



LUNATECH

How to build a decision service using JBoss Rules (a.k.a. Drools)

Peter Hilton • 42 KTM • 17 March 2010

contents

- **Introductions - Peter, JBoss Rules and Drools Expert**
- **Decision Service architecture**
- **Business rules in JBoss Rules**
- **Space Invaders 🎮**



Peter Hilton - senior software developer

- Professional software developer since 1996
- Java web application developer at Lunatech since 2004
- Web application functional design and implementation
 - Play framework, JBoss Rules, Seam, Java EE...
- Technical/agile project management
 - Heavy user of JIRA and Confluence
- Implemented a JBoss Rules decision service in 2009
- Designed <http://plancruncher.com> last month





LUNATECH
RESEARCH

Language localisation in Java, JSF and Seam

2009-05-22/PeterHilton@www.lunatech.com

Meeting-avoidance for self-managing developers



Peter Hilton
Senior developer
Lunatech Research

LUNATECH
RESEARCH

www.devbox.com



LUNATECH
VENTURES



LUNATECH
VENTURES

The dirty secrets of Agile Software Development

Peter Hilton * 11 June 2009

Business plan symbology

Peter Hilton • 11 February 2010

Drools Expert

An inference-based business rules engine, also called a 'production rule system': a library that executes business rules

Drools Guvnor

Business Rules Management System, i.e. a multi-user environment for managing and versioning business rules with a web UI

Drools Fusion

A Complex Event Processing module for Drools Expert that allows rules to reason about 'business events' in real-time

Drools Flow

A work-flow engine, also called a process engine, that is integrated with the rules engine: manages steps in a process.

Drools Expert: what it is

- **Simply a Java library that you use in a Java application**
- **Minimum installation is 3 Drools JARs (3.1 MB) plus ANTLR, MVEL, Eclipse JDT, XStream (5.6 MB)**
- **Provides a Java API (all-in, 222 classes and interfaces)**
- **Drools Rule Language (DRL), which supports custom DSLs**
- **Can read rules from Excel Spreadsheets**
- **Eclipse tools available**



Drools Expert: what it is for

- **The rules engine provides a way to extract business rules from Java code**
- **Business rules are coded in a declarative rules language**
- **Rules are therefore not procedural or object-oriented code: they do not specify execution order**
- **More legible rules (when you are used to the syntax)**
- **Rules added to the code independently (in any order)**



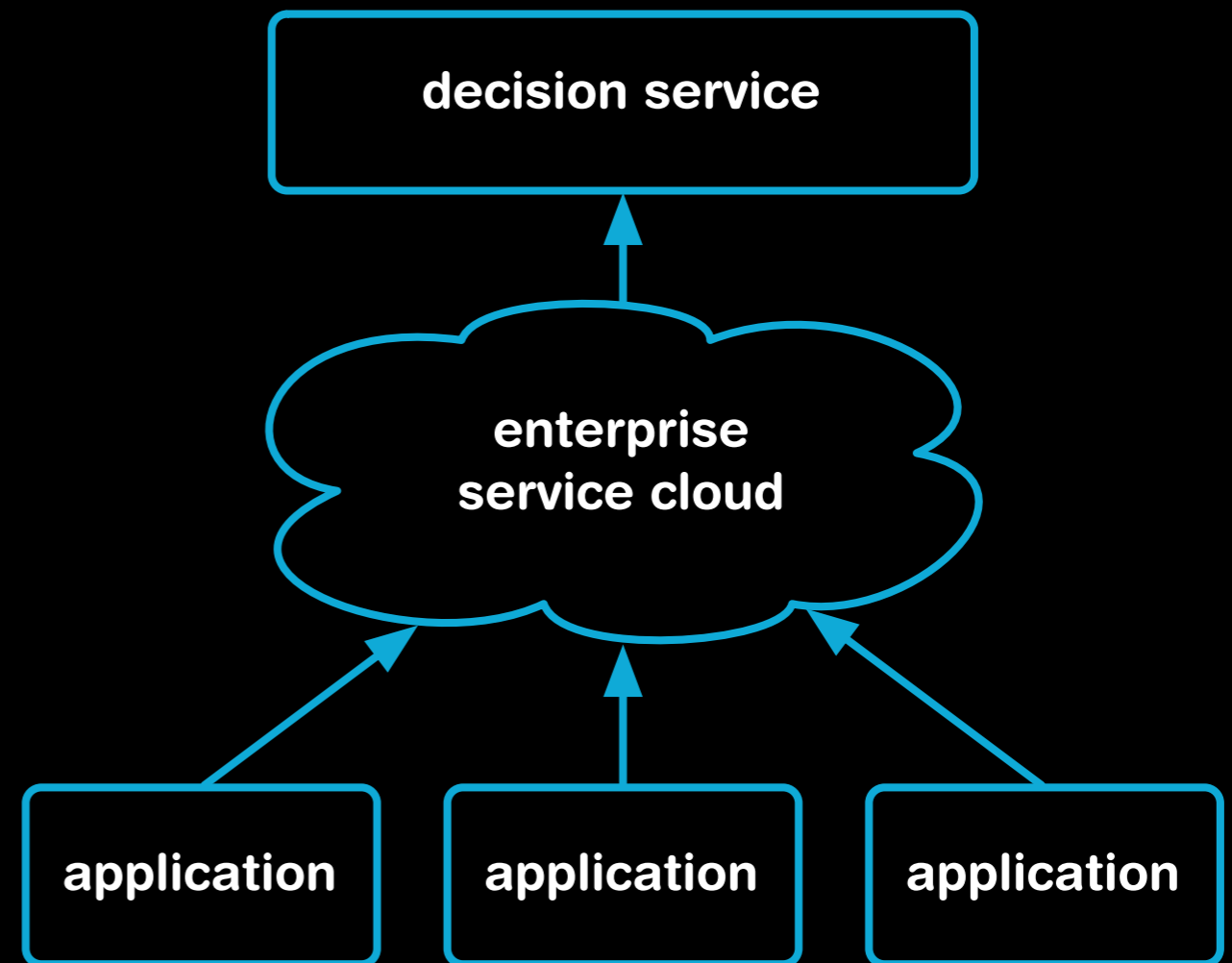
contents

- Introductions - Peter, JBoss Rules and Drools Expert
- **Decision Service architecture**
- Business rules in JBoss Rules
- Space Invaders 🚗



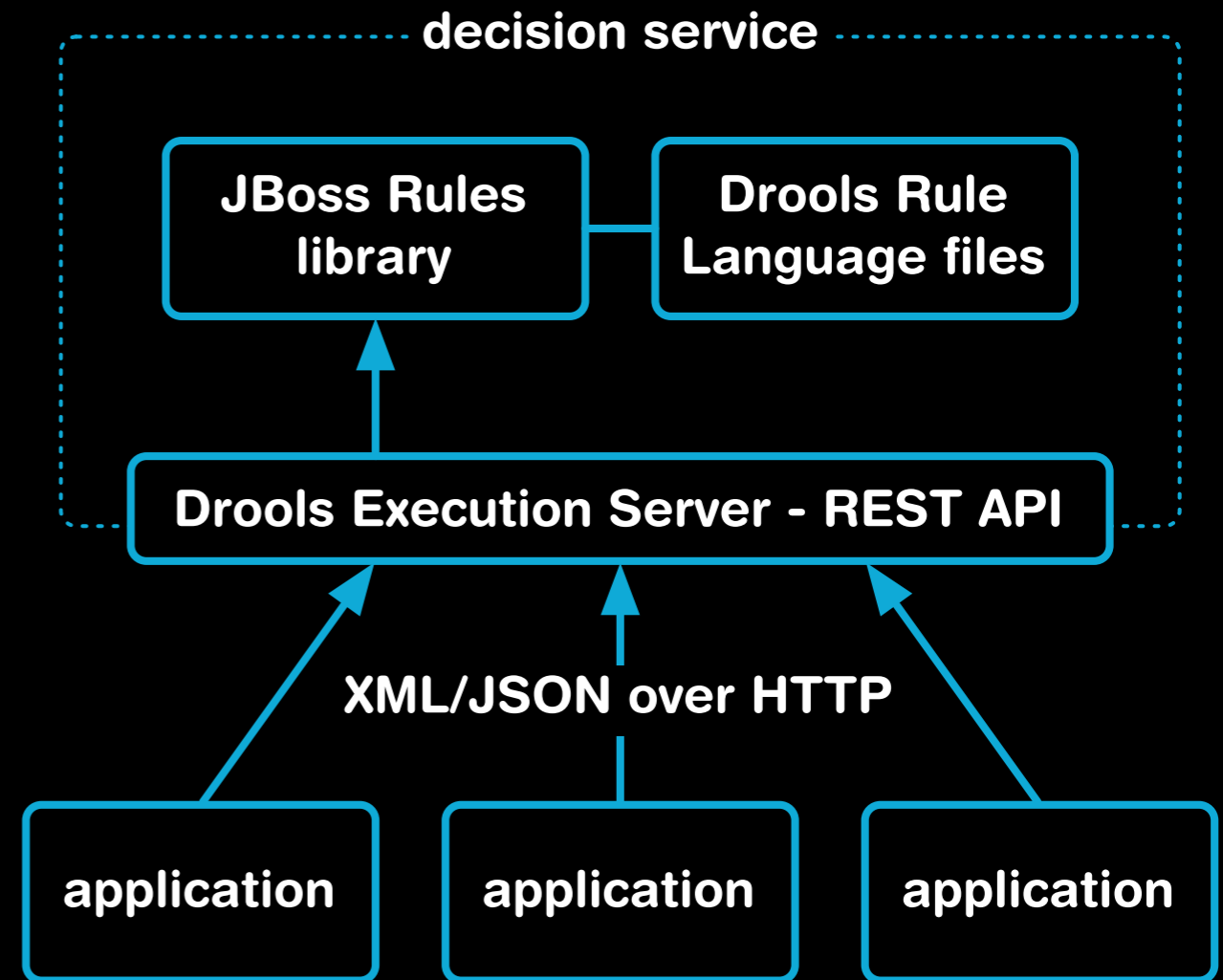
Decision services

- **A software component that is a business logic black box**
- **Typically used in a Service-Oriented Architecture**
- **Remote business logic instead of remote data**




Decision service architecture

- JBoss Rules includes an 'Execution Server' that runs out-of-the-box
- The Execution Server is a Servlet application WAR with a REST web services API (403 LOC)
- Could be used for an Ajax application's server-side logic



contents

- **Introductions - Peter, JBoss Rules and Drools Expert**
- **Decision Service architecture**
- **Business rules in JBoss Rules**
- **Space Invaders** 



Business rules example

- Desktop PC 'product configuration' application
- Decision service:
 - defines lists of available components
 - validates selections
 - validates combinations
 - generates messages



Uw Mac op maat.

Zet de puntjes op de i met behulp van de onderstaande opties.



Processor

Met twee Quad-Core Intel Xeon "Nehalem"-processoren beschikt u in de Mac Pro over twee processorkernen voor het summum aan kracht en prestaties. Kies uw

- Two 2.26GHz Quad-Core Intel Xeon [- € 2.340,00]
- Twee 2,66-GHz quad-core Intel Xeon-processors [- € 1.080,00]
- Twee 2,93-GHz quad-core Intel Xeon-processors



Geheugen

De 8-core Mac Pro ondersteunt tot 32 GB 1066-MHz DDR3 ECC SDRAM-geheugen. Kies meer geheugen om de algehele systeemprestaties te verbeteren.

- 6GB (6x1GB) [- € 3.330,00]
- 8GB (4x2GB) [- € 3.240,00]
- 12GB (6x2GB) [- € 3.060,00]
- 16GB (8x2GB) [- € 2.880,00]
- 32GB (8x4GB)



RAID-kaart

U kunt de opslagprestaties en gegevensbescherming verbeteren door de Mac Pro RAID-kaart en meerdere harde schijven. Opmerking: de Mac Pro RAID-kaart vereist als u voor SAS-schijven kiest.

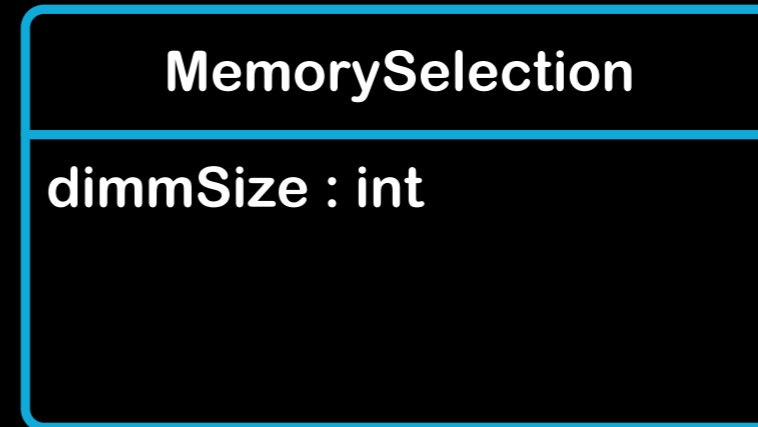
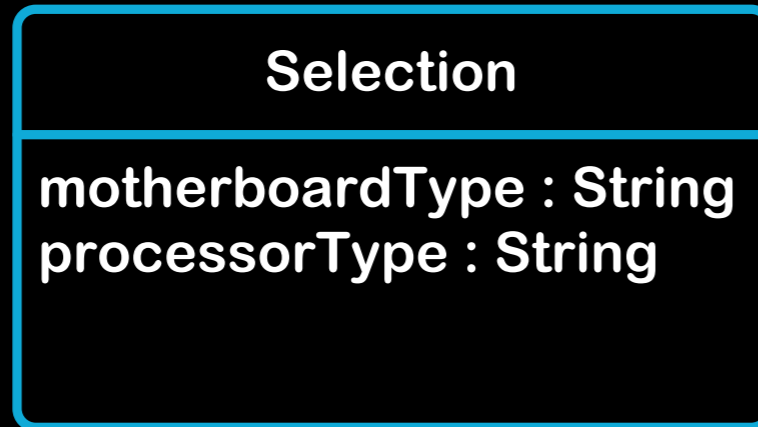
- Geen [- € 630,00]
- Mac Pro RAID-kaart



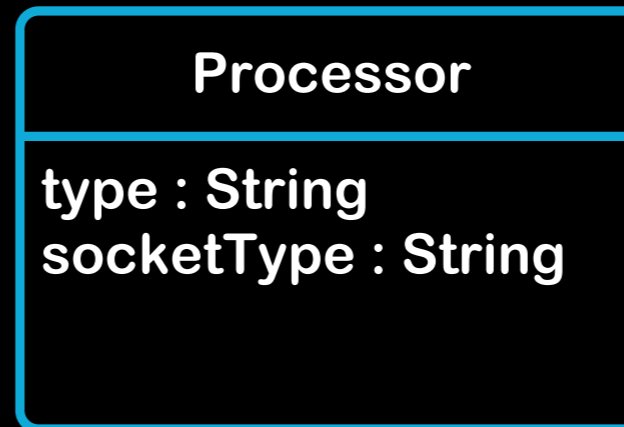
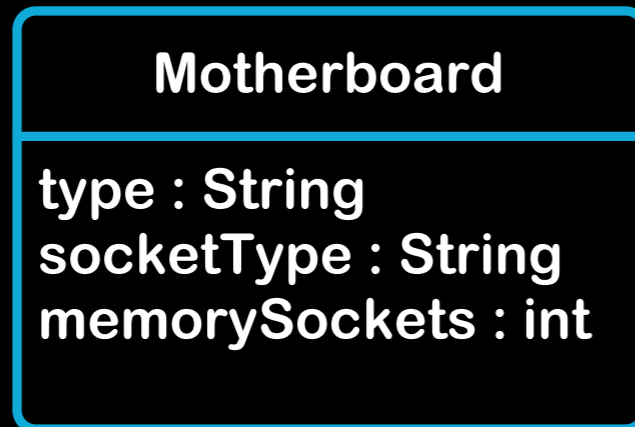
LUNATECH

Business rules example: data model

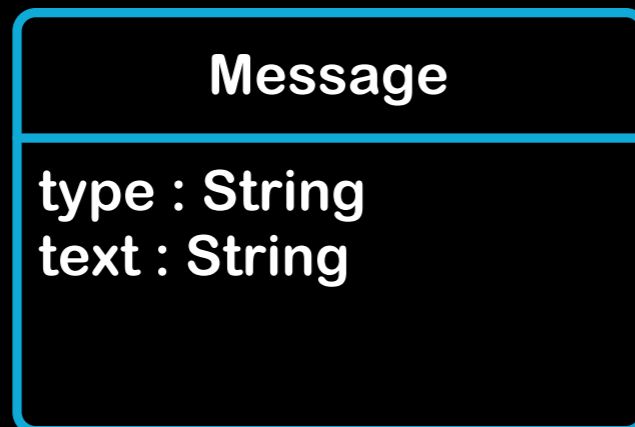
Input data:



Reference data:



Output data:



```
// Java code that runs a rules session and fetches results.

// Load and compile the config.drl rules file
KnowledgeBuilder builder = KnowledgeBuilderFactory.newKnowledgeBuilder();
Resource rules = ResourceFactory.newClassPathResource("config.drl", getClass());
builder.add(rules, ResourceType.DRL);
KnowledgeBase knowledgeBase = KnowledgeBaseFactory.newKnowledgeBase();
knowledgeBase.addKnowledgePackages(builder.getKnowledgePackages());

// Define the query
List<Command> commands = new ArrayList<Command>();
commands.add(CommandFactory.newInsert(customer));
commands.add(CommandFactory.newQuery("messages", "messages"));

// Execute the rules session, preceded by a batch of commands
Command batch = CommandFactory.newBatchExecution(commands);
ExecutionResults results = session.execute(batch);

// Fetch result messages from the query
QueryResults queryResults = (QueryResults) results.getValue("messages");
for (QueryResultsRow row : queryResults) {
    System.out.println(row.get("value"));
}
```

```
# Lists of available components...

# Activated for each Motherboard fact found in working memory
rule "Motherboard reference data loaded"
when
  # True when there is at least one Motherboard object in working memory
  exists Motherboard()
then
  System.out.println("Found a motherboard");
end
```



```
# Lists of available components... (continued)
```

```
# Activated for each Motherboard fact found in working memory
```

```
rule "Motherboard reference data loaded"
```

```
when
```

```
  # Bind the matched Motherboard object to the $motherboard variable
```

```
  $motherboard : Motherboard()
```

```
then
```

```
  System.out.println("Found motherboard: " + $motherboard);
```

```
end
```




```
# Lists of available components... (continued)

# Activated for each Motherboard fact found in working memory
rule "Motherboard reference data loaded"
when
  # Bind the matched Motherboard object to $motherboard
  $motherboard : Motherboard()
then
  System.out.println("Found motherboard: " + $motherboard);
end

# Generate a list:

# Query the contents of working memory for Motherboard facts
# the output data for use in a user-interface list.
query "motherboards"
  value : Motherboard()
end
```



```
# Lists of available components... (continued)
```

```
# Rule that inserts reference data into working memory
```

```
rule "Insert motherboards"
```

```
when
```

```
  # True when there are no Motherboard objects in working memory
```

```
  not Motherboard()
```

```
then
```

```
  # Insert a new Motherboard object into working memory
```

```
  Motherboard integrated = new Motherboard();
```

```
  integrated.setType("integrated");
```

```
  integrated.setSocketType("none");
```

```
  integrated.setMemorySockets(0);
```

```
  insert(integrated);
```

```
  Motherboard standard = new Motherboard();
```

```
  standard.setType("standard");
```

```
  standard.setSocketType("pga");
```

```
  standard.setMemorySockets(2);
```

```
  insert(standard);
```

```
end
```



```
# Result messages...
```

```
# Declare a JavaBean type in-line for use in rules
```

```
declare Message
```

```
  type : String
```

```
  text : String
```

```
end
```

```
# Inserts a new message "Found first motherboard" when there is a Motherboard  
# fact in working memory. This only happens once, because the left-hand side  
# also checks that the message itself is not yet in working memory.
```

```
rule "First motherboard reference data loaded"
```

```
when
```

```
  Motherboard()
```

```
  not Message(text == "Found first motherboard")
```

```
then
```

```
  # Inserts a Message fact
```

```
  Message message = new Message();
```

```
  message.setType("DEBUG");
```

```
  message.setText("Found first motherboard");
```

```
  insert(message);
```

```
end
```

```
# Defining functions
```

```
# Since the Message type only has a default constructor, it is somewhat verbose to  
# insert the message; it is more convenient to define a function in the rules file:
```

```
function void insertDebugMessage(KnowledgeHelper drools, String text) {  
    Message message = new Message();  
    message.setType("DEBUG");  
    message.setText(text);  
    drools.insert(message);  
}
```

```
rule "First motherboard reference data loaded"
```

```
when  
    Motherboard()  
    not Message(text == "Found first motherboard")
```

```
then  
    # Use the function defined above  
    insertDebugMessage(drools, "Found first motherboard");
```

```
end
```

```
# Fetch validation result messages using a query that excludes DEBUG messages
```

```
query "messages"  
    value : Message(type == "RESULT")
```

```
end
```

```
# Validating user selections...

# Check that the selection is not empty
# Reminder: the Selection JavaBean has motherboardType and processorType properties
rule "No motherboard selected"
when
  # True when there is a Selection whose motherboardType property is null
  Selection(motherboardType == null)
then
  insertMessage(drools, "No motherboard selected");
end
```



```
# Validating user selections... (continued)

# Check that the selection's motherboardType matches the type property
# value of an available motherboard
rule "Selected motherboard type does not exist"
when
  # True when the motherboardType property is not null; bind the value to $type
  Selection($type : motherboardType != null)

  # True when there is no Motherboard whose type property value is $type
  not Motherboard(type == $type)
then
  insertMessage(drools, "Selected motherboard type does not exist");
end
```



```
# Filtering available components...
```

```
# Remove each motherboard that does not match the processor type selection
```

```
rule "Filter motherboards for selected processor socket type"
```

```
when
```

```
  # True when the selection specifies a processor type
```

```
  Selection($processor : processorType != null)
```

```
  # True when there is a processor whose type is $processor, the selected type;
```

```
  # bind this processor's socket type to $socket
```

```
  Processor(type == $processor, $socket : socketType)
```

```
  # True when there is Motherboard whose socket type is not $socket, the selected
```

```
  # processor's socket type
```

```
  $motherboard : Motherboard(socketType != $socket)
```

```
then
```

```
  retract($motherboard);
```

```
end
```



```
# Ad-hoc validations...
```

```
rule "Memory must be selected in matching pairs"
```

```
when
```

```
# True when there is a memory selection, with memory size $selectedDimmSize  
MemorySelection($selectedDimmSize : dimmSize)
```

```
# True for a list from MemorySelection facts of the specified size  
# (only matches facts in the collect's pattern, not all of working memory);  
# bind the size of the resulting list to $quantitySelected
```

```
ArrayList($quantitySelected : size)
```

```
  from collect( MemorySelection(dimmSize == $selectedDimmSize) )
```

```
# True when $quantitySelected is odd, checked by evaluating a Java expression
```

```
eval($quantitySelected % 2 != 0)
```

```
then
```

```
  insertMessage(drools, "Odd number of " + $selectedDimmSize + "GB DIMMs selected");
```

```
end
```



contents

- **Introductions - Peter, JBoss Rules and Drools Expert**
- **Decision Service architecture**
- **Business rules in JBoss Rules**
- **Space Invaders 🎮**



Space Invaders Enterprise Edition



Space Invaders Enterprise Edition

- **Even Space Invaders has 'business logic' that can be more clearly expressed in rules language**
- **This implementation extracts the original logic into a JBoss Rules implementation**
- **Declarative rules avoid mixing this logic with a real-time event loop**



```
rule "Reverse aliens if one reaches the edge of the screen"
```

```
when
```

```
  $alien : AlienEntity()
```

```
  exists (AlienEntity(x < 10) or AlienEntity(x > 750))
```

```
then
```

```
  $alien.setHorizMovement(-$alien.getHorizMovement());
```

```
  $alien.setY($alien.getY() + 10);
```

```
end
```

```
rule "Process bullets hitting aliens"
```

```
when
```

```
  $shot : ShotEntity()
```

```
  $alien : AlienEntity(this != $shot, eval($shot.collidesWith($alien)))
```

```
  $otherAlien : AlienEntity()
```

```
then
```

```
  game.getEntities().remove($shot);
```

```
  game.getEntities().remove($alien);
```

```
  $otherAlien.setHorizMovement($otherAlien.getHorizMovement() * 1.04);
```

```
end
```



```
rule "End the game when all aliens are killed"  
  salience 1  
  when  
    not AlienEntity()  
    exists ShipEntity()  
  then  
    game.notifyWin();  
    game.getEntities().clear();  
end
```

```
rule "End the game when an alien reaches the bottom"  
  when  
    exists AlienEntity(y > 570)  
  then  
    game.notifyDeath();  
    game.getEntities().clear();  
end
```





@PeterHilton

peter.hilton@lunatech.com

www.lunatech-research.com

