Rules and Processes

Introduction

Agenda

- Rules
  - Structure
  - Best practices
- Other types of rules
  - DSL language
  - Decision Tables
  - Rule Templates
- Integration with processes
  - Business Rule Task
  - Ad Hoc Proceses
  - Task and Process Event management

What is a rule?

Rules have conditions and actions. The triggering of a rule can modify the information

Rules can trigger other rules, and attributes can decide when rules are available to trigger

```
rule "rule name"
//0+ rule attributes
when
  //1+ specific condition
then
 //1+ specific action
end
```

```
rule "our example"
no-loop
when
   i: Item(price < 22)
then
   modify (i) { setClassification("cheap"); }
end
```

Condition structure

- Based on Java objects (with getters or virtual properties)
- Plain model or nested objects? Plain objects preferred - could be declared in DRL too! -
- Condition examples

```
s: String(this == "hello")
o: Order(deliveryDate == null)
i: Item(price <= 22 && > 1, category == null) from o.getItems()
priceSum: Number() from accumulate(Item($p: price), sum($p))
```

Action structure

- Plain Java or MVEL
- Can modify the knowledge of the session
  - **insert(Object), delete(Object), update(Object):** Mark a specific object as new / not present / updated in the memory of the session, to inform the session it should reevaluate rules associated to those types
  - **insertLogical(Object):** Binds the insertion to the logic of the rule; If the condition that triggered this insertion stops being true, the object is automatically deleted from the session
  - **modify(Object) { code section }**: Update that has attribute granularity. Allows for specific rules that are property reactive (@watch only for specific property changes)

Rule attributes

- **no-loop:** This rule should not trigger itself (by modifying the data)
- **salience N:** Where N is a number that determines the priority
- **ruleflow-group NAME:** Determines a sub-group of rules to call from a process

Tricks and best practices

- **Rule atomicity:** make many rules that cannot be broken down any further
  - One condition, One action
  - Easier for the rule engine to evaluate
  - Easier to audit: following which rules trigger we can know why they did, without needing to look at the data
- **Rule attributes are for special cases**
  - Don’t try to put an order on your rules. The engine determines that order in a data driven way
- **Rule tricks:**
  - Init rule: A rule with no conditions. Will trigger only once
  - Queries, inline casting, OOPath...

Types of Rule files ← RuleTypesTest

DRL: Uses the syntax we discussed so far

DSLR: With the help of a DSL dictionary, it lets you create rules using predefined natural language phrases that are translated in realtime to DRL

Decision Tables: Translates a spreadsheet, row by row, into a different rule. Good for repetitive rules

Rule Template: Similar to Decision Tables, but the source data is not a spreadsheet, and is defined in runtime. i.e. a SQL or Lucene query.

THEY ALL TRANSLATE TO DRL

Business Rule Tasks

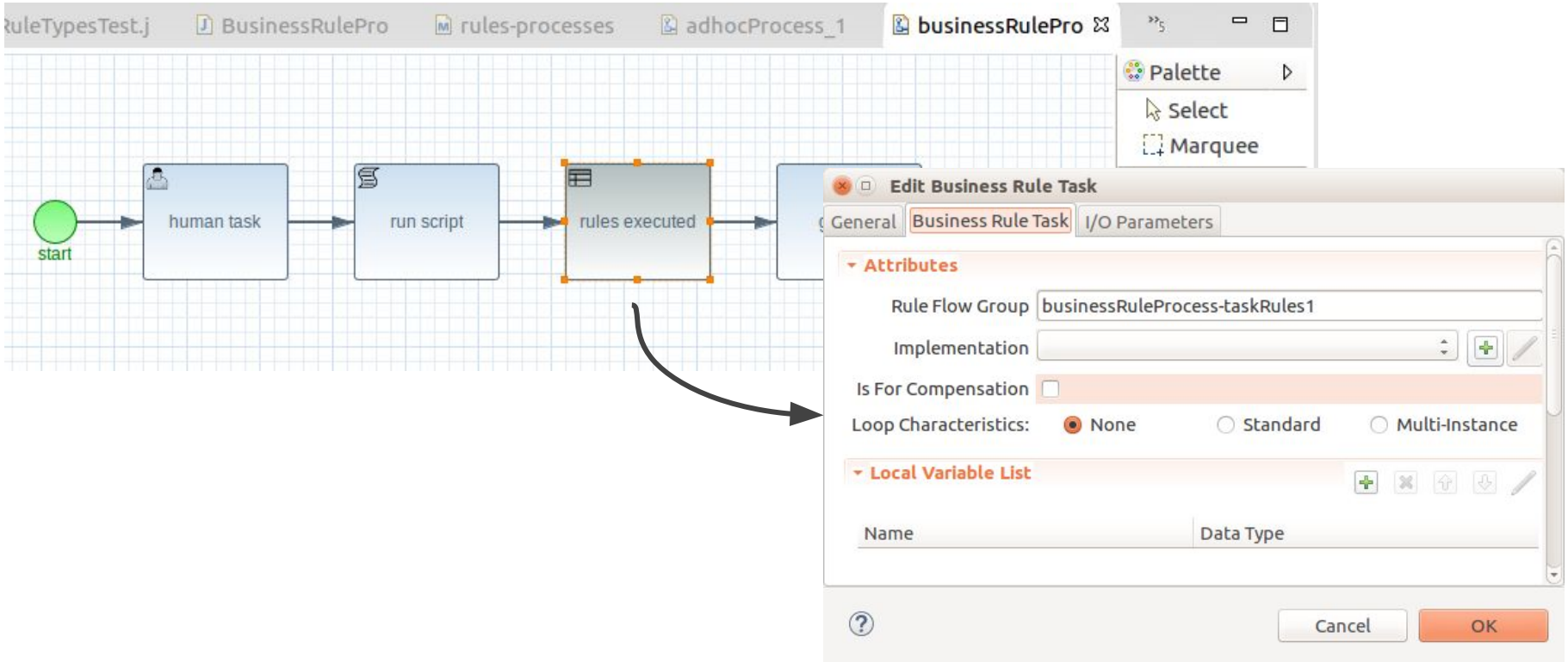
- **On the process**
  - **Business Rule Task:** Will define the connection point to the rules in the property **Rule Flow Group**
- **On the rules:**
  - **ruleflow-group:** Any rule with this attribute will be used when a process arrives at said task

Easy integration between rules and process. Everything happens in the same engine

A rule can also create or signal a process instance in the action part: kcontext variable

BusinessRuleProcessTest

Business Rule Task example



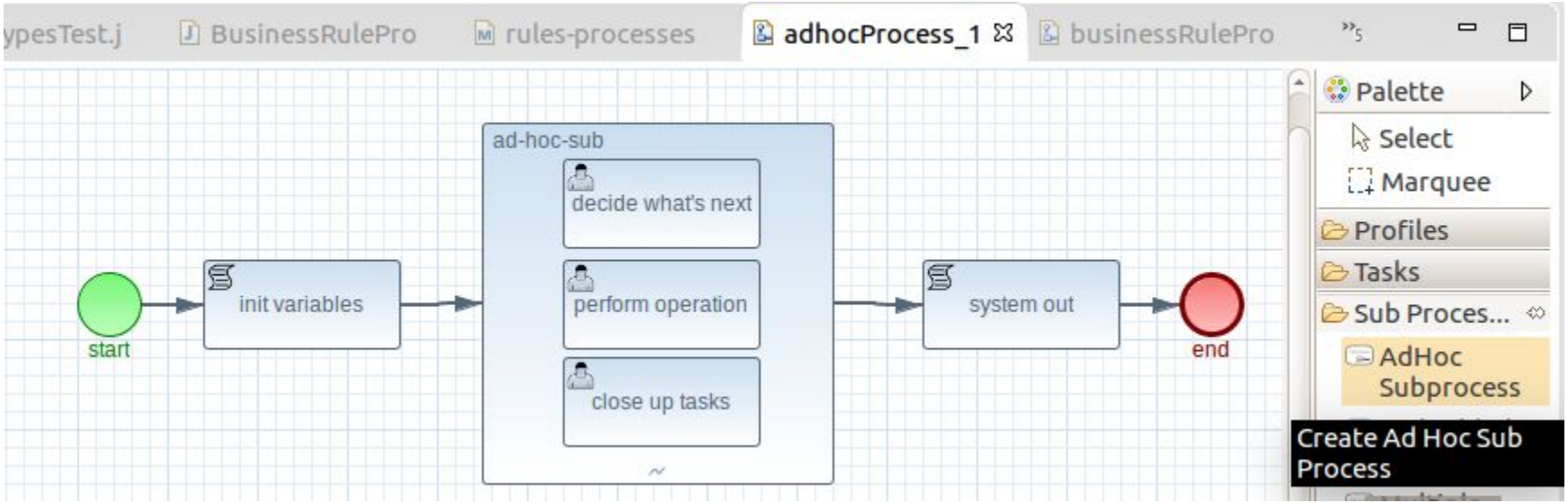
Business Rule Tasks best practices

- **Keep rule flow group process specific:** Always use the process ID as a prefix for the rule flow group name or names you will use from your rule
- **Use only when the process should determine when the rules fire:** For cases where the code or.other events should trigger the rule, there are other methods
- **Nesting process -> rules -> process**: If it makes sense, the rules triggered by one process could create a new process instance. Everything will be handled inside the same ksession
- **Avoid a lot of gateways in a process:** If it needs to make a decision based on multiple conditions, use rules
- **Avoid following a sequence of steps in the rules:** If it needs to do so, create a process

Ad Hoc Processes

- Decisions on which is the next task to execute can be too complex for putting on a diagram
- Dynamic of next task to perform is too complex?
- Diagram is too clotted with gateways?
- Ad hoc processes just define the tasks involved in a sub process, but not how they are connected
    - A completion condition is defined to determine when the ad hoc sub process should finish
    - The next task to be created in the ad hoc subprocess execution is determined by an external component: either manually, or **by rules**

Ad Hoc Process example



Ad Hoc Process disclaimers

- **Very little use by the jBPM community:** Whatever we do around ad hoc processes governed by rules will be custom for us
- **Requires custom code to connect rules and next task to be generated:** In our code, KieAdHoc class does this for us
  - No specifics on how to connect rules and processes. Left ambiguous on purpose
  - Asdf
- **AdHocProcessTest**

Other process / rules integration mechanisms

Common places for integrating Rules and Processes / Tasks:

- **At task events:** Using a TaskLifeCycleEventListener to gather TaskEvent objects, and then creating rules to govern different situations
  - Something similar is currently supported for task addition and completion
- **At process events:** Using a ProcessEventListener to gather events and determine what to do if many processes are executed at the same time
  - Con: You need a separate session to manage multiple processes at once because of our current runtime manager implementation
- **Anywhere complex decision has to be made** with a set of given data, whether it is process related or not
  - Simply create a session, insert the data, and fire the rules

Other process / rules integration mechanisms

```
global AlertSystem alerter;

rule "monitor active tasks"
when
    i: Number(intValue > 100) from accumulate(
        t: Task(taskData.status == Status.Ready || taskData.status == Status.Reserved),
        count(t)
    )
then
     alerter.alertAdmins("Too many currently open tasks! " + i);
end
```

Questions?

Thank you!