Drools White Paper
Table of Contents:

Introduction .............................................................................................................................................. 2
The rule/drools paradigm .......................................................................................................................... 3
Drools content and tooling ...................................................................................................................... 6
Architectural design ................................................................................................................................. 7
Development Process Impact .................................................................................................................. 11
Drools platform and OpenESB ............................................................................................................... 13
Use Cases of a Rule engine and why the need of a rule engine is now more needed than in the past?..... 17
Introduction

In Information Technology Industry, there have been technical tools or paradigms that made big impact on how to implement Business requirements. The best know are certainly the Cobol language, relational databases, web technologies, the java language and many others.
During all that period, many tools/paradigms were created and had only a limit usage to very particular cases. One of them is the rule engine paradigm that now comes up again and gains more usage.
We shall first present the rule paradigm and what it changes in our way of implementation business logic.
We shall then present the drools framework and the tooling around it and how it can be used in a project.
We shall present an implementation with all the tooling we need to develop for each project.
Using drools will have a big impact on how we organize our projects. We shall expose how team can be organized and how this can help us in doing agile development.
There are tools that will help us in accelerating drools project and will reduce the development time needed for integrating drools in projects. This tools will help us to have the best usage of drools : the drools platform and OpenESB.
We shall finish our paper with a few criteria that we shall consider to use drools (or not).
The rule/drools paradigm

In classical software languages, the developer writes code in a procedural way. Even objet-oriented languages are using “procedural code” to implement the business logic. Therefore, it is only up to the developer’s skill to implement correctly the requirement. As in technical logic implementation, “procedural” software language fits very well like, user interfaces, database call, etc. It is easy to translate from a requirement that is expressed in an easy way to implement like user interfaces drawings.

In business logic implementation, it is more complex. Most of the time, the business requirements are composed of a list of rules that applies to a non-explicitly defined entity model. For example, to calculate something or to make a decision, date is needed and on this data, rules have to be applied. The business do not express in which order the orders have to be applied.

In a “procedural” approach, there are many design patterns that will be developed to make rule implementation easy and to be sure that all rules will be verified. But most of the time, the “if-then-else” code will used a lot and there will be cases where some rules could have been applied but were forgotten.

The rule paradigm we are introducing is a “declarative” implementation. The only thing the developer has to implement is the rule itself, which consists of condition part and an execution part (when-then).

On top of this, a rule applies to facts (A fact is an instance of an object, java class for example). It is of course possible to update, retract or insert a fact in the then part of the rule. It is now easy to see how the rules can make other rules applied, etc.

---

1 This is another source in something called the « spaghetti » paradigm.
Here is an example of two rules (in drools).

```
Rule
  when
    Cheese( $cheddar : name == "cheddar" )
    $person : Person( favouriteCheese == $cheddar )
  then
    System.out.println( $person.getName() + " likes cheddar" );
end

Rule
  when
    Cheese( $cheddar : name == "cheddar" )
    $person : Person( favouriteCheese != $cheddar )
  then
    System.out.println( $person.getName() + " does not like cheddar" );
end
```

The main question that arises immediately is that approach optimized where the number of rules is important and the rules complex. Behind the scene, an algorithm is implemented for that. The most implemented one is the “Rete” Algorithm which was created by Dr Charles Forgy on in PhD work in 1983. You can read more on him here\(^2\).

The purpose of the rete algorithm is to represent as a network the rules and then be able to know the consequences a rule can have on data manipulation (by inserting a new fact, updating or retracting an existing fact) but also when I rule can be fired.

Here is a simplified graph representation of the two rules described above.

Even the conditions are represented in the network graph. Therefore, the Rete algorithm is very efficient even when the numbers of rules are increasing a lot. This algorithm is implemented in many variants by many rule engine open source or not. If you want more reading about this, I invite you to read that article³

³ http://blog.athico.com/2012/05/drools-54-artificial-intelligence.html
# Drools content and tooling

Drools project was created in 2001 as a codehaus project\(^4\). Jboss engaged the committer in 2005, Redhat acquired jboss in 2006.

Drools is composed of 5 parts:

<table>
<thead>
<tr>
<th>Tool</th>
<th>What it does and How to use it</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drools Expert</td>
<td>This the the rule engine itself</td>
</tr>
<tr>
<td></td>
<td>Drools Expert is a java API that can used from a java program. It can be included in a java project using all well-known configuration tool like maven.</td>
</tr>
<tr>
<td></td>
<td>To be able to use it, there must be an integration phase. In the 6.2 version, there a rule server but exposes over rest the native Drools Java API interface. The deployment of new version of a rule set must be programmed.</td>
</tr>
<tr>
<td>Drools Fusion</td>
<td>Complex event processing features.</td>
</tr>
<tr>
<td></td>
<td>In the Java API, there are methods dedicated to that part</td>
</tr>
<tr>
<td>Drools Workbench</td>
<td>Web UI for authoring and management. This part is often called BRMS (Business Rule Management System). This part is a web application that allows to handle the complete lifecycle of a rule</td>
</tr>
<tr>
<td>Guvnor</td>
<td></td>
</tr>
<tr>
<td>Jbpm</td>
<td>Process/workflow integration for rule orchestration. It is very useful in complex rule projects where it becomes possible to organize rules.</td>
</tr>
<tr>
<td>OptaPlanner</td>
<td>automated planning system</td>
</tr>
</tbody>
</table>

\(^4\) [http://docs.codehaus.org/display/DROOLS/Home](http://docs.codehaus.org/display/DROOLS/Home)
Architectural design

As it can be seen from the previous part, drools out-of-the-box is capable to realize core functionalities

- The complete authoring and lifecycle of a rule set can be done through the BRMS part.
- The runtime itself is fully functional and can be used in a java program.

Still in a real project, many other components are needed to integrate drools in an IT project:
<table>
<thead>
<tr>
<th>Component</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT Application</td>
<td>The rule Engine can be called from one or more applications. In the other direction, during rule execution, one or more external systems may be needed to grab necessary data.</td>
</tr>
<tr>
<td>Logging DB</td>
<td>During its execution, logging are produced to described what is happening (which rule was fired, what data was given, is produced, etc..). This data is then stored in the DB. This data is essential to prove and explain the decision that were taken.</td>
</tr>
<tr>
<td>Monitoring Tool</td>
<td>This tool has two functionalities 1) search in the data logged and manipulated by the rule engine (given as an input) 2) Query and display KPI</td>
</tr>
<tr>
<td>Authoring tool (Guvnor)</td>
<td>Authoring tool CRUD all rules, decision tables, test scenarios etc. Rule Repository that handles the compete lifecycle of the rulese.</td>
</tr>
<tr>
<td>Administrative Tool</td>
<td>Tools that allows to 1) Runtime sanity, Promote rule version 2) Build version of rule sets and deploy them on the fly to any of the runtime.</td>
</tr>
<tr>
<td>Drools Runtime</td>
<td>The rule Engine can be called from one or more applications. In the other direction, during rule execution, one or more external systems may be needed to grab necessary data.</td>
</tr>
</tbody>
</table>
applications. In the other direction, during rule execution, one or more external systems may be needed to grab necessary data.

These runtime shall expose business Service not native drools runtime. The Core drools Runtime is included. They shall also send all logging informations to store it in the logging DB.

With such a complete set of tools wrapped around drools, we are ready to run a real drools project.
As it can be seen, there is a lot of specific development that is needed:

- All the wrapping around the runtime,
- The logging,
- The monitoring tool,
- Etc.
Development Process Impact

When using drools and its Web authoring tool, it is possible to run in parallel the IT project and the rule project. Indeed, the rule project just needs an Entity project to apply the rules on. As we are working in java, it is going to be a small library (jar file). Then within Guvnor, we will be able to develop the rules using all available set of rules in Guvnor: business rules, rule templates, Decision Tables, etc. And in the same tool, it is possible to use the test scenario feature that allows to test all rules without the need of another system. It works with the given/Expect principle. In the next picture, the usage of drools allows the IT software to be developed without rule implementation problems and the same for the rule Implementation.

The two teams can work in parallel and there can be join periods (for testing). This fits very well in agile development teams like Scrum.
Here is an example with the given rules and the test scenario associated:

As it can be seen, the rule is if there is an object of type “Account” we insert an object of type Customer and set its attribute name to “Hello”.

In the following test scenario, we verify it.

It is very convenient to design all rule implementation, test it standalone and then when the IT project is ready, integrate it.
Drools platform and OpenESB

Two Open Source project can help a lot when integrating drools in a project:

- The “chtijbug community”\(^5\) and their drools platform
- The “OpenESB community”\(^6\) and their integration tool

With the same example, we now have the following diagram:

\(^5\) http://www.chtijbug.org
\(^6\) http://www.open-esb.net
When using drools, external data may be needed. It is a common practice to gather these data in the same java program where the drools runtime is running.

- The drools platform offers a maven plugin to generate a drools runtime from a wsdl/xsd web service definition. It generates a java web service application that can be deployed in an application server like apache tomcat 7. At the same time, it creates a project in the BRMS tool (Guvnor) and uploads there the Entity model so that now Business Analyst can start author the business rules.

- The drools platform offers an administrative tool and a logging functionality that implements all needed features without any additional integration development.

- On the other side, with the usage of OpenESB tooling, it is possible by configuration to define service orchestration to grab the needed data from all possible sources using the rich component set given by openESB. Then a call can be done to the drools runtime using the web service runtime generated and then the final result can be sent to the right application. It is even possible to update all needed applications with the result.

The only thing we have to configure/develop

- Pojo model and all the rules
• Generate a Runtime using the xsd/wsdl or pojo defined. All the logging & Co functionalities, deployment and building of version of rules are included by default with the drools platform.

• Configure an OpenESB application to be called and that does all the low level actions.

Here is an openESB application that calls the rule engine with the soap connector and then calls a database with the database Connector (insert, update, etc.)
Here is a picture of what can be displayed when using the drools platform.

In future release of OpenESB, the drools platform runtime will be integrated as a Service engine in two modes:

1. As a rule engine with rule flow
2. As a bpmn2 engine that will allow bpm process to run (including Human task, etc.).
Use Cases of a Rule engine and why the need of a rule engine is now more needed than in the past?

In the last decade, Rule engine have been used where the important costs of such tools (most of the time proprietary) was compensated by the added value they bring. But they can only fit in big projects.

Another field where rule engines are used is in integration project where most SOA tools offer a rule engine. But in most cases, the usage of a rule engine is very limited.

In the last decade, drools (and few others) as a rule engine is trying to offer a valuable alternative to those proprietary solutions. The maturity of those open sources tools and the tools they offer are now very near to the legacy solutions. Up to now, the companies behind those open source tools are competing against those legacy tools on the same kind of projects and with a lot of success as their TCO (total cost of ownership) are much smaller than the proprietary solutions.

But the draw back of that competition is that all other types of projects (and size of budgets) will never be able to afford such tools.

People who want to use a rule engine like drools only have two alternatives: not using it or find people that can help them to use such tools. The open source community and the Internet technology now offer the possibility to connect customers to passionate people that offer services on drools. And as drools becomes much more known and used, more and more people have the knowledge and the know how on it.

In the software industry, traditionally you were buying a tool (or a support for the tool) and you install it on you own machines (in house or in the cloud). Now more and more software are proposed and sold as SAS (Software as a Service). Those software can be connected to the legacy systems of the customer and access all the data. Therefor, they are able to grab data of various systems in-house and on the web to get additional information (like the commercial register or companies calculating risks on companies, etc.). They can now make decisions on much more data now available on-line and at the time it is needed.

To implement all that business logic, traditional implementation in a development language will cost too much and is not agile enough. Now, a new partner (it can be a data provider as well as a new customer) must be integrated in the IT system in days or weeks but no more. It is where the usage of a rule engine like drools fits the best. Drools even fits well as it offers a bpm tooling to implement Complex
Business Process with Human task interaction possible. So when a new partner/customer comes in it is possible to create a business process for him and the rules around to integrate its IT systems to ours.
About LogiCoy

Our Clients are Our Number One Priority

The Open Source Philosophy

Best Choice for Drools Support

The Future with LogiCoy

Summary and Next Steps
About LogiCoy

LogiCoy is a California-based corporation established by former Sun Microsystems Senior Directors, Architects and Developers. With offices in the US, UK and India, LogiCoy offers 24/7 support services and has extensive expertise in providing SOA-based Enterprise Application Integration and Composite Application-based solutions. Our products and solutions are based on various open source technology stacks including OpenESB, Mural MDM, OpenEMR, NHIN Connect, MySQL, etc.
Summary and Next Step

As a leading company in developing world-class products and supporting OpenESB community, LogiCoy consistently dedicates to provide service and products with best quality and lowest cost. Discover more about our products and services visiting our Website (logicoy.com) and our LinkedIn and Facebook pages!

You can also contact us via email or telephone

- Fred Aabedi, CTO: fredaabedi@logicoy.com
- Geraldine Etienne, Business Development Manager: geraldine.etienne@logicoy.com
UK Office:
3 Charlton Lodge, Temple Fortune Lane, London, NW11 7TY, UK
Phone: +442081237299

US office:
10850 Wilshire Blvd. Suite 1120, Los Angeles, CA 90024 USA
Toll Free: 877-44LOGIK or 877-445-6445
Direct: 626-278-4085