Control and Security
  - Versioning Repository
  - Assisted deployment and distribution
    - “Hot-swap” knowledge base updates
  - Assisted authoring
  - Integrated testing
Guvnor tooling: guided editor

- “Stylesheet” for technical rule language
- Controlled authoring

```
WHEN
1. There is a Mortgage [m]
2. There is an Applicant with:
   age greater than 70
   mortgage equal to m

THEN
1. Set value of Mortgage [m] approved false
```
Guvnor tooling : Decision Tables

- Compact notation for rule logic
  - Generate DRL from templates

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>salience</th>
<th>name</th>
<th>age</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td>Bill</td>
<td>30</td>
<td>12345</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>2</td>
<td>Ben</td>
<td></td>
<td>&lt;otherwise&gt;</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>6</td>
<td>Weed</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>7</td>
<td>&lt;otherwise&gt;</td>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>

Waiting for the [DMN](#) standard....
JBPM
(previously: Drools Flow)
jBPM

- Business Process Engine
  - BPMNv2 compliant
    - Only a covers a subset of the features

- Native **Rule** support
  - Branching logic
  - “Rule tasks”
  - Fact sharing

- Native **Event** support

- Native Human Task support
  - Based on WS-HumanTask specification
jBPM - Integration

- Management console
  - Process Instance management
  - Active Task Lists
  - Human Task Form management
  - Reporting

- Pluggable persistence : JPA
- Pluggable transactions : JTA
- Pluggable human task service : WS-HumanTask
- Pluggable process repository (optional)
- History logging (for querying / monitoring / analysis)
- Framework support : Seam, Spring, OSGi, ...
jBPM - Architecture

End User

Your Application

Your Services

Eclipse Editor

Developer

Core Engine

History Log

Task Service

Rules

Guvnor Repository

Web-Based Designer

Core Services

jBPM Console
jBPM Designer

- BPMNv2 authoring environment
  - Guided editor
  - Workflow patterns
  - Process Simulation
Drools Fusion
Drools Fusion

- Enables “temporal awareness” in Drools Expert
  - Event semantics
    - **Event**: significant state change at a given time
    - **Complex** Event Processing
      - Detection
      - Processing
        - Aggregation
        - Correlation
        - Abstraction
      - Reaction

```java
declare Encounter
  @Role( event )
  @Timestamp( dateTime )
  patientId : String
  providerId : String
  dateTime   : Date
end
```
Temporal Rules

- Sliding Windows
  - Temporal
  - Length
- Temporal Constraints
  - Allen temporal logic

```java
rule "fusion"
when
...
$dx : Diagnosis( code == "1.2.3" )
accumulate ( $e : Encounter( this after $dx )
  over window:length[ 30d ],
  $num : count( $e );
  $num < 3
)
then
  /* inadequate follow-up */
end
```
Drools fusion – enabled rules

- Scheduling and delaying

```
rule “fusion”
  timer( 1h )
  when
    $dx : Diagnosis( code == “1.2.3”) 
    not Encounter( this after[0, 15d] $dx )
  then
    /* wait up to 15d + 1hr, then alert! */
end
```

- Scalability optimizations
  - Automatic fact retraction
  - Stream processing
Drools Core

Expert + Fusion + jBPM

- Hybrid SOA / EDA architectures

<table>
<thead>
<tr>
<th></th>
<th>SOA</th>
<th>EDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coupling</td>
<td>Loose</td>
<td>Decoupled</td>
</tr>
<tr>
<td>Interaction</td>
<td>Synchronous</td>
<td>Asynchronous</td>
</tr>
<tr>
<td>Coordination</td>
<td>Scheduling</td>
<td>Reaction</td>
</tr>
<tr>
<td>Trigger</td>
<td>Consumer</td>
<td>Producer</td>
</tr>
<tr>
<td>Management</td>
<td>Orchestration</td>
<td>Pub/Sub</td>
</tr>
<tr>
<td>Communication</td>
<td>One-to-one</td>
<td>Many-to-Many</td>
</tr>
</tbody>
</table>
Drools Core

Expert + Fusion + jBPM

- Hybrid SOA / EDA architectures
OptaPlanner
(previously Drools Planner)
Constrained Optimization Problems

- Problem: Maximize/Minimize a goal...
  - Assigning values to variables
    - Degrees of freedom
  - With limited resources
    - (CPU) Time, (Memory) Space, Money, etc..
  - Under a number of constraints
    - Not all variable values, or combinations thereof, are admissible
  - Optimum vs Optimal solution(s)
    - Potentially many solutions satisfy the constraints
    - Some are better than others w.r.t. the goal
Constrained Optimization Problems

- Maximize a \textit{goal}...
  - \textit{when the problem is difficult}
  - \textbf{NP-Complete} Problems
    - Do not “scale well” → impossible to solve \textit{in practice}
    - \textit{But a “good enough” solution may be acceptable}
Drools Planner

- Define a problem
  - Annotated (OO) **domain model**
    - Planning variables
      - Admissible **domain**
      - Owning **planning entity**
  - **Solution**
    - Collection of entities with assigned variables
  - **Constraints**
    - As Drools rules!
      - Scalable engine performance
      - Incremental, Differential evaluation
Drools Planner Solver

- Explores the “solution space”
  - Highly configurable
  - Uses the constraints as guides
- **Meta-heuristic N-phase optimization:**
  - **Initialization** (Global search)
    - Brute Force
    - Best Fit, First Fit, Last Fit
  - **Refinement** (Local search)
    - Hill Climbing
    - Simulated annealing
    - Taboo search
• Planner Demo
  – Nurse Rostering
  – Routing

  – Examples can be downloaded and executed from http://www.optaplanner.org/