Extending General Purpose Engines with Domain Specific Resources

Edson Tirelli
etirelli@redhat.com
Senior Software Engineer
JBoss, a Division of Red Hat

Agenda

- **Presentation**
- Solvers and enablers
- Abstraction as an enabling tool
- Domain Specific Resources
  - Functions
  - Set-based Functions (Accumulate Functions)
  - Domain Specific Evaluators
  - Domain Specific Languages
  - Domain Specific Processes (Work Items)
- Questions & Answers

Presentation

- **The spirit:**
  - “No fluff, just stuff!”
  - “Open source, open minds!”
- **The goal:**
  - Show actual code
    - Most of the features are available in high level tooling
  - Show what + why + how
  - Approach bottom → up

Who am I? Who are you?

- **Who do you want to be when you grow up?**
  - A solver?
    - People brings you problems, you solve them.
  - An enabler?
    - People brings you problems, you enable them to solve the problems.
- **The role of Rules Engines:**
  - The tool to solve the problems?
  - The tool to empower your users to solve the problem?

This talk is...

- **About enabling others to solve their problems**
  - Do more by doing less

- **For instance:**
  - Drools is First Order Logic complete
  - BUT, the “user” just wants a way to find which aircrafts are scheduled to be at the same gate at the same time!
  - The user does not care about means, he cares about ends

The need for abstractions

"Any problem in computer science can be solved with another layer of indirection.

But that usually will create another problem."

-- **David Wheeler**

Evolution of...

- **Languages:**
  - 1st generation: machine language
    - 2nd generation: assembly language
      - 3rd generation: C, COBOL, ...
        - 4th generation: ...
- **Paradigms:**
  - Imperative
    - Functional
      - Object Oriented
        - Object-Functional
          - ...

Use of “frameworks”

- **Standards:**
  - Communication
  - Data management
  - …
- **Tooling:**
  - BRMS’s
  - BPMS’s
  - DBMS’s
  - …

Abstractions

- **The use of higher abstractions:**
  - Brings the problem closer to the human “way of thinking”
  - Facilitate the understanding of the whole (big picture)
  - De-emphasize the focus on details
  - Promotes declarative approaches
- **Enables:**
  - Awareness
- **Improves:**
  - Agility
  - Maintainability
  - Correctness

Domain Specific Resources

- **Domain Specific Resources**
  - Facilitate the creation of solutions specific to a given domain
  - Improve the perceived expressiveness of general purpose tools

- **Examples:**
  - Domain Specific Languages
  - Function libraries

Drools Vision

“A common platform to model and govern the business logic of the enterprise.”

Drools is...

- **Designed for extensibility**
  - Function libraries (duh!)
  - Set-based functions (Accumulate Functions)
  - Domain Specific Evaluators
  - Domain Specific Languages
  - Domain Specific Processes (Work Items)

Function Libraries

- **Common resource found in any programming language and most tools:**
  - Implemented inline:

```
function double rate( double original, double current ) {
    return current/original;
}
```

  - Imported:

```
import function org.drools.libs.SomeClass.rate;
```

Function Libraries

- **Using Functions:**

```
rule “Actual rate is above goal”
when
    Investment( goalRate < ( rate( original, current ) ) )
then
    // congratulate the manager
end
```

- **Other uses:**

```
eval( rate( $original, $current ) > $goal )

Number( this > $goal ) from rate( $original, $current )
```

Accumulate (set functions)

- **In Drools it is responsible for set operations**
  - Similar to Jess “accumulate” and JRules “collect”
- **Accepts custom code and accumulate functions:**
  - Custom code: bad ☹
  - Accumulate functions: good ☺

```
rule “Accumulate Function usage example”
when
    Order( $number : number )
    $orderTotal : Number( ) from accumulate(
                $item : Item( orderNumber == $number ),
                sum( $item.value ) )
then
    // order total is $orderTotal
end
```

Accumulate Functions

- **Operate on sets of values, instead of named parameters**
- **Plug into the “accumulate” CE**
- **Example:**
  - Requirement: calculate the taxes for a set of items, where different product categories have different tax rates.

```
rule “Accumulate Function usage example”
when
    Order( $number : number )
    $taxes : Number( ) from accumulate(
                $item : Item( orderNumber == $number ),
                sumTaxes( $item ) )
then
    // total taxes == $taxes
end
```

Accumulate Functions

- **To create: implement the AccumulateFunction interface**

```
public interface AccumulateFunction extends Externalizable {
    public Serializable createContext();
    public void init(Serializable context);
    public void accumulate(Serializable context, Object value);
    public void reverse(Serializable context, Object value);
    public Object getResult(Serializable context);
    public boolean supportsReverse();
}
```

- **To register: either call the API or use a configuration file**

```
KnowledgeBuilderConfiguration conf = ...
conf.setOption(AccumulateFunctionOption.get( “sumTaxes",
                        new SumTaxesAccumulateFunction() ) );
```

```
drools.accumulate.function.sumTaxes = org.foo.SumTaxesAccumulateFunction
```

Evaluators

- **Evaluators define relationships between two facts**
  - make constraints easy to express
  - hide implementations details like ref data access
- **Example:**
  - **isWithin**: evaluator that relates a GPS position to the corresponding geo-political city.

```
rule “Speed limit warn”
when
    $city : City( $maxSpeed : maxSpeed )
    $vcle : VehicleStatus( gpsPosition isWithin $city,
                           currentSpeed > $maxSpeed )
then
    // send a warn to the driver about speed limits
end
```

Evaluators

- **Evaluators support parameters**
  - Some operators require parameters or qualifications
- **Example:**
  - **after**: temporal evaluator that correlates two events according to their occurrence time. It may take parameters to further qualify the temporal relationship.

```
rule “Stock price sudden drop”
when
    $st1 : StockTick( $pr : price )
    $st2 : StockTick( this after[1s,10s] $st1,
                      price < ($pr * 0.9) )
then
    // more than 10% drop in stock price within 10s
end
```

Evaluators (operators)

- **To register: either call the API or use a configuration file**

```
KnowledgeBuilderConfiguration conf = ...
conf.setOption(EvaluatorOption.get( “isWithin",
                        new IsWithinEvaluatorDefinition() ) );
```

```
drools.evaluator.isWithin = org.foo.IsWithinEvaluatorDefinition
```

Domain Specific Languages

- **Higher level language syntax**
  - Specific to a domain
  - Usually structured-XXX language
    - e.g. structured-English
  - Simple to define grammar
- **Drools:**
  - Simple sentence template engine
  - Maps arbitrary sentences into DRL code
  - Support to regular expressions for flexibility

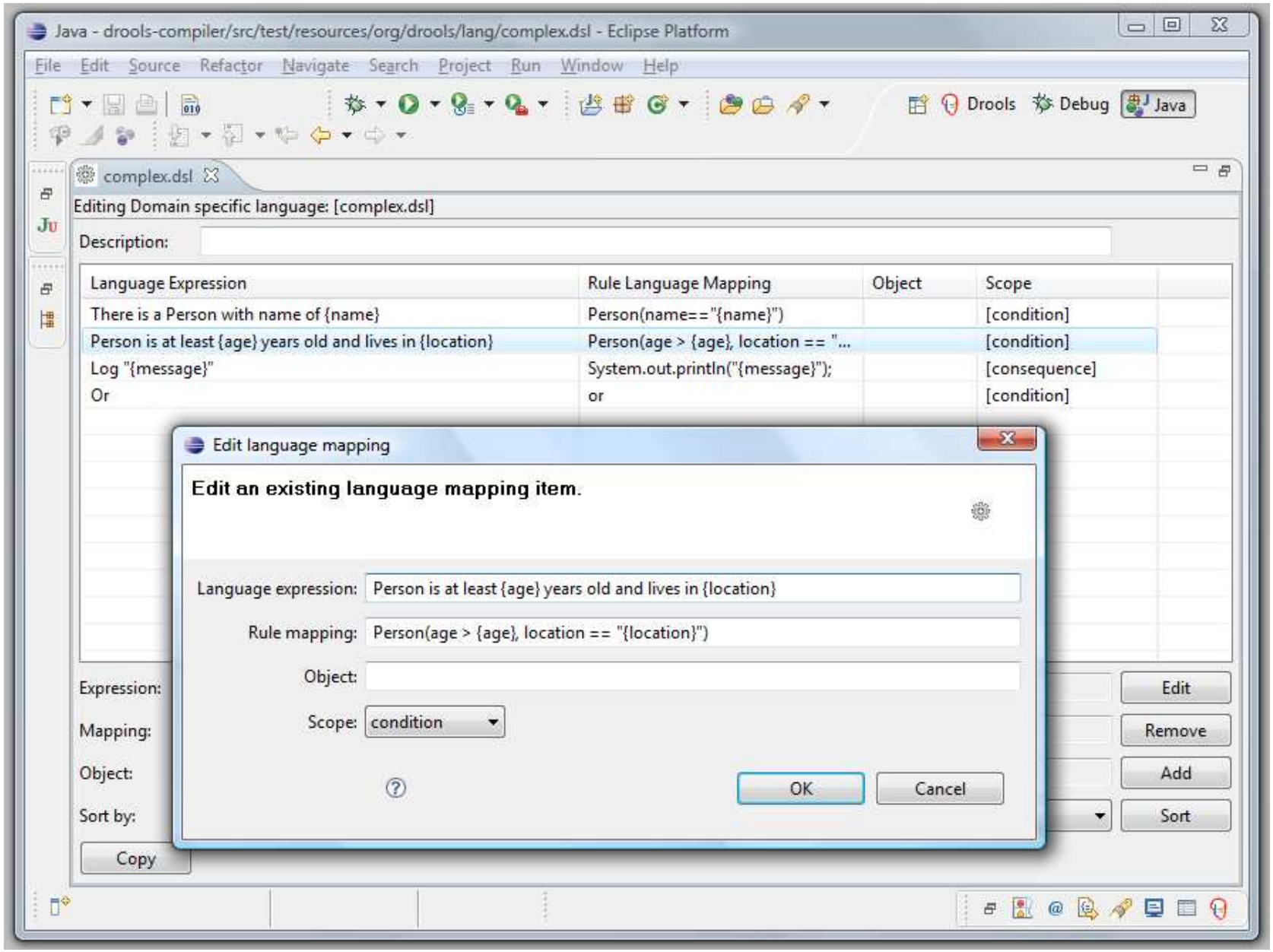
```
[condition][]There is a {fact} = ${fact} : {fact}( )
[condition][]The SIN is {sin:regexp:\\d{3}-\\d{3}-\\d{3}} = Person( sin == {sin} )
[consequence][]Log {message} = System.out.println( {message} );
[keyword][]rule = regra
```

Domain Specific Languages

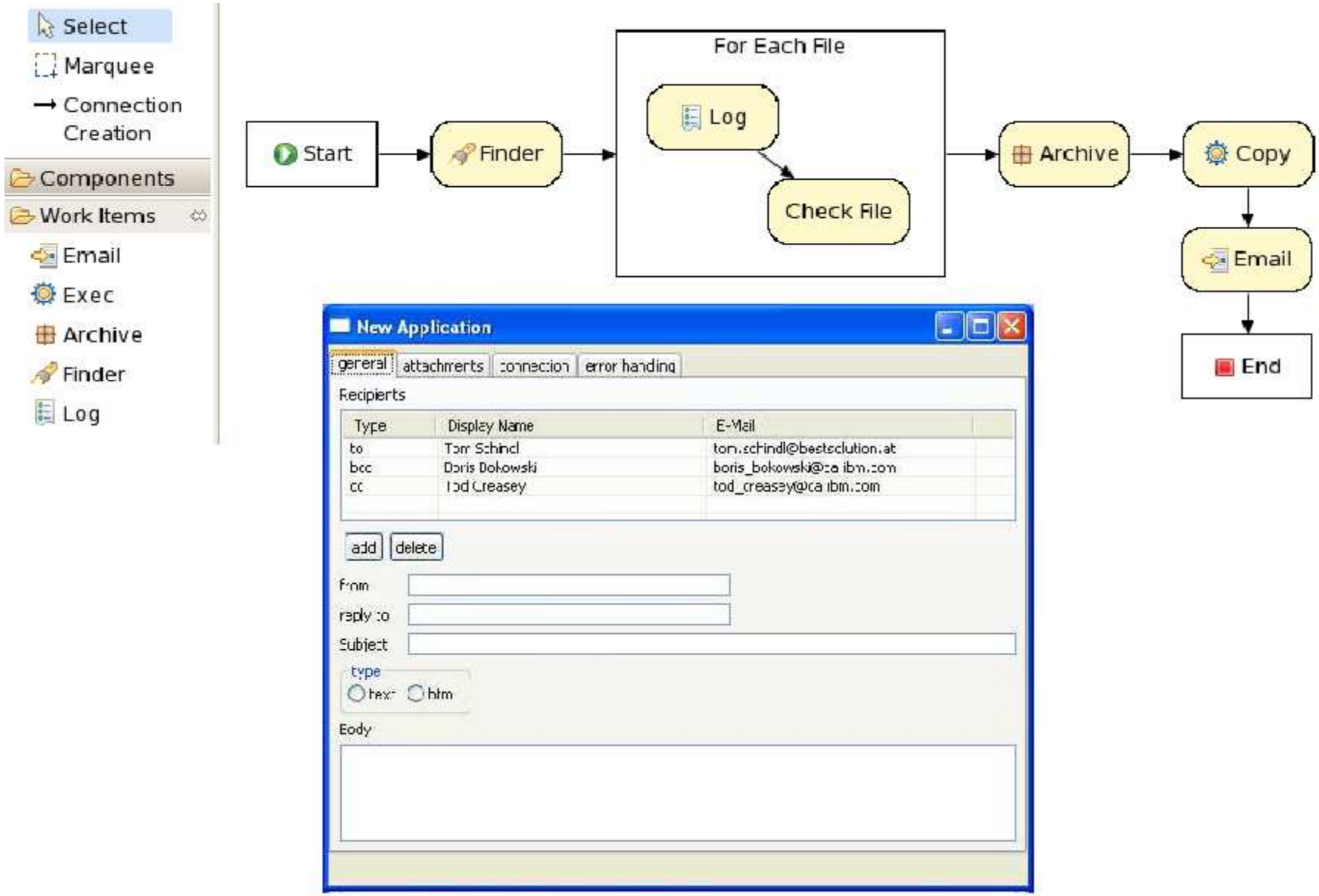
```
rule “Decrease balance on a debit operation”
when
   There is an account
   There is a transaction whose type is DEBIT
then
   Decrease the account balance by the transaction amount
end
```

```
rule “Decrease balance on a debit operation”
when
   $account : Account( $actNbr : accountNumber )
   $transaction : Transaction( accountNumber == $actNbr,
                               type == DEBIT )
then
   $account.balance -= $transaction.amount;
end
```

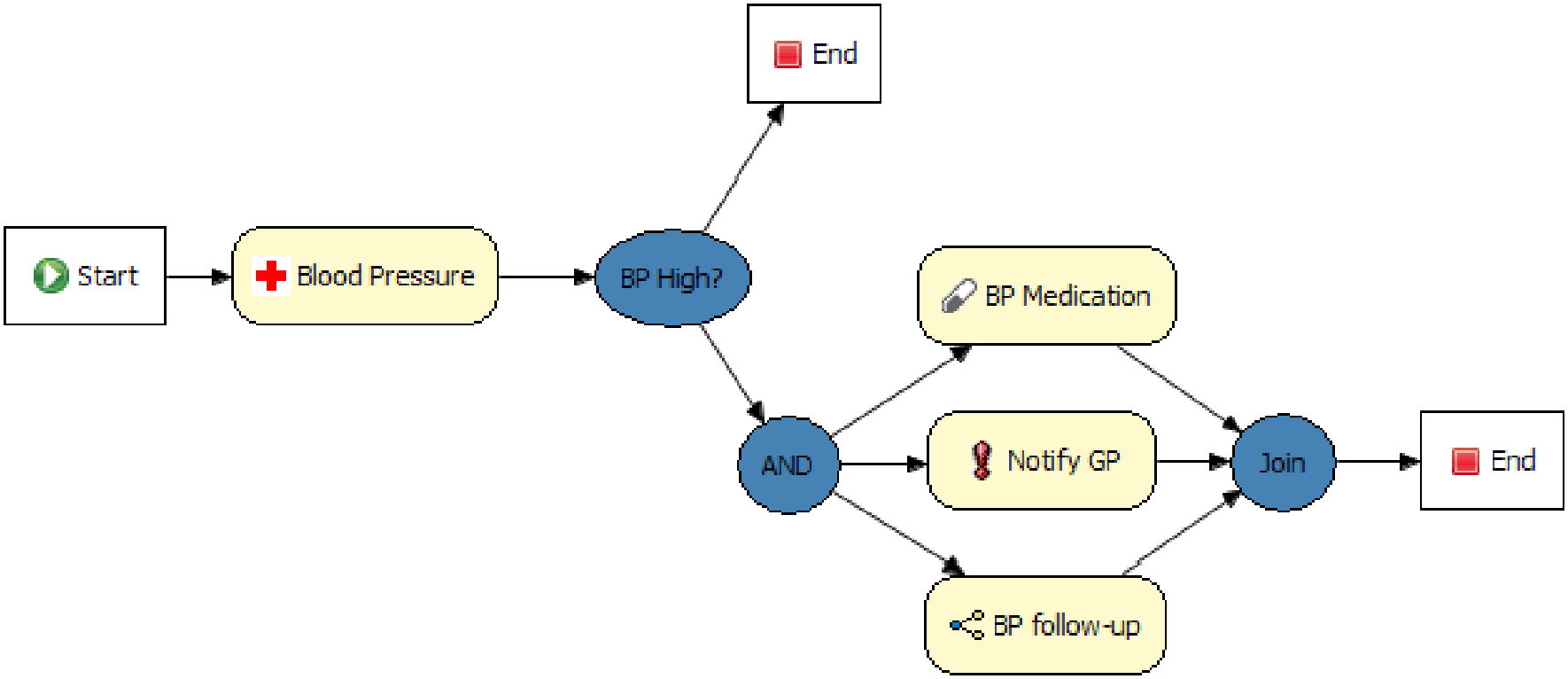
Domain Specific Languages



Domain Specific Processes



Domain Specific Processes



Domain Specific Work Items

- **To create: implement the interface**

```
public interface WorkItemHandler {
    void executeWorkItem(WorkItem workItem, WorkItemManager manager);
    void abortWorkItem(WorkItem workItem, WorkItemManager manager);
}
```

- **And add a configuration for it**

```
drools.workDefinitions = MyWorkDefinitions.conf
```

```
[
    "name" : "Log",
    "parameters" : [ "Message" : new StringDataType()  ],
    "displayName" : "Log",
    "icon" : "icons/open.gif“,
    "customEditor" : “org.drools.work.SampleCustomEditor“
]
```

Q&A

- **Drools project site:**
  - http://www.drools.org (http://www.jboss.org/drools/)
- **Documentation:**
  - http://www.jboss.org/drools/documentation.html

Edson Tirelli
etirelli@redhat.com
Senior Software Engineer
JBoss, a Division of Red Hat