What a rule-based program is

- Made up of discrete rules, each of them models a specific aspect of your domain
- Simpler because you can concentrate on the rules for one situation at time
- Flexible in the face of fragmentary or poorly conditioned inputs

Declarative Vs. Imperative
(What to do) Vs. (How to do it)

Features

- Hybrid reasoning
  - Forward and Backward Chaining
  - Reactive, Query and Materialized Views
- Functional Programming
- Truth Maintenance
- Temporal Reasoning
- Complex Event Processing
- Calendar and Timer Support
- Dynamic Object Classification (Traits)

Examples

```java
rule "Check age" when
  $a : Applicant( age < 18 )
then
  modify( $a ) { valid = false }
end

rule "Acknowledgement Failure" when
  $req : BuyReq()
  not BuyAck( this after[15s] $req )
then
  channel["alerts"].send( new Alert( $req ) );
end

rule "Buy More Stock" when
  TradingWindow( status == OPEN )
  Accumulate( $t : Ticker( name == "RHT") over window:time(4m),
              $avg : avg( $t.price ) )
eval( $avg > 100 )
then
  channel["buy channel"].send( new BuyReq( "RHT" ) );
end
```
Drools Integration

... with Apache Camel

Apache Camel is a powerful open source integration framework based on known Enterprise Integration Patterns.

Drools ships with an Apache Camel endpoint implementation for out of the box integration with hundreds of other products that also integrate through Apache Camel.

For a list of Apache Camel components, visit:

http://camel.apache.org/

Drools Server

The Drools Execution Server (drools-server) module is a war, which can be deployed in an application server (such as JBoss AS), and allows applications to execute KnowledgeBases remotely for any sort of client application. This is not limited to JVM application clients, but any technology that can use HTTP, through a REST interface.

The execution server leverages both Drools Camel and Drools Spring modules to support stateless and stateful sessions in a native way.

Check the Drools Integration documentation for more information:

www.jboss.org/drools/documentation.html

Sample Configuration - Camel

Example 1.1. Drools Endpoint configured with the CXFRS producer

```xml
<bean id="droolsPolicy" class="org.apache.camel.component.DroolsPolicy" />
<camelContext id="camelContext" xmlns="http://camel.apache.org/schema/spring">
  <route>
    <from uri="cxf:schema://bean://rsServer"/>
    <policy ref="droolsPolicy"/>
    <unmarshal ref="xstream"/>
    <to uri="drools:model/ksession1"/>
    <marshal ref="xstream"/>
  </route>
</camelContext>
```

Sample Configuration - Spring

Example 2.2. kbase definition example

```xml
<drl:kb base="kbase" node="note">
  <drl:resources>
    <drl:resource source="classpath:/drools/spring/IntegrationExampleTest.xsl" type="XSLT"/>
    <drl:decisiontable-conf input-type="XSL" worksheet-name="Table2"/>
  </drl:resources>
  <drl:configuration>
    <drools:resources/>
    <drl:resources source="classpath:/drools/container/spring/model.xsd"/>
    <drools:configuration enabled="true"/>
    <drools:accumulate-functions/>
    <drools:accumulate-function name="func1" ref="func1\"/>
    </drools:accumulate-functions>
  </drl:configuration>
</drl:kb>
```

... with Spring

Spring is a popular framework for enterprise Java development.

Drools ships with a Spring-ready module that allows Drools to be configured and deployed into the Spring container without any glue code.

For more details on spring, visit:

http://www.springsource.org
Open-source business process management project offering:
- generic process engine supporting native BPMN 2.0 execution
- targeting developers and business users
- collaboration, management and monitoring using web-based consoles
- powerful rules and event integration

Core Engine
Core engine is a workflow engine in pure Java
- Lightweight
- Embedded
- Generic
- Extensible

BPMN2
OMG Specification
- Model, Notation, Execution
- Understandable
- Collaboration and Choreography

Persistence and Transactions
Persistence (JPA, pluggable)
- Runtime Persistence
- History logging
- Services
Transactions (JTA, pluggable)
- Command scoped
- User-defined

Console
Web-based management
- Targets Business Users
Features
- Manage process instances
- User task lists / forms
- Reporting

Integration
Domain-Specific processes
- Extend palette with custom, declarative service nodes
Human-Task Service (WS-HT)
- Task Lists, Task Lifecycle
Task Clients
- Task form generation

- Manage process instances: start new process instances, inspect the state of currently running instances (status window and graphically)
- Design: create/update your business process via graphical representation.
- Test and Debug: test specific process scenarios and temporarily halt process executions to inspect their current state
- Simulate: visualize process progress, manipulate the execution clock, send triggers, validate assertions at specific execution points, and more!
- Design: create and update your business processes in the Web-based environment
- Validate: validate your business processes visually.
- Generate: generate fully executable process and task forms. Generate your process images and view your process in numerous formats.
- Connect: connect to the jBPM Service Repository to install numerous domain-specific service nodes.
- Migrate: migrate your existing jBPM 3 process definitions to BPMN2 with a single click.
Types

Limited Entry
- Body contains boolean states indicating which constraints or actions apply

Extended Entry
- Body contains domain values

Analysis
- Find errors and omissions

Let Guvnor guide you
- Wizard to construct different types
- Avoid errors
- Generate expanded form
Guvnor - Templates

Scaffolding for rules

- Quickly build similar rules
- Author structure with the guided rule editor
- Insert place-holders for variables
- Define values for variables

Flexible authoring

- Keyboard or mouse navigation
- Keyboard or mouse multi-cell selection
- Copy and Paste or rows
- Sorting column data
- Column resizing

- Merging of cells
- Rapid authoring and maintenance

- Grouping of cells
- Hide areas whilst editing others
- Focus attention to sub-sections
Guvnor – Decision Tables

Flexible authoring

- Keyboard or mouse navigation
- Keyboard or mouse multi-cell selection
- Copy and Paste or rows
- Sorting column data
- Column resizing
- Merging of cells
- Rapid authoring and maintenance
- Grouping of cells
- Hide areas whilst editing others
- Focus attention to sub-sections

Integration

- BRL columns
  - Ultimate in flexibility
  - Use BRL fragments in your decision table
  - Mix and match with regular column types

- Use jBPM Work Items as actions
  - Call your favourite jBPM Work Items
  - Pass bound variables in Work Items
  - Use Result Parameters to update Facts
Managing Assets With Guvnor

Guvnor UI Overview:

Managing Assets:

**Asset Attribute View**
- Save/Archive/Copy/Rename/Move/Validate assets.
- View and edit asset meta data
- Asset life cycle management
- Version management
- Discussion and discussion history
- Atom/Feed

**Asset Edit View**
- Each asset type has its own editor to edit asset content.
Managing Packages:
- Package assets list view: List all assets contained by the package based on asset types.
- Package Attribute View – Same as Asset Attribute View
- Package Edit View: Configure, build and deploy packages
Traditional applications fall short on flexibility:

- Disapprove loan when applicant is younger than 0
- Disapprove loan when applicant is older than 70 except if there is a guarantor below 60

With the guided rule editor, users can read, write and redefine business rules themselves:

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

   The following does not exist:
   3. There is a Guarantor with:
      age less than 60
      mortgage equal to m

THEN
1. Set value of Mortgage [m] approved false
```

With DSL, developers can build simple, readable rule pieces and users combine them as needed:

```
WHEN
1. There is a mortgage m
2. Mortgage m has an applicant above 70
3. Mortgage m has no guarantor below 60

THEN
1. disapprove mortgage m
```
QA - Testing & Verification

**Scenario Tests**

- Test Scenarios are written by the rule authors to validate that the knowledge base is working.
- Scenario report shows the test coverage and the test statuses.

<table>
<thead>
<tr>
<th>Scenarios for package: mortgages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run all scenarios</td>
</tr>
</tbody>
</table>

**Overall result:** FAILURE

- Results: 71% (2 failures out of 7 expectations)
- Rules covered: 80% (80% of the rules were tested)

**Uncovered rules:** Row 3 Pricing loans, Row 2 Pricing loans

- Are they old enough: 100% (0 failures out of 1)
- Good credit history only: 0% (Missing Expectations)
- NINjas: 100% (0 failures out of 1)
- No bankruptcies: 100% (0 failures out of 2)
- Pricing low end: 33% (2 failures out of 3)

**Package Analysis**

- Package analysis uses Drools rules to statically analyze knowledge modules.
- It produces a report that contains information about the quality of the knowledge module.

- **Errors** (0 items).

- **Warnings** (11 items).
  - Rule 'RegexDslRule' has no RHS.
  - **Impacted rules:**
    - RegexDslRule
  - Rule 'Dummy rule' has no RHS.
  - Rule base covers == Asset, but it is missing != Asset
  - Rule base covers == 30, but it is missing != 30

- **Notes** (0 items).
Every organization has planning problems, such as:
- Employee rostering
- Task scheduling
- Vehicle routing
- Assembly line optimization
- Bin packing

Yet, they hardly optimize those problems. Why? Because those problems are “NP-complete”: computationally very difficult and humanly impossible to optimize.

A hospital can accommodate more patients when optimized with Drools Planner

### Patient admission schedule

Assign each patient a hospital bed.

<table>
<thead>
<tr>
<th>General ward</th>
<th>Largest admission first</th>
<th>Drools Planner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room 11 bed 1</td>
<td>November</td>
<td>November</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>1-3</td>
<td>1-2</td>
<td></td>
</tr>
<tr>
<td>Room 11 bed 2</td>
<td>E</td>
<td>G</td>
</tr>
<tr>
<td></td>
<td>2-4</td>
<td>5-7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Intensive care</th>
<th>Largest admission first</th>
<th>Drools Planner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room 21 bed 1</td>
<td>November</td>
<td>November</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>1-6</td>
<td>1-6</td>
<td></td>
</tr>
<tr>
<td>Room 22 bed 1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1-5</td>
<td></td>
</tr>
</tbody>
</table>

Better planning algorithms reduce costs, improve service quality and help save the environment:
- Exam scheduling with Planner finds a feasible solution where common algorithms fail.
- Nurse rostering with Planner has on average 53% less unwanted shift assignments.
- Process allocation to computers with Planner has on average 28% less congested resources.

To confirm these numbers for yourself, download Planner and run the examples.
Community Information

- Events and up to date info
- Important links
- Online documentation
- Articles & Samples
- Tips, Tricks & FAQ for users (rule developers)
- Patterns, best practices and guidelines
- Presentations slides and videos

Getting help (and helping out)

- As normal, the maniacs can be found in the drools IRC room at irc.codehaus.org.
- Web client available for those behind a firewall http://irc.codehaus.org
- 2 mailing list:
  - "user" list for end user issues
  - "dev" for questions about with the internals
- Jira to check existing issues and log yours is available at https://issues.jboss.org/browse/JBRULES

Download and Install

- Download binaries, documentation, examples and sources in a single file
- Projects are available on both the central Maven repository and on the JBoss one
- Latest releases readily available including Beta and Candidate ones
- Drools plugin for Eclipse available for download as part of the Jboss Tools at http://www.jboss.org/tools/download.html or using the standard Eclipse Marketplace Client
- https://community.jboss.org/wiki/JBossRules
Branding

JBoss BRMS is the branded product offering from Red Hat; for which you can buy support subscriptions and professional services, such as training and on-site consultancy. JBoss itself is a Red Hat brand for its Java middleware.

JBoss BRMS is made up from a number of community projects - Drools Expert, Fusion, jBPM and Guvnor.

Multi-Year Support

Red Hat commits to supporting a product release over a large number of years, the length varies from product to product. Cumulative patch releases ensure you get the fixes without the risk of new features. And the assurance that we will be there for the long term. .org does not do maintenance releases it’s a continuously forward moving R&D project. If you want the fixes you are going to get all the new features too, and the risks that come with them.

Getting community support for previous .org releases can be hard, leaving you to the mercy of a volunteer based community.

Sanity through Sanitization

A common fear with OSS is due to the transparency of R&D. From casual inspection this can seem quite hairy as end users are exposed to all the unstable, unfinished and experimental works. This leaves a lot of uncertainty and fear in using OSS in a production environment.

Closed source companies get to do all their R&D behind closed doors, and end users are only exposed and aware of the highly polished marketing.

.com addresses this issue by removing or demarcating experimental or unstable features. To ensure there is a level of sanity and trust in what you are using. The product is typically 3 to 6 months behind in features, but offers strong levels of stability as a result.

Unburden R&D

Everything that goes into ensuring a great product for the enterprise version entails a lot of work. You need a lot of resources that have meticulous attention to detail. Their priorities will be different to the R&D developers priorities. Stability and maintenance are the enemy of innovation. To ensure we have continued innovation at a project level it’s important that we isolate R&D from these pressures. So everything about the product process is about freeing up the .org R&D developers so they can focus on ideas and innovations.

Multi-Year Support

Released products are tested across the range of JBoss and Red Hat products and also multiple versions. So if you are using JBoss BRMS you’ll know it’ll work on the range of AS services, or the JBoss SOA stack.

We also check compatibility across other platforms such as WebSphere and even IBM z/OS. .org. Community releases do not go through this compatibility matrix level of testing.

Community Independence

By separating .org from .com it ensures that R&D can be bottom up user and community driven. This ensures a healthy eco system for ideas and innovation and collaboration - compared to a top down marketing driven model. The life cycle of .com and .org feeding into each other ensures the best of both worlds and help’s maintain an important but delicate balance of innovation versus sanity and stability.

I should add that .org is still relevant and important for .com customers as it provides an open environment for them to get involved and upstream their work, which may eventually end up in the product.

Direct Impact on Road Map

Customers you have a direct line to influence roadmaps and bugs. Resources are prioritized for customers versus community jira’s or mailing lists posts. It is also the only way to get Red Hat engineering resources of legacy releases, they will ignore .org legacy issues in the community.