Drools at Intermountain Healthcare

Herman Post
Homer Warner Center
5/9/2011
What is Drools?

- Drools is a “Business Logic Integration Platform which provides a unified and integrated platform for Rules, Workflow and Event Processing.”
- Currently represents 4 types of logic:
  - Expert System – “production rules system”
  - Workflow or BPMN2
  - Complex Event Processing (CEP)
  - Solver – network, scheduling based problems
Drools Tools

- Guvnor – web based BRMS, authoring, testing, versioning, security, repository
- Eclipse plugins – rules, BPMN
- Decision tables – map to rules
- Domain Specific Language (DSL)
What is Drools?

- Open source (Apache License) started in 2001 under SourceForge
- Drools federated under JBoss in 2005, JBoss acquired by RedHat 2006 (Linux distro)
- JBoss Rules is commercially supported version of Drools
  - Testing, training, support
- Drools core team of 5 supported by JBoss, managed as open source project
- Drools has 2 independently published books this year
Drools Resources

- [http://www.jboss.org/drools/](http://www.jboss.org/drools/)
- Documentation
  - [http://www.jboss.org/drools/documentation.html](http://www.jboss.org/drools/documentation.html)
- Downloads – documentation (5.0)
  - Best source – start with Expert
- Blog – releases, significant info
  - [http://blog.athico.com/](http://blog.athico.com/)
- Forum – very active, good info
  - [http://www.nabble.com/drools---user-f11823.html](http://www.nabble.com/drools---user-f11823.html)
- Wiki – see ‘Articles’, ‘Tips, Tricks & FAQ’
**Expert – Drools Rules**

- **Production Rules System**
  - Roots in Artificial Intelligence from 1970’s
  - Early systems include
    - MYCIN – Stanford U., diagnosis of bacterial infections; 500 rules diagnosing 100 infections
    - XCON – Carnegie Mellon, used by DEC for many years for complex systems configuration (VAX)
    - Derivative of CLIPS, JESS
  - Model human reasoning
  - Handle 100,000’s of rules efficiently – Rete algorithm
• “The term "Production Rule" originates from formal grammar - where it is described as "an abstract structure that describes a formal language precisely, i.e., a set of rules that mathematically delineates a (usually infinite) set of finite-length strings over a (usually finite) alphabet" ([wikipedia](#)).”
  • `<integer> ::= ['-'] <digit> {<digit>}`
  • `<digit> ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
• What does this have to do with Drools?
  When
    `<condition>`
  Then
    `<actions>`
A Production Rule System is Turing Complete with a focus on knowledge representation to express propositional and first order logic in a concise, non-ambiguous and declarative manner. The brain of a Production Rules System is an Inference Engine that is able to scale to a large number of rules and facts. The Inference Engine matches facts and data against Production Rules - also called Productions or just Rules - to infer conclusions which result in actions. A Production Rule is a two-part structure using First Order Logic for knowledge representation.” (from Drools Expert)
Production System

Inference Engine
(ReteOO / Leaps)

Production Memory
(rules)

Pattern Matcher

Working Memory
(facts)

Agenda
Forward Chaining
rule "Glucose <= 40, Insulin On"

when
    $msg : GlucoseMsg(glucoseFinding <= 40, currentInsulinDrip > 0 )
then
    glucoseProtocolResult.setInstruction(GlucoseInstructions.GLUCOSE_LESS_THAN_40_INSULIN_ON_MSG);

end
Rule Semantics

When

<one or more facts in Working Memory, or data from external sources, matches this rule>

Then

<update, insert or retract some facts in Working Memory>
Advantages of Rules Engine

- Declarative Programming
  - ‘what’ not ‘how’
  - First Order Logic without imperative language (1.3.1)
- Logic and Data Separation
  - Loose coupling of business logic and applications
- Speed and Scalability
- Centralization of Knowledge
- Knowledge Engineering Tools
- Understandable Rules
- Common Rule Syntax (shareable)

(from Drools Expert 1.2.1)
Drools Guvnor

- Business Rules Management System (BRMS)
- Knowledge asset repository
  - Versioning, build and package
- Security
- Authoring
  - Guided rules, models
  - Test scenarios, analysis, justification
  - Decision tables
  - Domain Specific Language (DSL)
- Eclipse integration
Drools Workflow ➔ jBPM5

- JBoss merger of two workflow solutions:
  - Drools Workflow and JBoss jBPM5
    - Business Process Management Notation 2 (BPMN2)
    - JBoss Business Process Management 5 (jBPM5)
  - Drools source moved to jBPM project
  - Runtime still integrated into Drools Process Engine
“Drools Flow is a workflow or process engine that allows advanced integration of processes and rules.”

“…end business users struggle understanding the differences between rules and processes, and more recently rules and event processing. For them they have this problem in their mind and they just want to model it using some software.” (Flow documentation)
Drools Flow Features

- Sequencing of
  - Rule groups – rules and flow seamlessly integrated
  - Processes – anything you want (in code)
  - WS HumanTask 1.0 implementation
  - Domain specific Workitems – preconfigured Processes that appear as tools
  - Sub-flows
Drools BPM Features

- Timers
- Internal and external events
- Branching Logic
- Drag&drop authoring
- Process Management Console (5.1)
- Eclipse Human Task Plugin
BPM Features

• *Explicit, declarative, stateful* programming model
• Trend towards stateless, simplified programming
  • REST, Spring
• Do we sometimes force state onto clients just to make our services stateless?
• Some problems are stateful
  • Long running clinical protocols
• Current Foresight model – state is protocol dependent stack of AlertEvents – each protocol is unique
• BPM is an explicit model for stateful programming – all problems have the same toolset, look the same
Drools BPM
"Complex Event Processing, or CEP, is primarily an event processing concept that deals with the task of processing multiple events with the goal of identifying the meaningful events within the event cloud. CEP employs techniques such as detection of complex patterns of many events, event correlation and abstraction, event hierarchies, and relationships between events such as causality, membership, and timing, and event-driven processes."

-- wikipedia
Drools Fusion

- Support Events, with their proper semantics, as first class citizens.
- Allow detection, correlation, aggregation and composition of events.
- Support processing of Streams of events.
- Support temporal constraints in order to model the temporal relationships between events.
- Support sliding windows of interesting events.
- Support a session scoped unified clock.
- Support the required volumes of events for CEP use cases.
- Support to (re)active rules.
- Support adapters for event input into the engine (pipeline).

(Fusion documentation)
Fusion – Temporal Reasoning

- “Drools implements all 13 operators defined by Allen and also their logical complement (negation). This section details each of the operators and their parameters.”
- After, before, coincides, during, finishes, finished by, includes, meets, met by, overlaps, overlapped by, starts, started by
Current CDS at Intermountain

- Foresight System
  - Rule triggering
  - Data normalization
  - Rule execution
  - Alert distribution
- abandoned BlazeDS due to cost, limitations
- currently Java based rules in Knowledge Repository
Drools – anticipated advantages

- Rule accessibility by clinicians – BPMN, decision tables, DSL; collaborative authoring
- Inference engine for certain types of problems
- Knowledge Sharing - common rule representation or translation
- Declarative programming
- BPMN represents an explicit, declarative *stateful* programming model
- Authoring by domain experts
- Tight integration of rules and workflow
Current Drools Projects at Intermountain

- Screening Framework (in production)
  - Bayesian based diagnostic rules
  - Drools for data acquisition, data filtering, data transformation, activation, alerting, acknowledgement
- Mothers to be survey rules
- Analytics Health Repository POC – L2 to L3
  - create cohorts for diabetes, depression, Charlson comorbidity index
- ICU Sepsis Protocol POC
  - Rivers Protocol for severe sepsis, septic shock
Drools – current directions

- Vet out BPM for long running protocols
  - Clinician accessibility, collaborative authoring
- Runtime model
  - Performance, scalability, resource util., high availability
- Rapid development model – UI integration with Human Task
- Integration of alternate inferencing models – Bayesian, etc
Intermountain/Drools – current directions

- Drools Knitting Group – collaboration with UofU, openCDS (Kamamoto), Socratic Grid (Fry)
- Intermountain Drools Framework
  - CDS as service – API (openCDS?)
  - Common Fact Model (CEM based?)
  - Common rule output (AlertEvent?)
  - Guvnor based repository
  - Datadrive, Timedrive and synchronous access
  - State management
  - UI association to Human Tasks
  - Protocol testing – autogen TestUI; data-driven unit tests
Drools Knitting Group
Analytics Health
Repository POC

4/15/2011
Analytics Health Repository (AHR)

- Sub group in Intermountain EDW team
- AHR L2 – scrubbed EDW data
- AHR L3 – derived cohorts such as episode of depression, diabetes
  - Currently using PL/SQL
- Drools Proof of Concept
  - Demonstrate that Drools can be used to produce L3 cohorts in an efficient, accessible, re-useable manner
  - Use of authoring tools by non-programmers
Problem

- We implemented the first AHR Problem-based depression rule (#3) using Drools. The goal of this rule is to group depression related Problems into “episodes”, where an episode is defined as a sequence of problems where no two consecutive problems are more than 12 months apart. A patient may have multiple episodes of depression.
Our approach was to use Java and JDBC to read all Problems from AHR-L2 for all patients using Java and JDBC. We then group the problems by patient and insert them into a Drools “KnowledgeSession”. The Drools rules then do the following:

- Identify all depression related problems and insert them into “Working Memory”
- Identify the oldest Depression Problem and add it to a new Episode in Working Memory
- Cycle through any other Depression Problems in chronological order and add to the current Episode if within 12 months
- Close the current Episode and create a new Episode if there are remaining Depression Problems

When the rules have created Episodes for all patients they are then written to a Morbidity table in a development database that is similar to the AHR-L3 Morbidity table.
Identify Depression Problems from all Problems and place in WorkingMemory.

Test problemType against global HashSet and test certainty against implicit list.

```java
rule "Identify Depression Problems"
    salience 100
    no-loop
when
    $p : Problem( problemType memberOf depressionTypes,
               certainty memberOf ( certaintyTypes ))
then
    insert(new DepressionProblem($p));
end
```
rule "Insert a new Episode with the oldest Depression Problem"

when

not Episode()
$episodes : Episodes()
$deProb : DepressionProblem( $problemDate : recordedDate )
not DepressionProblem( recordedDate < $problemDate )
then
Episode e = new Episode();
e.setInitDate($deProb.getRecordedDate());
e.setTermDate(addMonths($deProb.getRecordedDate(), 12));
e.setEmpi($deProb.getEmpi());
insert(e);
retract($deProb);
$episodes.getList().add(e);
end
rule "Update Episode with chained Problems"
when
  $e : Episode()$
  $deProb : DepressionProblem( $problemDate : recordedDate )$
  not DepressionProblem( recordedDate < $problemDate )
  eval((e.getTermDate().getTime()) >= (problemDate.getTime()))
then
  $e.setTermDate(addMonths($problemDate, 12));$
  retract($deProb);
rule "Remove Episode when no more chained Problems"
when
  $e : Episode()$
  $deProb : DepressionProblem( $problemDate : recordedDate )$
  not DepressionProblem( recordedDate < $problemDate )
  eval($e.getTermDate().getTime() < $problemDate.getTime())
then
  retract($e);
end
Outcome

- For the Drools implementation, the AHR-L2 problems were read from the EDW Oracle database. The Java code used to read and write the data, and Drools Rules were run from a developer’s laptop. The episode data was written to a MySql database deployed on a Windows 2003 Server.
- The AHR PL-Sql implementation runs entirely on the EDW Oracle database server.

<table>
<thead>
<tr>
<th></th>
<th>Rows Read</th>
<th>Rows Written¹</th>
<th>Total Time (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHR PL-Sql</td>
<td>74945</td>
<td>73,364</td>
<td>100</td>
</tr>
<tr>
<td>Drools</td>
<td>157,291²</td>
<td>73,364</td>
<td>178</td>
</tr>
</tbody>
</table>

1 - A row corresponds to one Episode of depression for a patient.
2 - The Drools implementation reads all Problem data in order to apply filter criteria in the rule rather than the query. This strategy will enable reuse of the Problem rule session for future applications.
Lessons Learned

- Minimize Java code outside of KnowledgeSession
  - insert all Problems into WM
- Issues related to a shared KnowledgeSession among multiple rule writers.
  - Facts should be immutable and preserved
  - Derived knowledge should be preserved and published
    - (Episodes and DepressionProblems)
Lessons Learned

- Use of fact inheritance, requires no-loop
- “oldest logic pattern” for Episodes
- Lack of ‘else’ construct, must use complementary rule
- All three of us had similar implementations