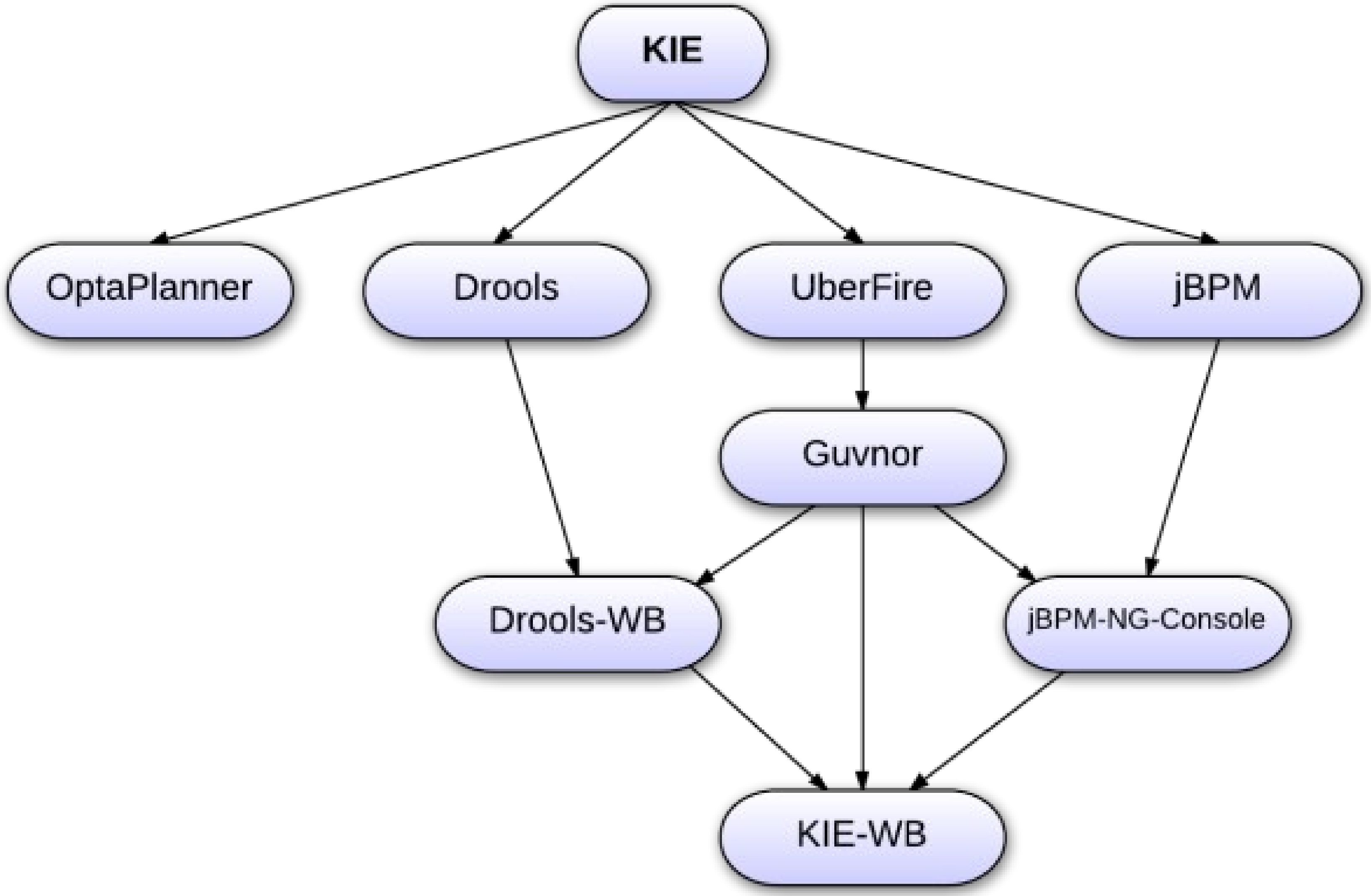
Introducing

Drools Business Logic integration Platform/

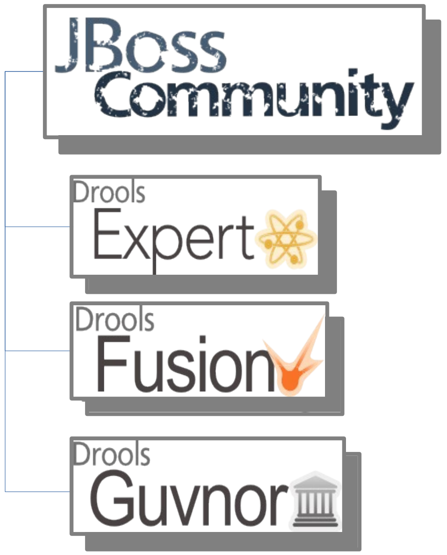
y Mario Fusco
Red Hat – Senior Software Enginee
mfusco@redhat.com

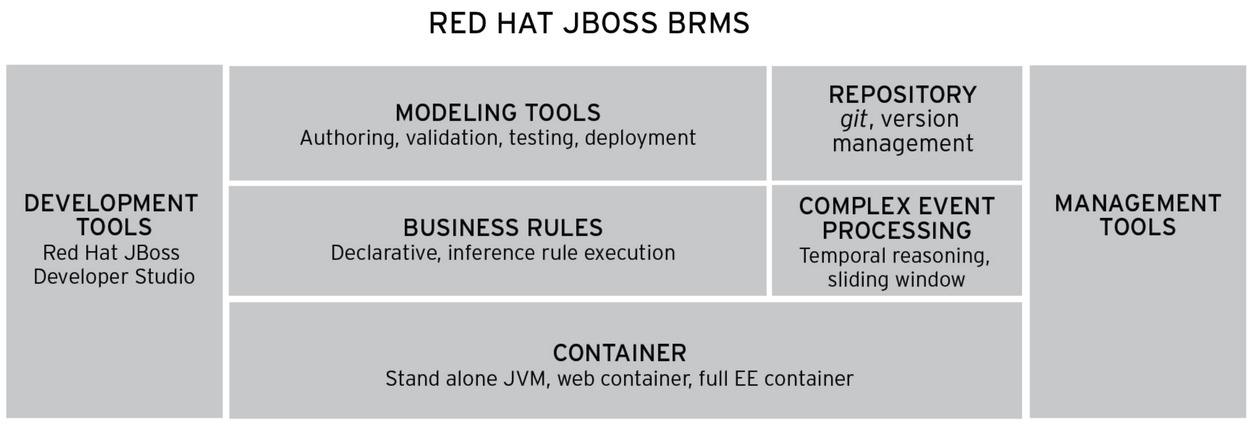
KIE – Knowledge Is Everything



RED HAT® JBOSS® BRMS

A single, integrated, certified distribution for Business Rules Management and Complex Event Processing, based on open source community projects:




RED HAT JBOSS BRMS
DEVELOPMENT TOOLS
Red Hat JBoss Developer Studio
MODELING TOOLS
Authoring, validation, testing, deployment
REPOSITORY
git, version management
BUSINESS RULES
Declarative, inference rule execution
COMPLEX EVENT PROCESSING
Temporal reasoning, sliding window
CONTAINER
Stand alone JVM, web container, full EE container
MANAGEMENT TOOLS

What a rule-based program is

- A rule-based program is made up of **discrete rules**, each of which applies to some subset of the problem
- It is **simpler**, because you can concentrate on the rules for one situation at a time
- It can be more **flexible** in the face of fragmentary or poorly conditioned inputs
- Used for problems involving control, diagnosis, prediction, classification, pattern recognition … in short, all problems without clear algorithmic solutions

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

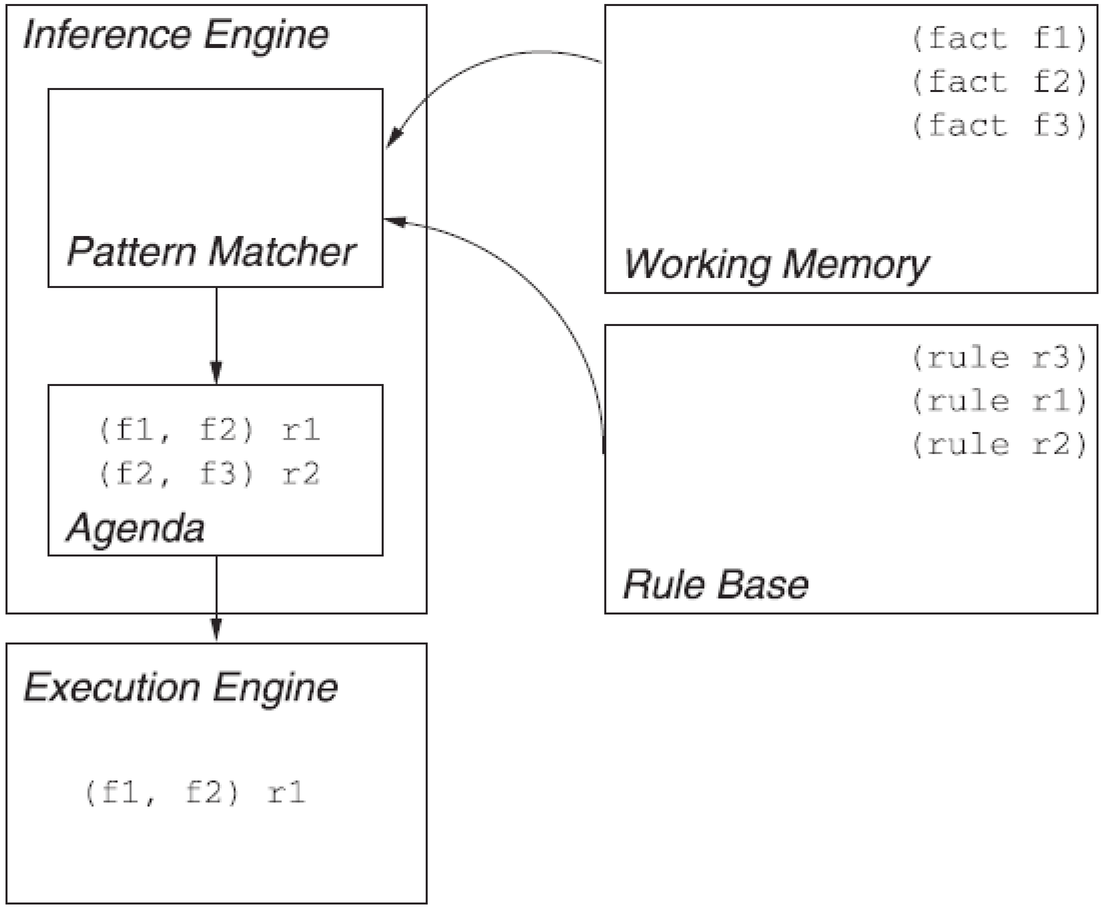
Advantages of Declarative Programming

- **Easier to understand →** It is more likely for a technically skilled business analyst to verify, validate or even change a rule than a piece of Java code
- **Improved maintainability →** We don't care about **how** to implement a solution only **what** needs to be done to solve a problem
- **Deals with evolving complexity →** It's easier to modify a rule than a Java program and to determine the impact of this change on the rest of the application
- **Modularity** → Each rule models an isolated and small portion of your business logic and is not part of a monolithic program
- **Requirements can be more naturally translated into rules**
- **Clear separation of business logic from the rest of the system**

When should you use a Rule Engine?

- The problem is beyond any obvious algorithmic solution or it isn't fully understood
- The logic changes often
- Domain experts (or business analysts) are readily available, but are nontechnical
- You want to isolate the key parts of your business logic, *especially the really messy parts*

How a rule-based system works



Rule's anatomy

Quotes on Rule names are optional if the rule name has no spaces

```
rule "<name>"
   <attribute> <value>
   when
     <LHS>
   then
     <RHS>
end
```

Pattern-matching against objects in the Working Memory

salience <int>
agenda-group <string>
no-loop <boolean>
auto-focus <boolean>
duration <long>
....

Code executed when a match is found

Imperative vs Declarative

A method must be called directly

Specific passing of arguments

```
public void helloMark(Person person) {
   if ( person.getName().equals( “mark” ) {
      System.out.println( “Hello Mark” );
   }
}
```

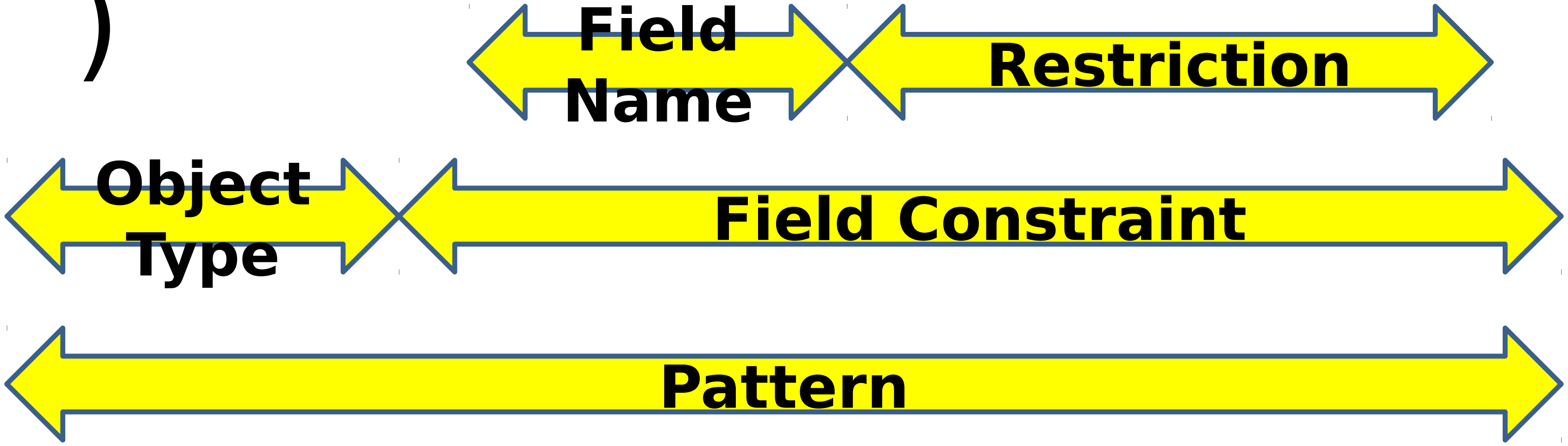
Rules can never be called directly

Specific instances cannot be passed but are automatically selected with pattern-matching

```
rule “Hello Mark”
   when
      Person( name == “mark” )
   then
      System.out.println( “Hello Mark” );
end
```

What is a pattern

```
Person( name == “mark” )
```



Rule's definition

```
// Java
public class Applicant {
    private String name;
    private int age;
    private boolean valid;
    // getter and setter
  here
}
```

```
// DRL
declare Applicant
    name : String
    age : int
    valid : boolean
end
```

```
rule "Is of valid age" when
    $a : Applicant( age >= 18 )
then
    modify( $a ) { valid =
true };
end
```

More Pattern Examples

```
Person( $age : age )
Person( age == ( $age + 1 ) )

Person(age > 30 && < 40 || hair in (“black”, “brown”) )

Person(pets contain $rover )

Person(pets[’rover’].type == “dog”)
```

Conditional Elements

```
not Bus( color = “red” )
exists Bus( color = “red” )
forall ( $bus : Bus( color == “red” ) )

$owner : Person( name == “mark” )
         Pet( name == “rover” ) from $owner.pets

$zipCode : ZipCode()
Person( ) from $hbn.getNamedQuery(“Find People”)
                   .setParameters( [ “zipCode” : $zipCode
] )
                   .list()
```

Hibernate session

'from' can work on any expression

Complex Event Processing

Event

A record of state change in the application domain at a particular point in time

Complex Event

An abstraction of other events called its members

Complex Event Processing

Processing multiple events with the goal of identifying
the meaningful events within the event cloud

Drools CEP

- Drools modules for Complex Event Processing
- Understand and handle events as a first class platform citizen (actually special type of Fact)
- Select a set of interesting events in a **cloud** or **stream** of events
- Detect the relevant relationship (patterns) among these events
- Take appropriate actions based on the patterns detected

Cloud vs. Stream Mode

Cloud Mode (default)

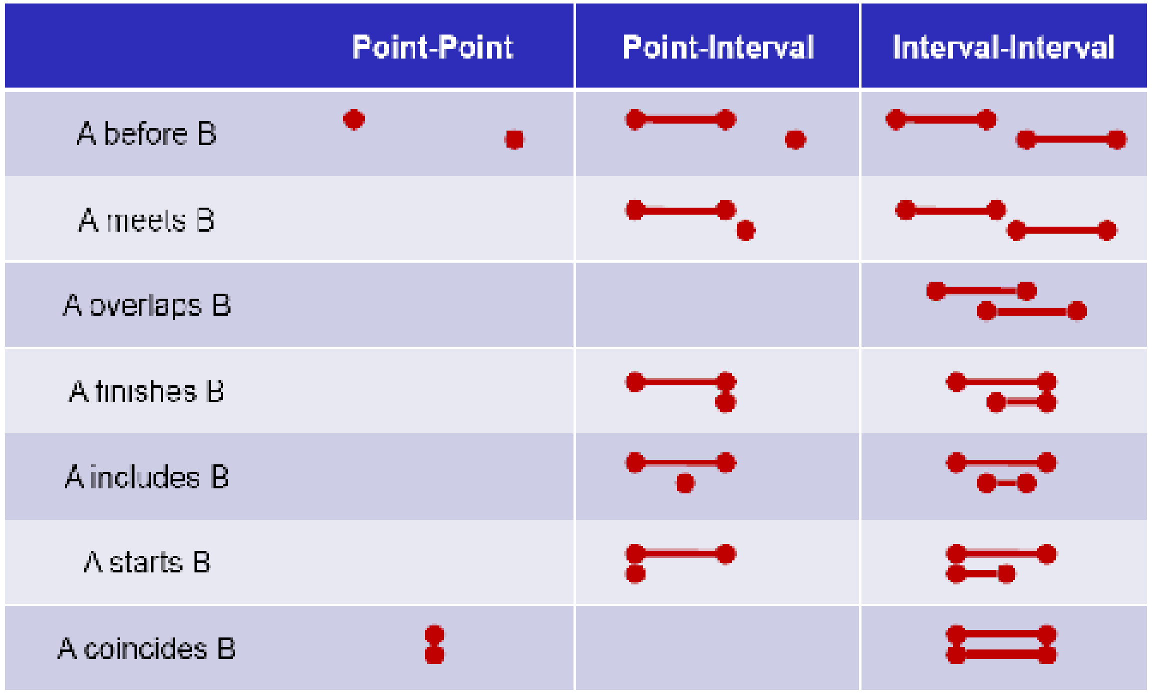
- **No notion of time**
- No requirement on event ordering
- Since they are based on the concept of “now” it is not possible to use sliding windows
- Not possible to determine when events can no longer match, so the application must **explicitly retract events** when they are no longer necessary

Stream Mode

- Events in each stream must be **time-ordered**
- The engine will force synchronization between streams through the use of the session clock
- **Sliding Window** support
- Automatic Event Lifecycle Management
- Automatic Rule Delaying when using Negative Patterns

Events as Facts in Time

Temporal relationships between events

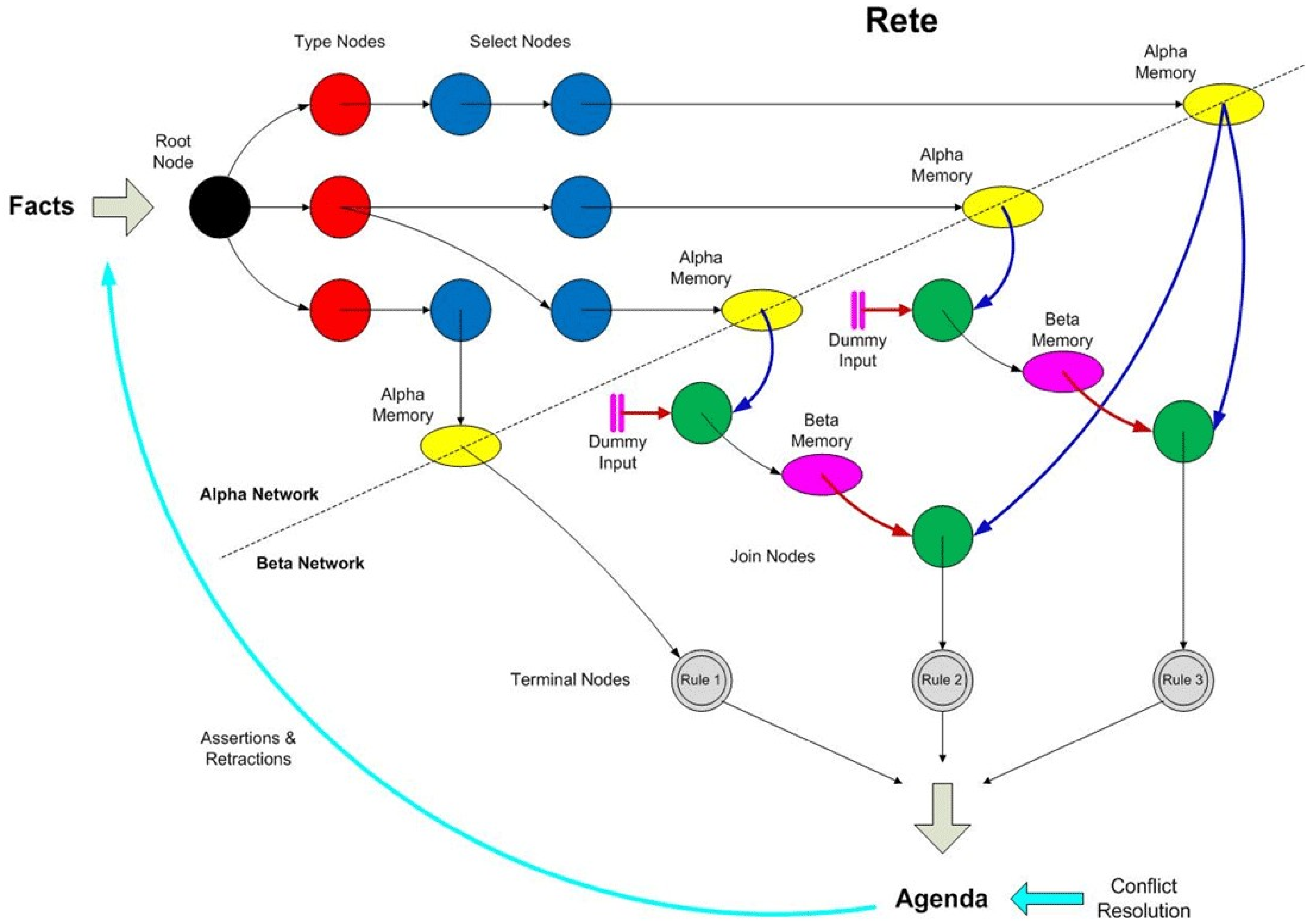


```
rule
    "Sound the alarm"
when
    $f : FireDetected( )
    not( SprinklerActivated( this after[0s,10s] $f ) )
then
    // sound the alarm
end
```

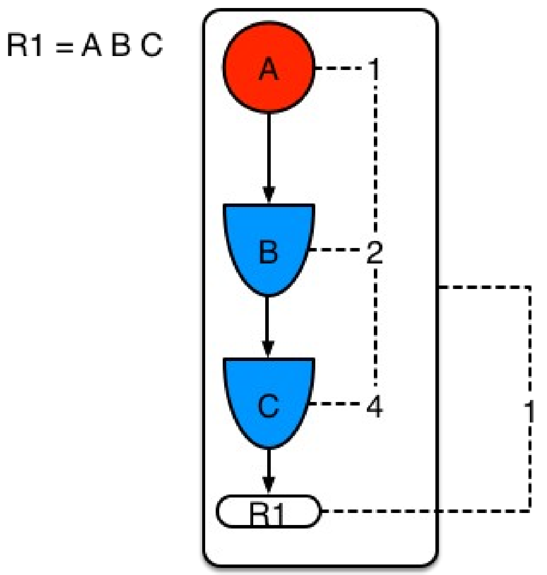
Innovations in Drools 6

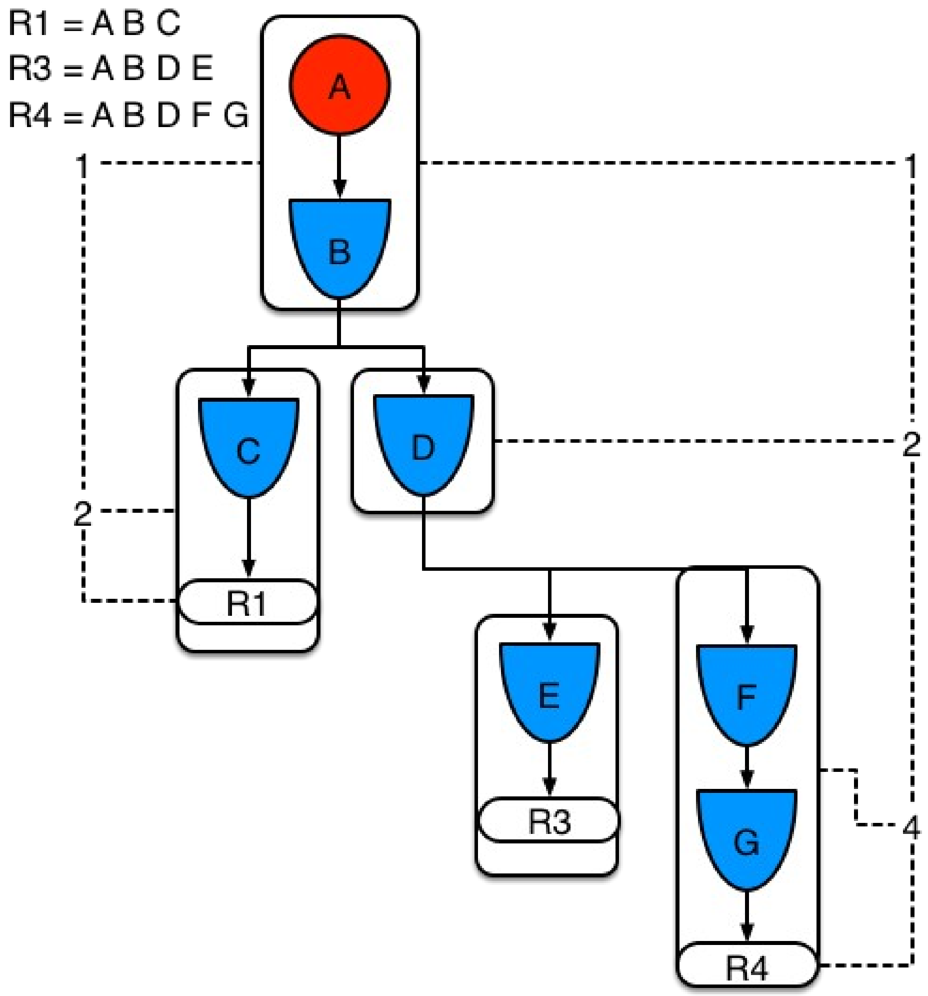
- A brand new engine: from ReteOO to **Phreak**
- From tuple based to set based propagation
- A git based repository …
- … combined with a maven based deployment model
- A simplified and mostly declarative API

From ReteOO ...

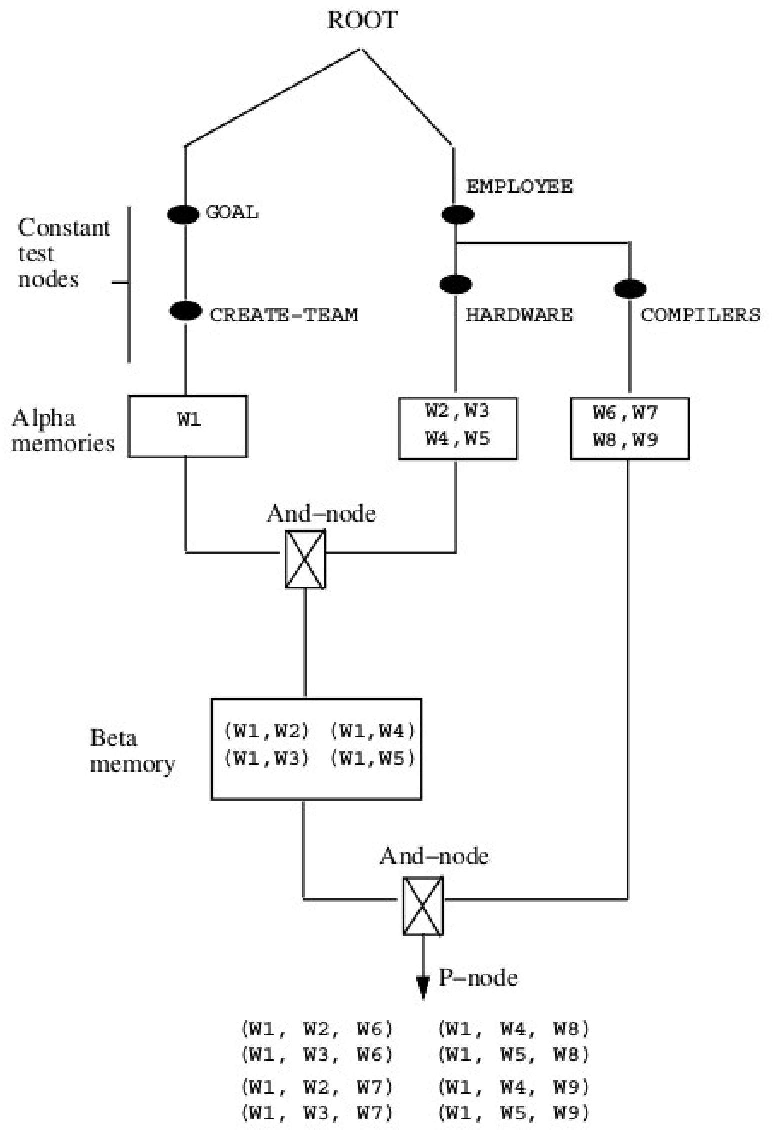

Rete
Type Nodes
Select Nodes
Root Node
Facts
Alpha Memory
Alpha Memory
Alpha Memory
Alpha Memory
Dummy Input
Beta Memory
Dummy Input
Beta Memory
Alpha Network
Beta Network
Join Nodes
Terminal Nodes
Rule 1
Rule 2
Rule 3
Assertions & Retractions
Agenda
Conflict Resolution

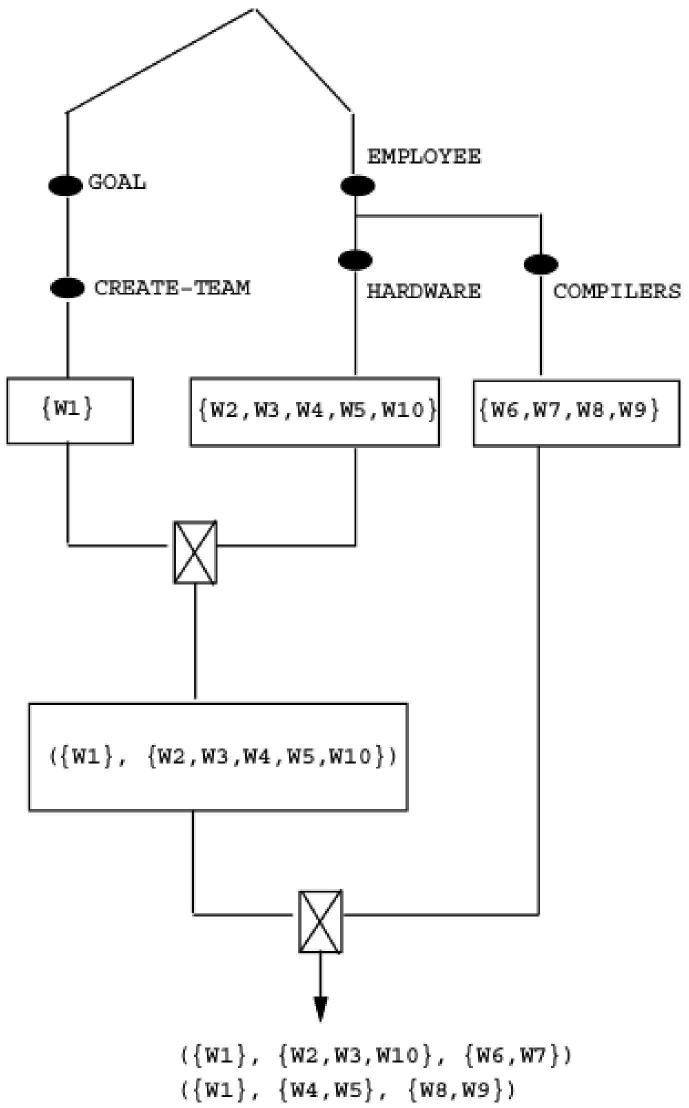
... to Phreak


R1 = A B C
A
B
C
R1
1
2
4
1


R1 = A B C
R3 = A B D E
R4 = A B D F G
A
B
C
D
R1
E
F
G
R3
R4
1
1
2
2
4

From tuple based to set based propagation





Advantages

- Preserves all ReteOO optimizations combining them with pros of other well known algorithms like Leaps, Collection Oriented Match, L/R Unlinking ...
- On average 20% faster then ReteOO (and up to 400% faster on specific use cases)
- Reduced memory footprint
- More forgiving in presence of badly written rules

Keep innovating

Extending an Object-Oriented RETE Network with Fine-Grained Reactivity to Property Modifications

Mark Proctor[1,2], Mario Fusco[2], and Davide Sottara[3]

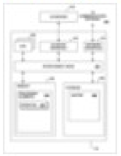
Dept. of Electrical & Electronic Engineering, Imperial College London, Londo
m.proctor13@imperial.ac.uk
JBoss, a Division of Red Hat Inc.
mfusco@redhat.com
Biomedical Informatics Dept., Arizona State University, Scottsdale (AZ)
davide.sottara@asu.edu

Building a Hybrid Reactive Rule Engine for Relational and Graph Reasoning

Mario Fusco[1](✉), Davide Sottara[2](✉), István Ráth[3], and Mark Proctor[1,4]

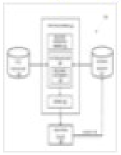
[1] A Division of Red Hat Inc., JBoss, Milan, Italy
mfusco@redhat.com
http://www.jboss.org
[2] Department of Biomedical Informatics, Arizona State University, Tempe, AZ, USA
davide.sottara@asu.edu
[3] Department of Measurement and Information Systems, Budapest University of Technology and Economics, Budapest, Hungary
rath@mit.bme.hu
[4] Department of Electrical and Electronic Engineering, Imperial College London, London, UK
m.proctor13@imperial.ac.uk

Compile-time grouping of tuples in a streaming application



www.google.it/patents/US20140095506
App. - Filed 21 Feb 2013 - Published 3 Apr 2014 - Michael J. Branson - International Business Machines Corporation
... Feb 17, 2011, **Mark Proctor**, Pattern behavior support in a **rule** engine ... 2014, **Red Hat**, Inc. Systems and Methods for Efficient Just-In-Time Compilation ...
Overview - Related - Discuss

Property reactive modifications in a rete network

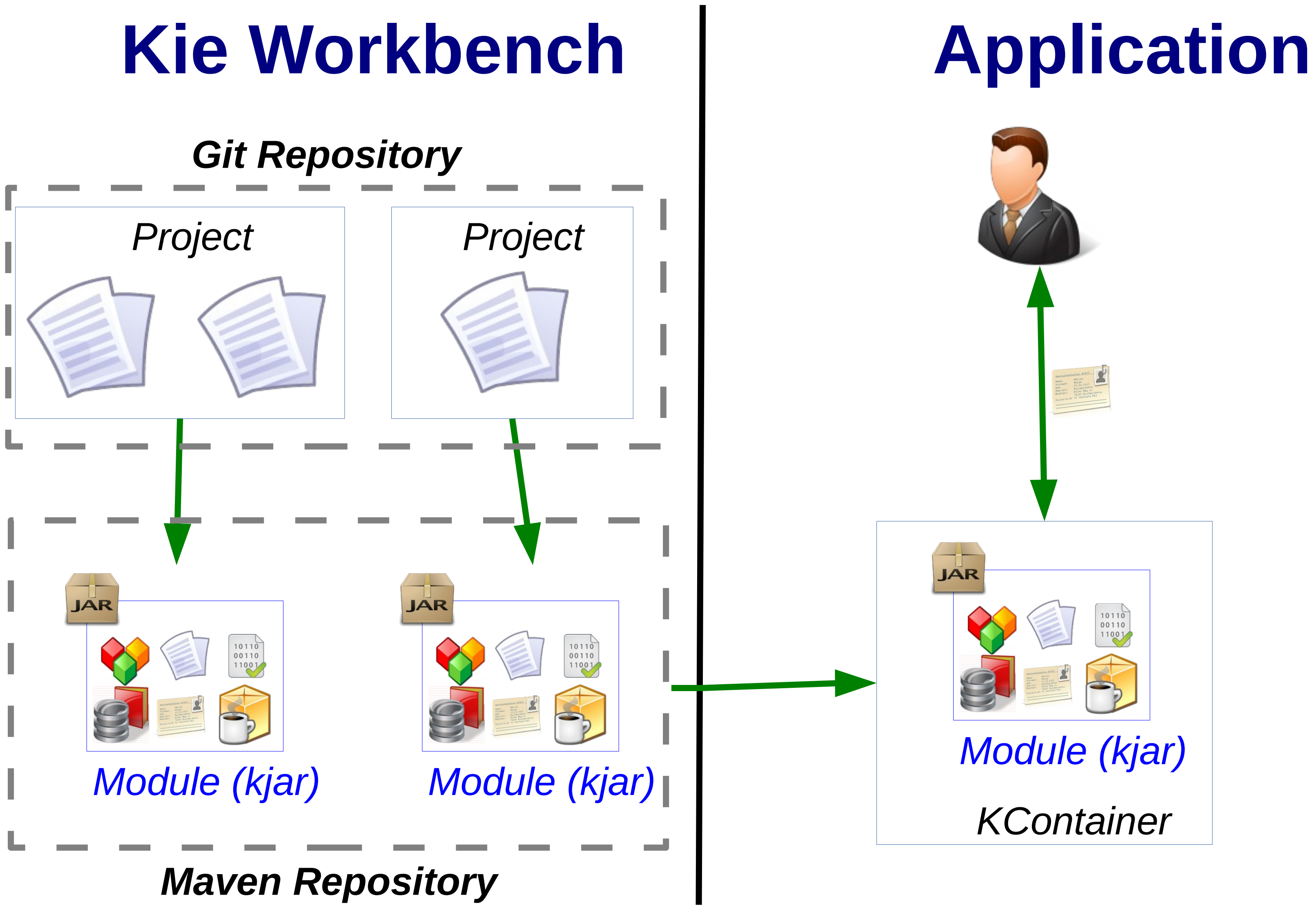


www.google.it/patents/US20140201124
App. - Filed 11 Jan 2013 - Published 17 Jul 2014 - Ma
Proctor - Red Hat, Inc.
A processing device executing a Rete **rule** engine mo
a particular property of an object ... Inventors, **Mark Pr**
Mario Fusco. Original Assignee, **Red Hat** ...
Overview - Related - Discuss

Lazily enabled truth maintenance in rule engines

www.google.it/patents/US8538905
Grant - Filed 2 Dec 2010 - Issued 17 Sep 2013 - Mark Proctor - Red Hat, Inc.
Some embodiments of a method to lazily enable truth maintenance in a **rule** engine have been presented. ... 2007, Dec 4, 2008, **Mark Proctor**, Method and apparatus to define a ruleflow ... Owner name: **RED HAT**, INC., NORTH CAROLINA ...
Overview - Related - Discuss

A git/maven based workbench



Defining Kbases and KSessions

```xml
<kmodule xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://jboss.org/kie/6.0.0/kmodule">

    <kbase name="ServerKB" packages="org.myproject.example.server,
                                      org.myproject.example.server.model"
          eventProcessingMode="stream" equalsBehavior="identity">
        <ksession name="ServerKS" default="true" />
    </kbase>

    <kbase name="ClientKB" packages="org.myproject.example.client">
        <ksession name="StatefulClientKS" type="stateful"/>
        <ksession name="StatelessClientKS" type="stateless"/>
    </kbase>
</kmodule>
```

```java
KieContainer kc = KieServices.Factory.get().getKieClasspathContainer();
KieSession serverKsession = kc.newKieSession( "ServerKS" );
KieSession clientKsession = kc.newKieSession( "StatelessClientKS" );
```

Loading a kjar from maven

```
<dependency>
    <groupId>org.mycompany</groupId>
    <artifactId>myproject</artifactId>
    <version>1.0.0</version>
</dependency>
```

```
KieServices ks = KieServices.Factory.get();
KieContainer kContainer =
    ks.newKieContainer(ks.newReleaseId("org.mycompany",
                                              "myproject",
                                              "1.0.0"));
KieSession kSession = kContainer.newKieSession("ksession1");
```