Drools: Rule engines in the microservices era

Mario Fusco – Drools Project Lead
Donato Marrazzo – Senior Solutions Architect
Matteo Mortari – Senior Software Engineer
Business Rules are at the heart of every organization

Laws, Regulations and Policies

Drive Process Decisions

Determine Pricing of Products and Services
Separate decision logic from application code
Write once, use anywhere. Agile rule lifecycle management.

Decision logic defined in business terminology and language
Domain experts directly involved in rule definition and writing.

Performance and scalability
From 10 to 1,000,000 rules.
Decision Management and Automation Value across industries

**Banking**
- Loan Organisation
- Credit Decisioning
- Sales Advisory
- Payments
- Accounting

**Insurance**
- Claims Processing
- Underwriting
- Quoting
- Rating
- Commissioning

**Capital Markets**
- Automated Trading
- Trade Order Mgmt
- Accounting
- Compliance
- KYC/AML
- On Boarding

**Public Sector**
- Claims Processing
- Entitlement Calc.
- Benefit Calc.
- Fraud Detection
- Screening

**Telecom**
- Offer Configuration
- Order Mgmt
- Fraud Detection
- Loyalty Programs
- Network Monitoring

**Transportation**
- Promotions Mgmt
- Loyalty Programs
- Customer Service
- Billing
- Contract Mgmt

**Retail**
- Recommendation
- Campaign Mgmt
- Order Mgmt
- Pricing

**Manufacturing**
- Order Mgmt
- Billing
- Contract Mgmt
Red Hat automation products

- Red Hat automation products
- On-Premise
- Private
- Public
- Container
New in Drools 7
Executable Model

- A pure Java DSL for Drools rules authoring
- Automatically generated by maven plugin
- Can be embedded in kjar
  - Faster compilation
  - Backward/Forward compatible
- Allow for faster prototyping and experiment of new features

```java
Result result = new Result();
Variable<Person> markV = declarationOf(Person.class);
Variable<Person> olderV = declarationOf(Person.class);

Rule rule = rule("beta")
  .build(
      pattern(markV)
        .expr("exprA", p -> p.getName().equals("Mark"),
               alphaIndexedBy(String.class, ConstraintType.EQUAL, 1, p -> p.getName(), "Mark"),
               reactOn("name", "age")),
      pattern(olderV)
        .expr("exprB", p -> !p.getName().equals("Mark"),
               alphaIndexedBy(String.class, ConstraintType.NOT_EQUAL, 1, p -> p.getName(), "Mark"),
               reactOn("name"))
        .expr("exprC", markV, (p1, p2) -> p1.getAge() > p2.getAge(),
               betaIndexedBy(int.class, ConstraintType.GREATER_THAN, 0, p -> p.getAge(), p -> p.getAge()),
               reactOn("age")),
      on(olderV, markV).execute((p1, p2) -> result.setValue(p1.getName() + " is older than " + p2.getName()))
  );
```
New in Drools 7

Rule Units

- Declarative approach to:
  - Partition a rules set into smaller units.
  - Binding datasources to a unit.
  - Orchestrate the execution of a unit.

- Aggregate of a data-source, global variables and rules.

- Better coupling between data and rules acting on that specific data.

```java
package org.mypackage.myunit;

public static class AdultUnit implements RuleUnit {
    private int adultAge;
    private DataSource<Person> persons;

    public AdultUnit() {}  

    public AdultUnit(DataSource<Person> persons, int age) {
        this.persons = persons;
        this.age = age;
    }

    // A DataSource of Persons for this RuleUnit
    public DataSource<Person> getPersons() {
        return persons;
    }

    // A global variable valid in this RuleUnit
    public int getAdultAge() {
        return adultAge;
    }

    // --- life cycle methods

    @Override
    public void onStart() {
        System.out.println(getName() + " started.");
    }

    @Override
    public void onEnd() {
        System.out.println(getName() + " ended.");
    }
}
```

---

#RedHatOSD
Business Optimizer

Optimize **Goals**  
With limited **Resources**  
Under **Constraints**
Need for Standards in Decision Management space

- Decisions are a common language across business, IT and analytic organizations improving collaboration, increasing reuse, and easing implementation.
- Business analysts wish to model and improve the decisions that their businesses make.
- Common notation which is understandable by all business users.
- Standardized bridge between the decision design and implementation.
- Usable alongside BPMN business process notation.
- Rules are just a portion of the logic needed to make a decision.
What is DMN?

DMN, which stands for Decision Model and Notation, is a relatively new standard managed by OMG, the organization behind BPMN. It is trying to do for Business Decision Management what BPMN did for Business Process Management a decade ago: empower the business to take charge of the logic that drives its operations, through a vendor-independent diagramming language.

– Bruce Silver, http://methodandstyle.com/what-is-dmn
Decision Model and Notation

Example:

- Application Risk score model
- Application Risk category table
- Decision Node
- Input Node
- Business Knowledge Model

Eligibility

<table>
<thead>
<tr>
<th>Eligibility Rules</th>
<th>Application</th>
<th>Applicant</th>
<th>Employment Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employment Status</td>
<td>Application</td>
<td>Applicant</td>
<td>Employment Status</td>
</tr>
<tr>
<td>Country</td>
<td>Application</td>
<td>Applicant</td>
<td>Country</td>
</tr>
<tr>
<td>Age</td>
<td>Application</td>
<td>Applicant</td>
<td>Age</td>
</tr>
</tbody>
</table>

Eligibility Policy

<table>
<thead>
<tr>
<th>P</th>
<th>Employment Status</th>
<th>Country</th>
<th>Age</th>
<th>Eligibility Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;UNEMPLOYED&quot;</td>
<td>-</td>
<td>-</td>
<td>&quot;NEIGIBLE&quot;</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>not(&quot;UK&quot;)</td>
<td>-</td>
<td>&quot;NEIGIBLE&quot;</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
<td>&lt;18</td>
<td>&quot;NEIGIBLE&quot;</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>&quot;ELIGIBLE&quot;</td>
</tr>
</tbody>
</table>
DMN Big Picture

DMN in context of BPMN

Installment calculation

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Type, Rate, Term, Amount</td>
<td>Installment, Application Risk Score, Eligibility Rules</td>
</tr>
</tbody>
</table>

Monthly Fee

- IF Product Type = "STANDARD" then 20.00
- ELSE IF Product Type = "SPECIAL" then 25.00
- ELSE null

Monthly Repayment

- Monthly Repayment = (Amount * Rate/12) / (1 - (1/Rate))

Routing

- Application Risk Score Model
- Application Risk Category Table
- Eligibility Rules

Application

- Eligibility
- Application Risk

Decline customer

- Offer produce

Routing = DECLINE

Routing = ACCEPT

Collect application data

Decide routing

#RedHatOSD
Architecture example

Decision Central

- Tools for business experts
- Rules
- Tools for developers

Decision Server(s)

- Decision Services

Client app

Author / Test Deploy Expose

#RedHatOSD
Demo
Demo architecture

Decision Server(s)

Account fees calculation request

Account fees calculated
Demo architecture

Decision Server(s)

Account fees calculation request

Account fees calculated
GRAZIE PER L’ATTENZIONE!

Mario Fusco – Drools Project Lead
Donato Marrazzo – Senior Solutions Architect
Matteo Mortari – Senior Software Engineer