goto; conference aarhus

JBoss Enterprise BRMS

Building highly scalable process and rule-driven applications

Eric D. Schabell

JBoss Technology Evangelist

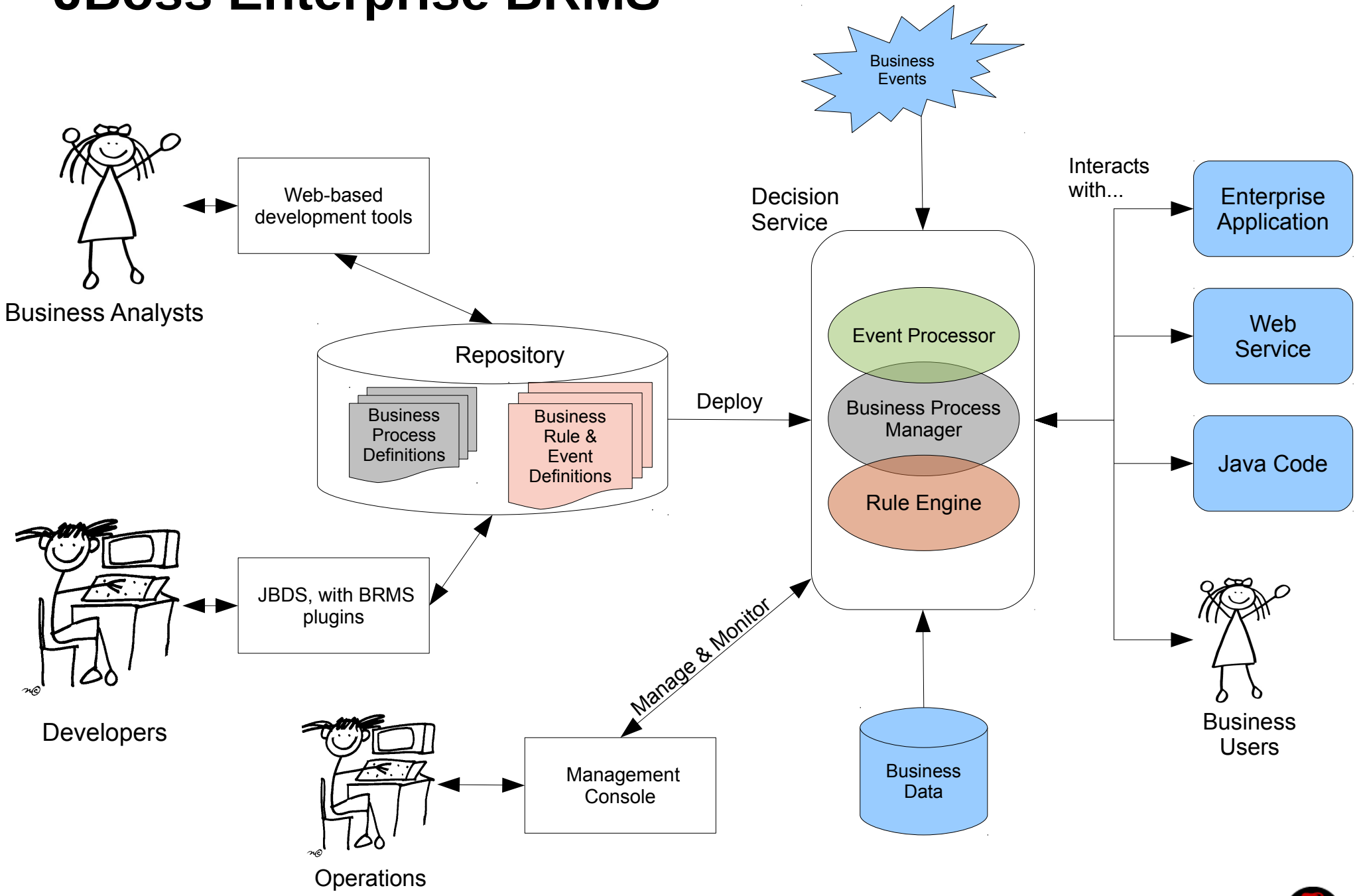
INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

gotocon.com

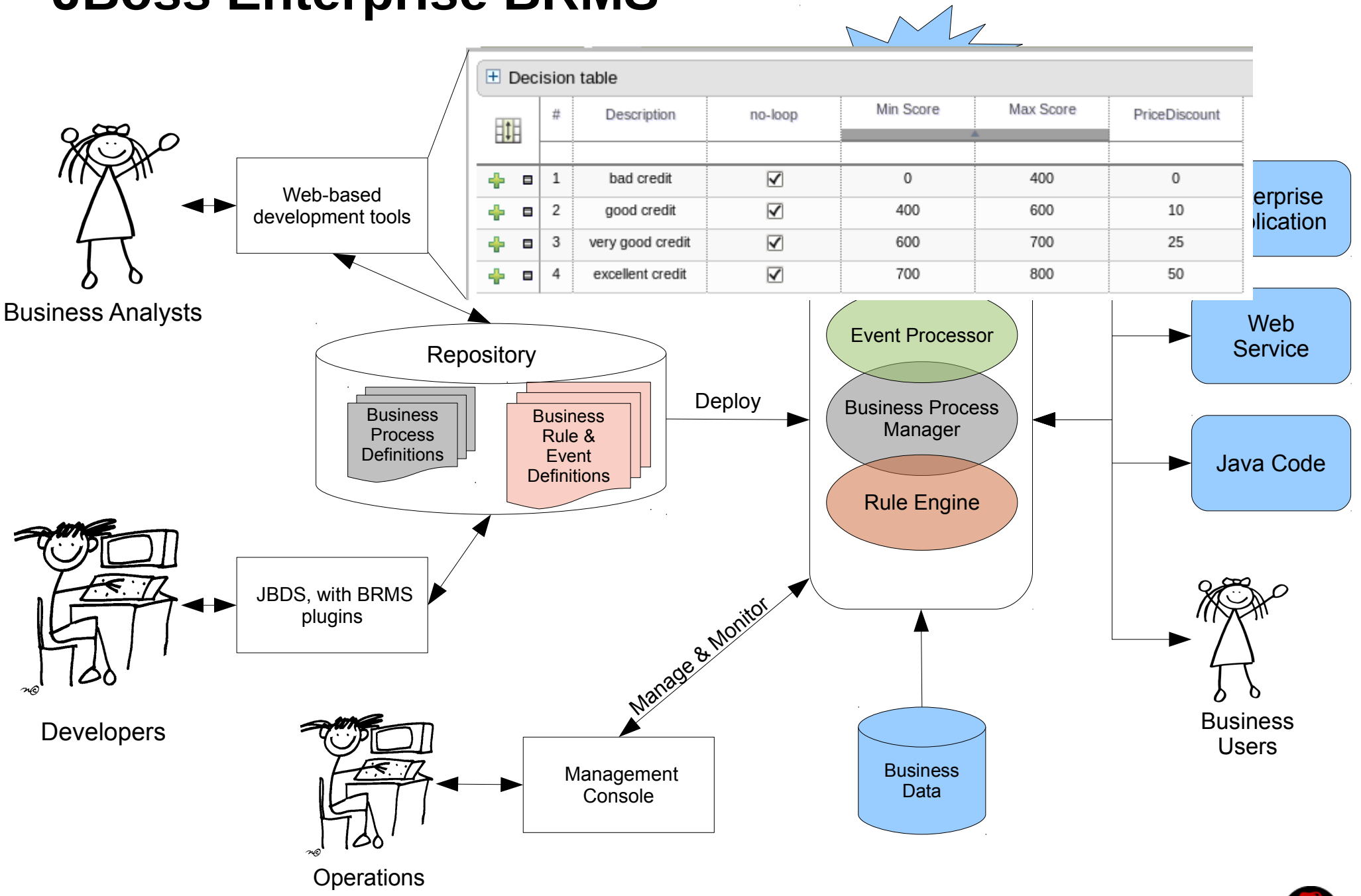
Developing BRMS Applications that Scale

- Overview
- BRMS Adoption Goals
- Under the Hood
- Best Practices
  - *BPM*
  - *Architecture*
  - *Rules Authoring*

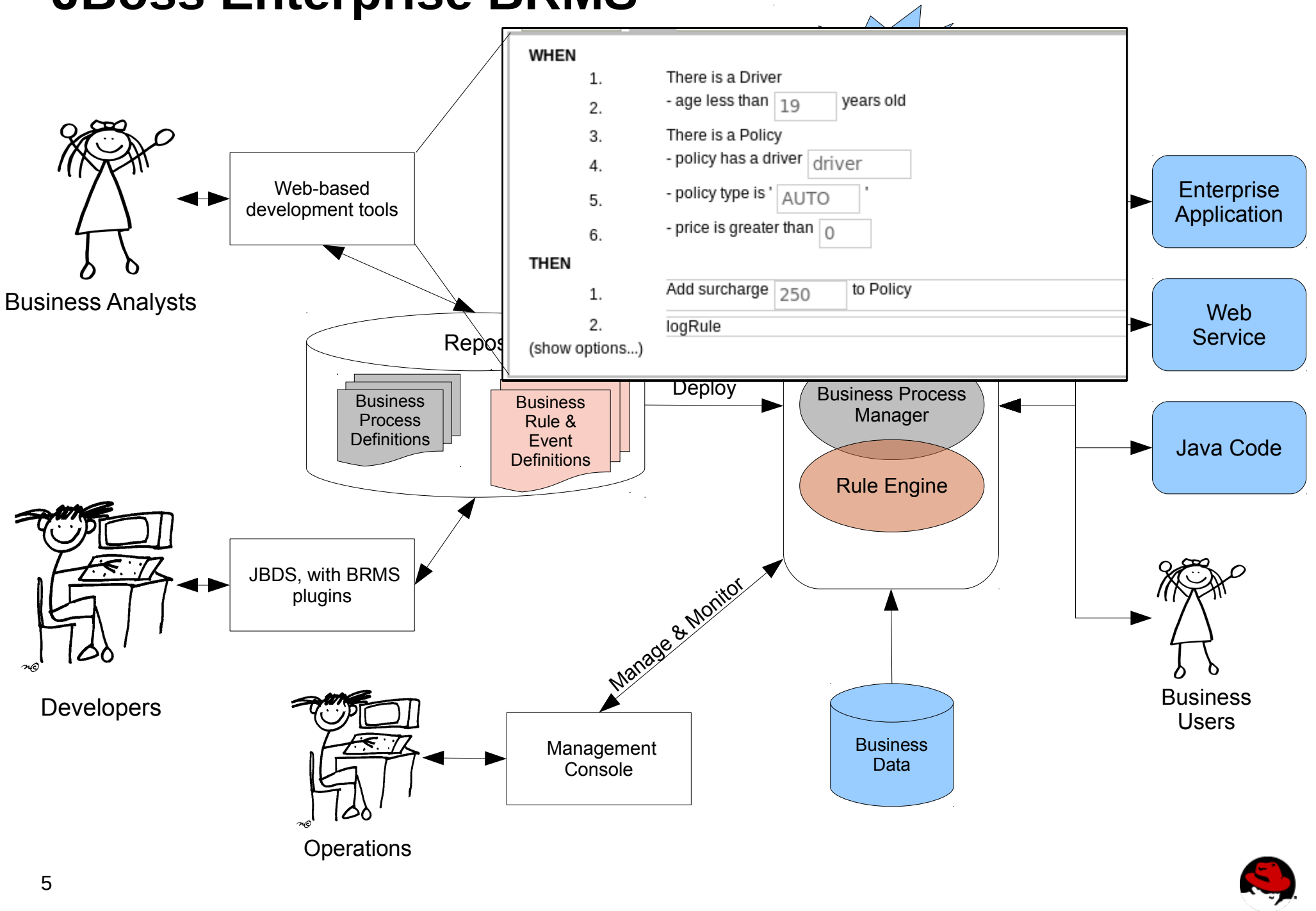
JBoss Enterprise BRMS



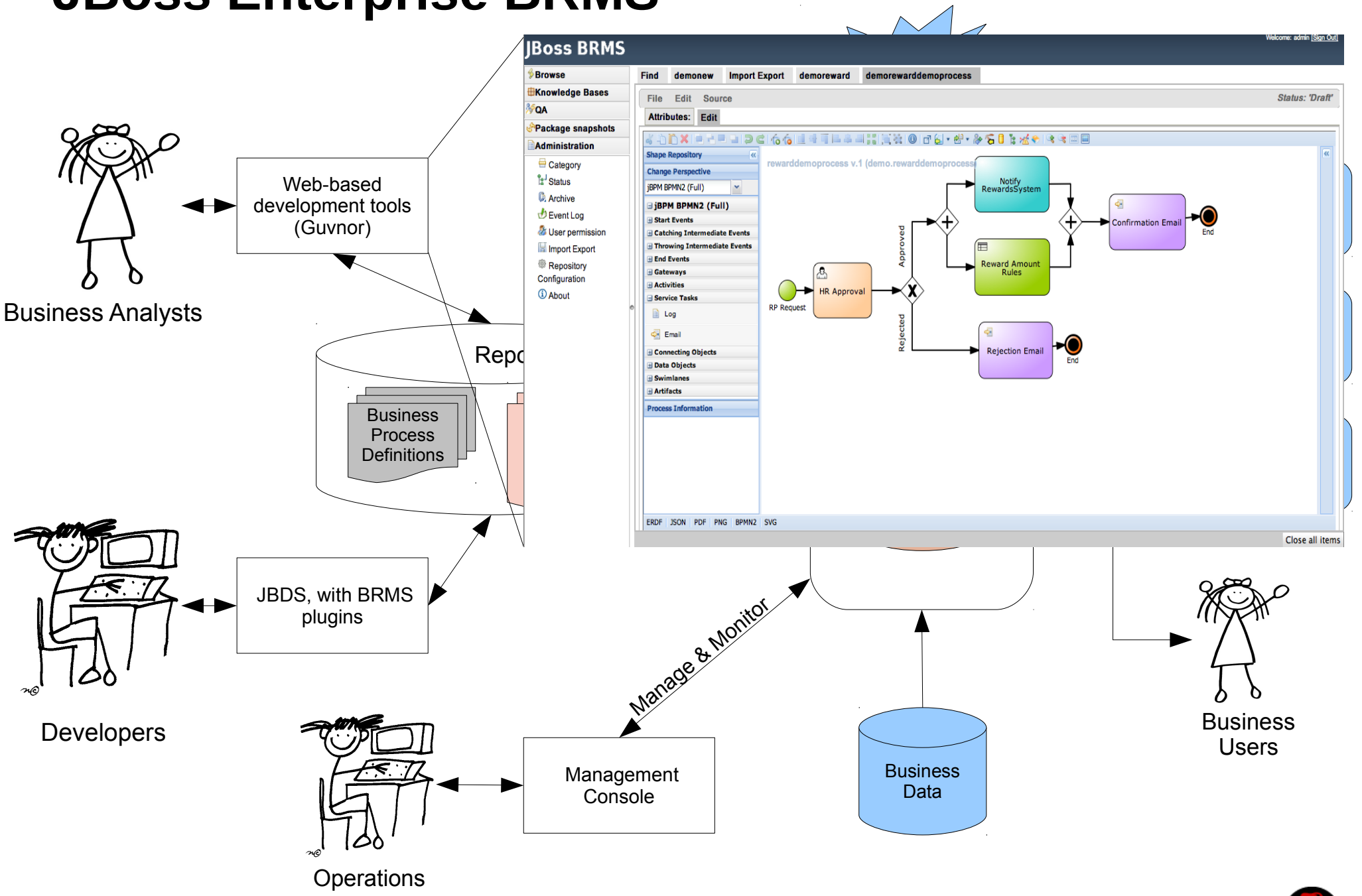
JBoss Enterprise BRMS



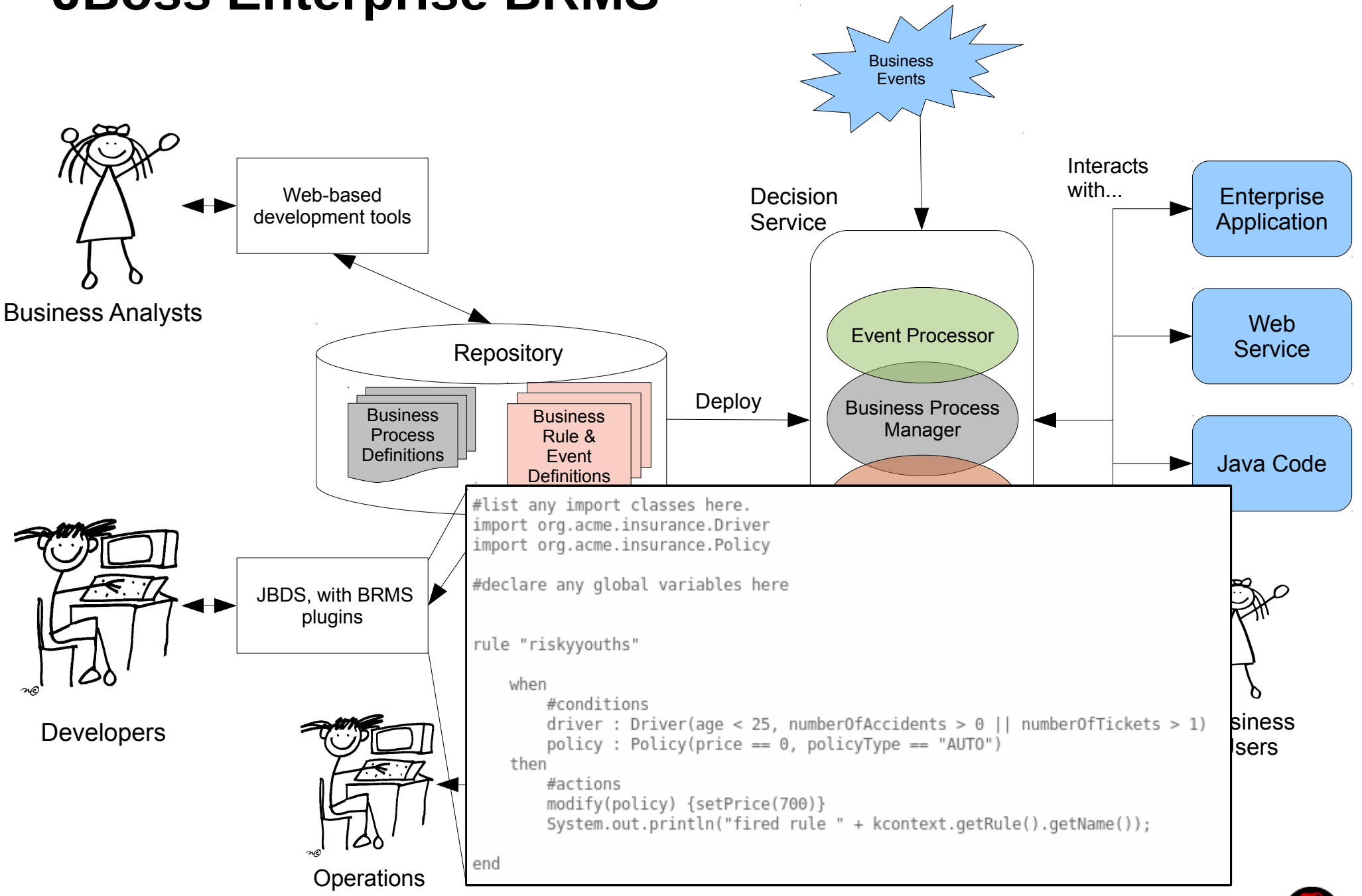
JBoss Enterprise BRMS



JBoss Enterprise BRMS

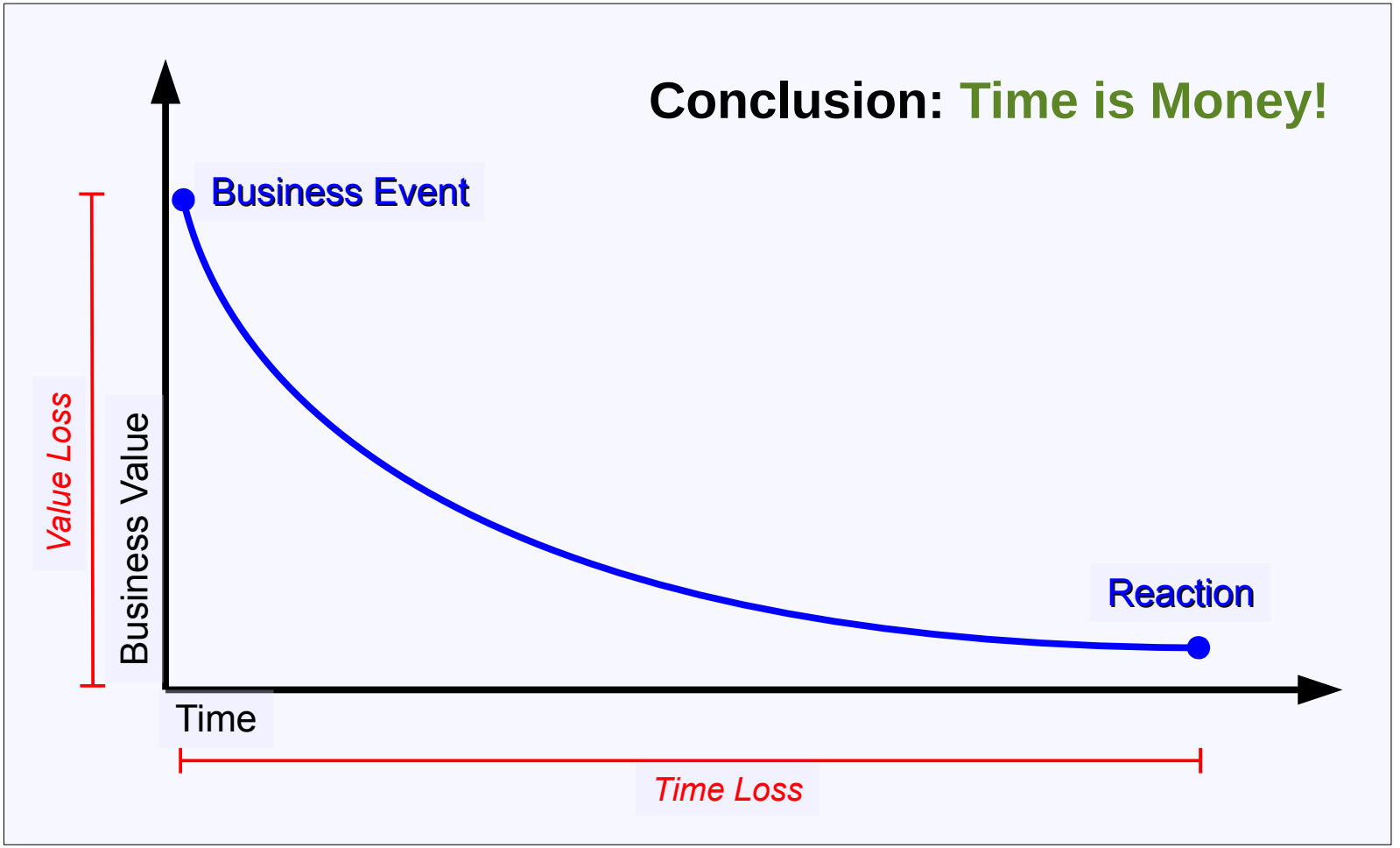


JBoss Enterprise BRMS



BRMS adoption goals

Goal #1: Decision Automation



Adapted from a presentation by James Taylor, Sep/2011

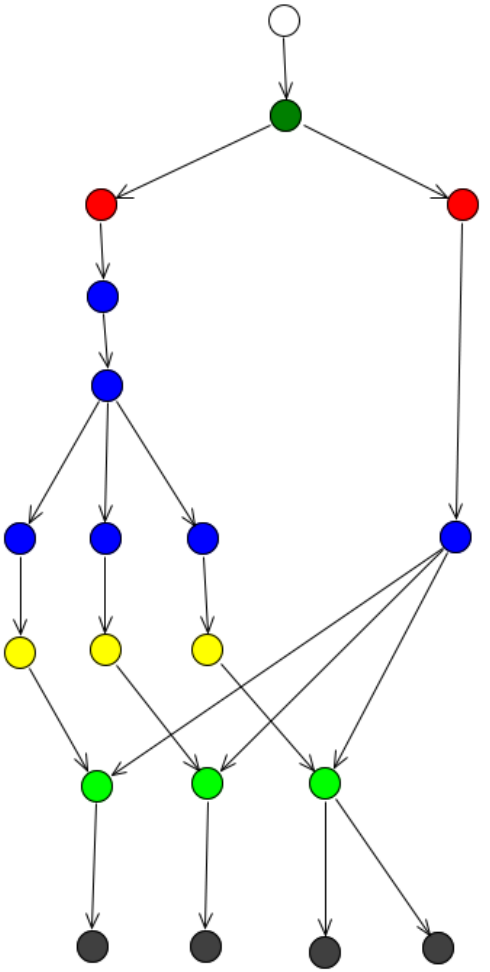
Goal #2: Expressiveness and Visibility

```
rule “Send shipment pick-up SMS alert”
when
    There is a shipment order
    There is a route assigned to the order
    There is a truck GPS reading and the truck is 15 minutes
                away from the pick-up location
then
    Send SMS to customer: “Arriving in 15 minutes”
end
```

*Focus on “**what to do**” instead of “how to do it”*

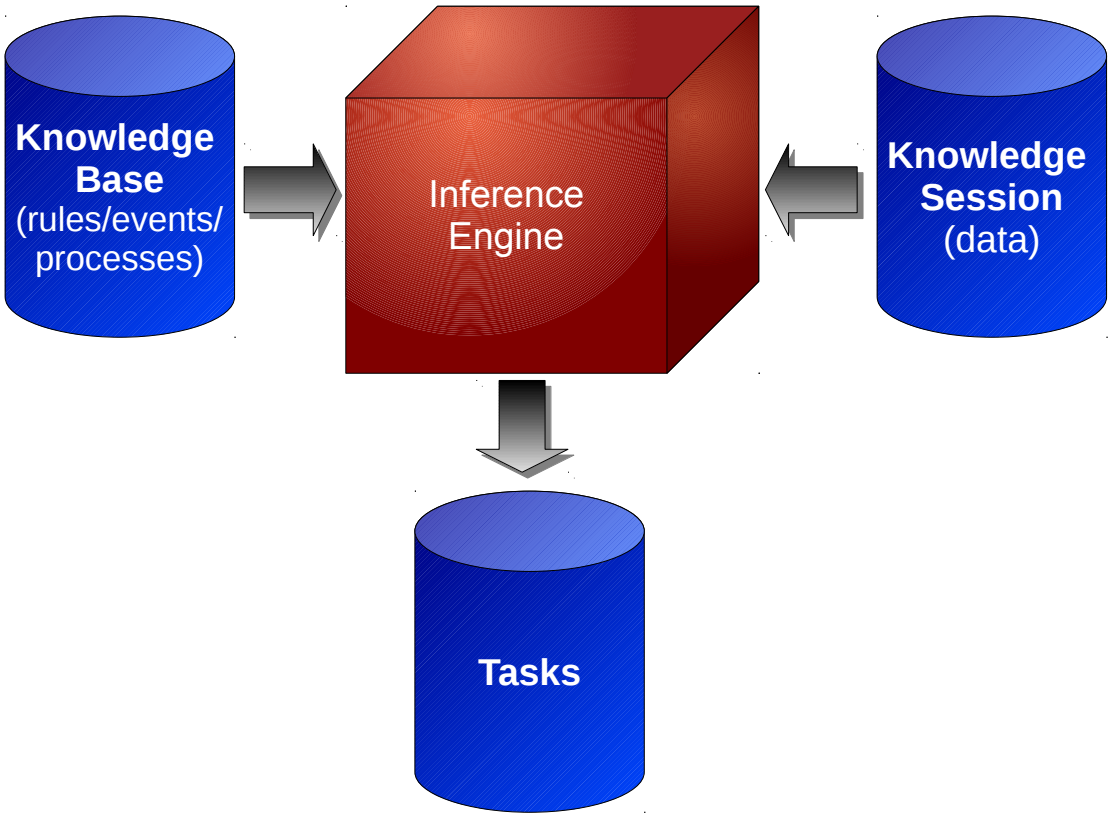
Goal #3: Performance and Scalability

- **Real time, online, systems**
  - *Millisecond response times*
- **Hundreds of thousands of rules**
  - *JBoss BRMS: 700k+ rules*
- **Millions of data instances (facts)**
- **Incremental (re-)evaluation**
  - *Changes in data can't reset reasoning*



Goal #4..#6: other technical goals

- **Logic, Data and Tasks split**
- **Centralization of Knowledge**
  - Consistency
  - Testing / Simulation
  - Auditing
- **Explanation Facility**



JBoss BRMS: under the hood

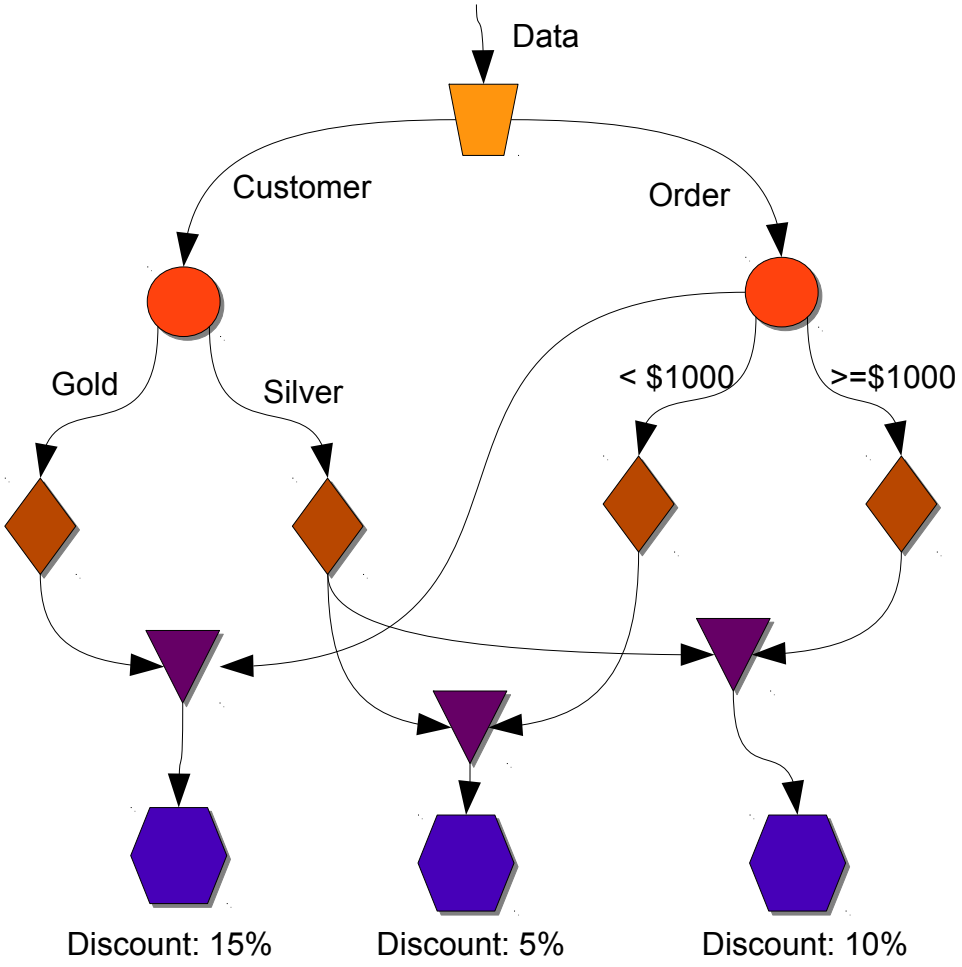
Engine's Algorithm – 30 seconds crash course

Decision Table: User's View

Customer	Order Total Amount	Discount
Gold		15%
Silver	< $1000	5%
Silver	>= $1000	10%

Clear, Concise, Objective

Rete Network: Computer's View

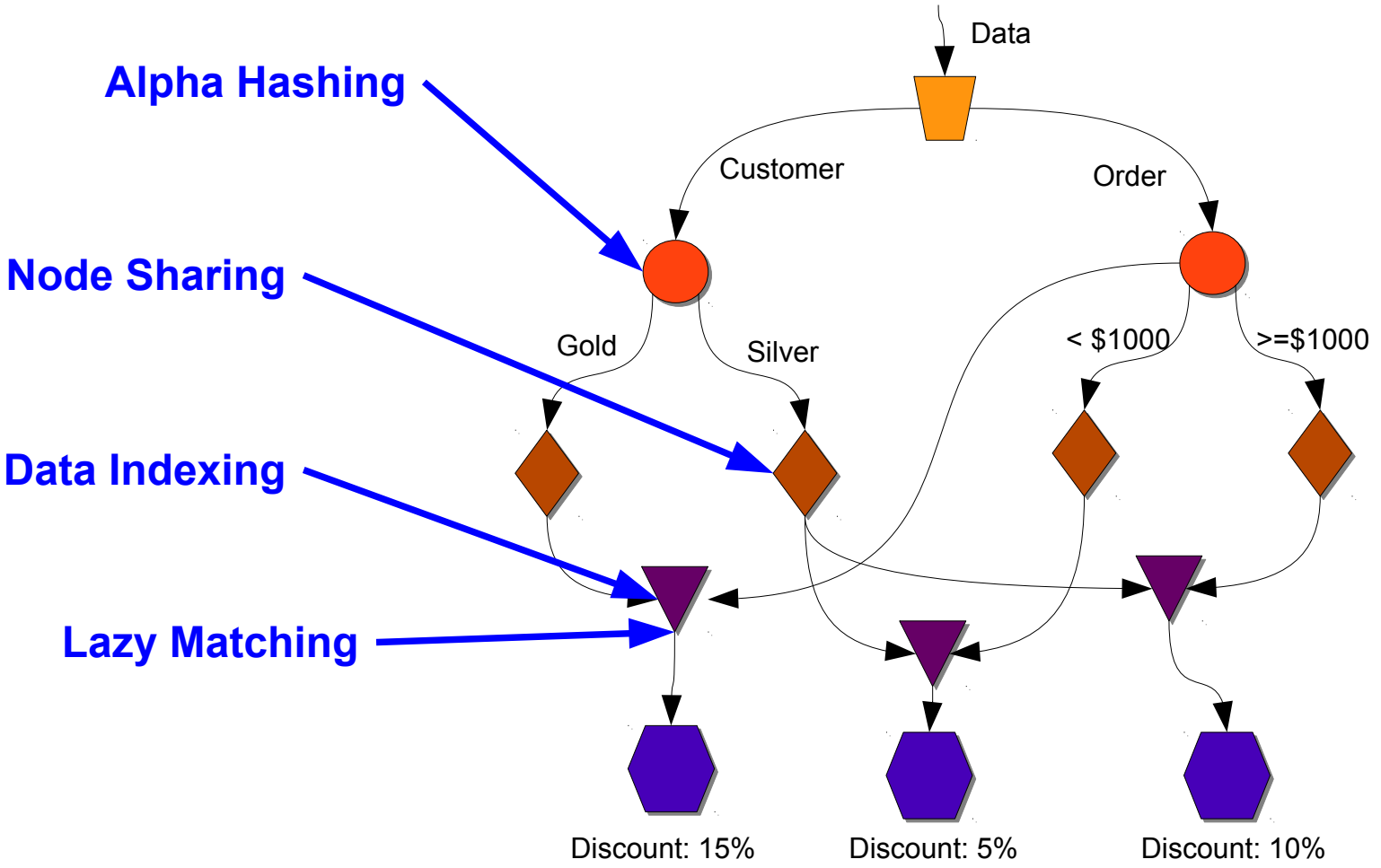


Efficient, Effective

JBoss BRMS – some optimizations

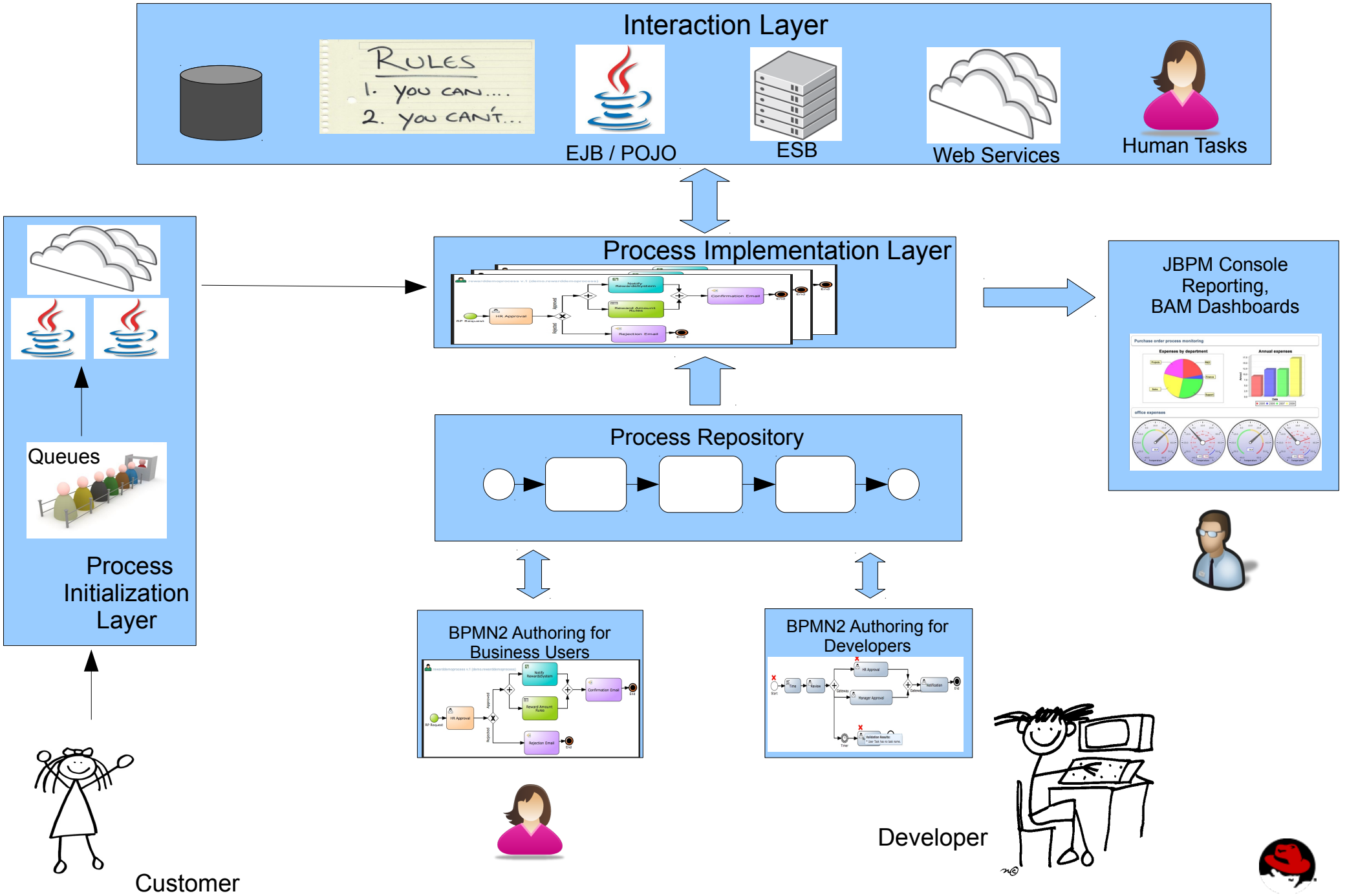
- **Support to POJOs as facts**
  - no mapping/data copy necessary
- **Full split between Knowledge Base and Session**
  - lightweight session creation
  - knowledge base sharing
- **Completely Dynamic Knowledge Base**
  - Hot addition/removal/updates of rules/queries/processes
- **Full support to First Order Logic and Set operations**
- **JIT compilation for constraints and data access**

JBoss BRMS – More optimizations



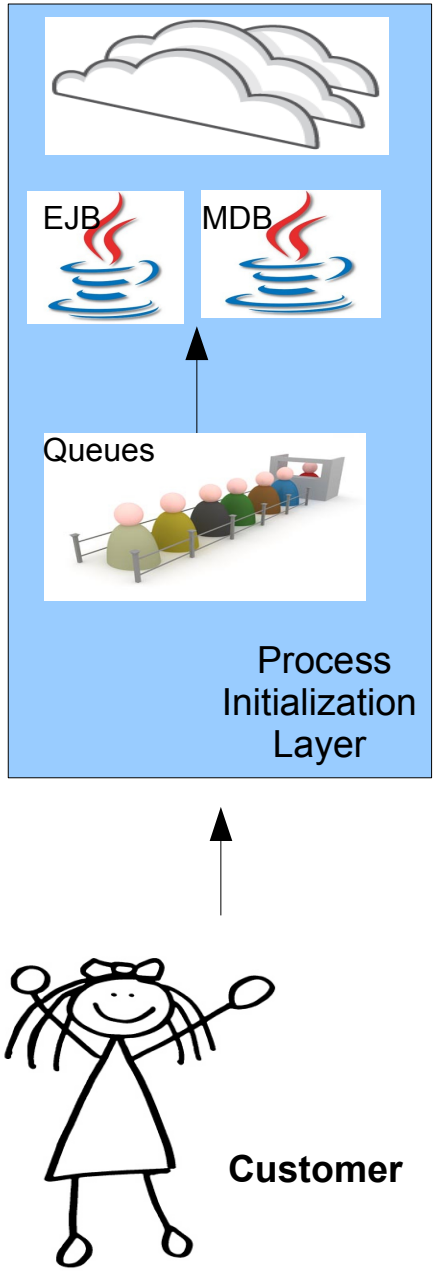
JBoss Enterprise BRMS

Best Practices – BPM Architecture

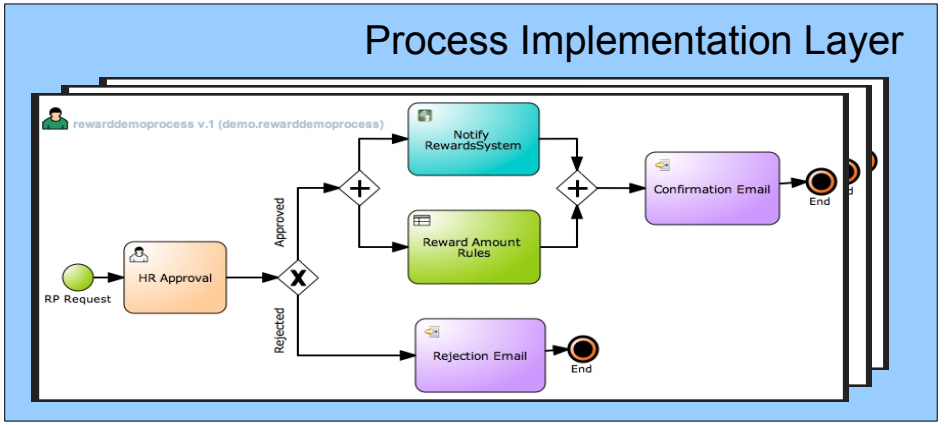


Best Practices – Process Initialization

- How do you start your process?
  - web services, EJB's, API call, RESTful
  - what about prioritization of processes
    - use message queues
    - other complex ideas to start processes
- Centralize startProcess in single location
  - minimizes change effects in this layer



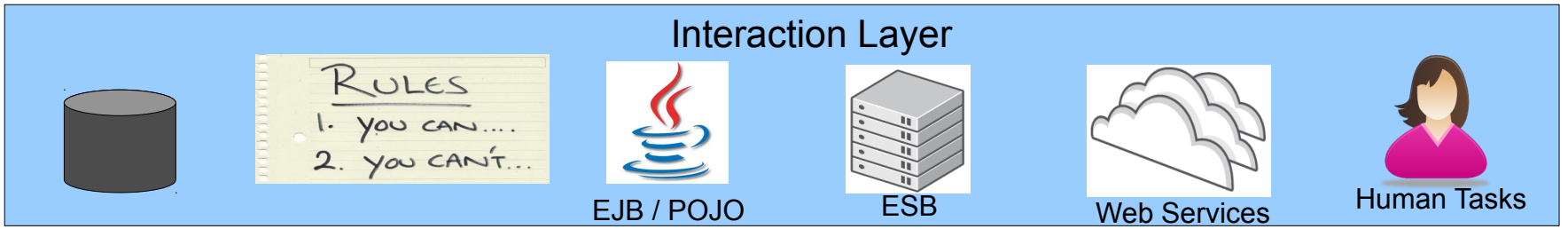
Best Practices –
Process Implementation



- Java nodes
  - do you keep it clean?
    - single unit of action per process step
    - human task / admin interfaces
- Centralize you jBPM API access
  - single WS / DAO / BOM / RESTful
- Domain specific nodes
  - extensions via work item handlers
- Design process for reuse
  - smallest unit of work

Best Practices – Process Interaction

- Processes interact with your Enterprise
  - Web Services, EJB, GUI, POJO, Exceptions, Bean Script, Rules...
    - jBPM API & jBPM History DB & RESTful
    - history / tasks / reporting
      - single DAO
      - single Web Service
      - JBoss ESB + jBPM
      - externalize rules calls in Decision Service (WS)

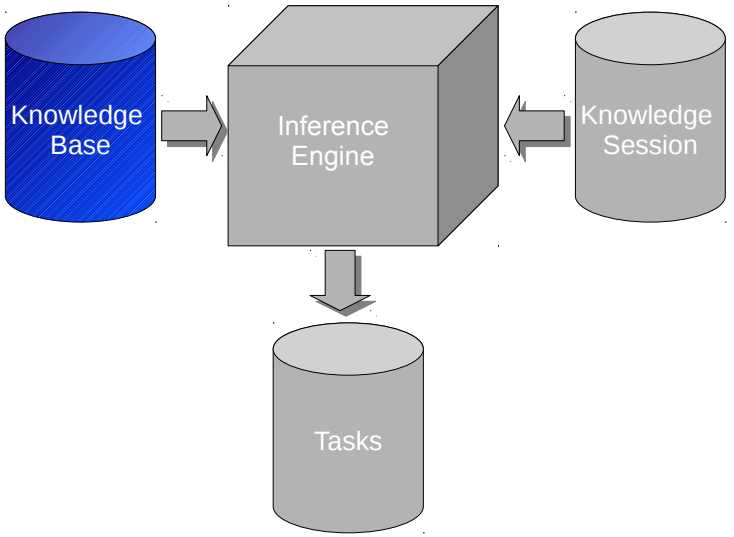


Best Practices – Always good BPM practices...

- Simplify everything (KISS)
  - apply OO to process design
    - methods == sub-process + context in/out
    - encapsulate == sub-process
    - reuse == process repo (maven potential)
    - unit testing == per node, sub-process, process
    - exception handling (Exception Framework)

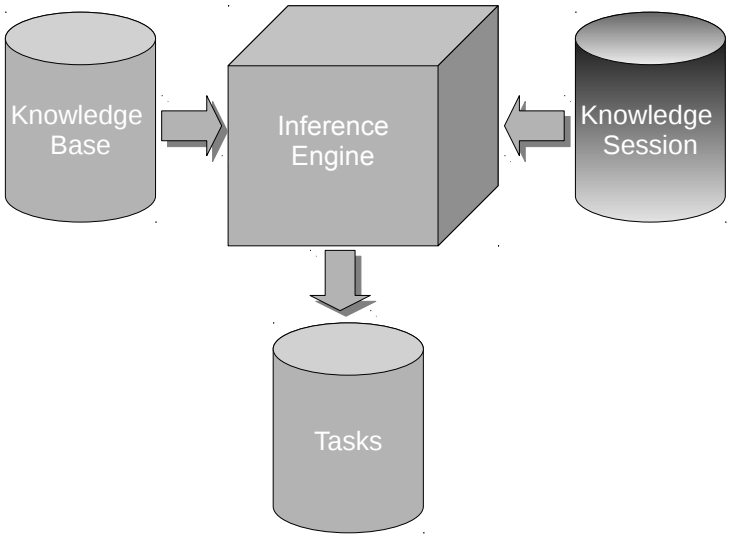
Best Practices – Rule Architecture

- **Partition your Knowledge Bases properly**
  - Subject matter
  - Transaction / Service / Unit of Work
  - Business Entity
- **Avoid monolithic Knowledge Bases**
- **Avoid fine grained Knowledge Bases**



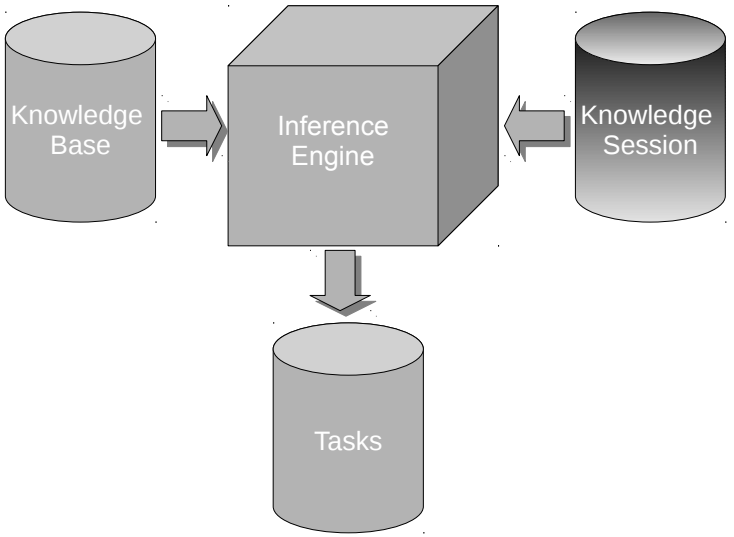
Best Practices – Rule Architecture

- **Batch data loads**
  - Load 1000 facts and fire the rules faster than fire rules after each loaded fact
- **Partition the data into multiple sessions**
  - Transaction / Service / Unit of work
- **Creating a new session is cheap**
  - Cheaper than removing facts



Best Practices – Rule Architecture

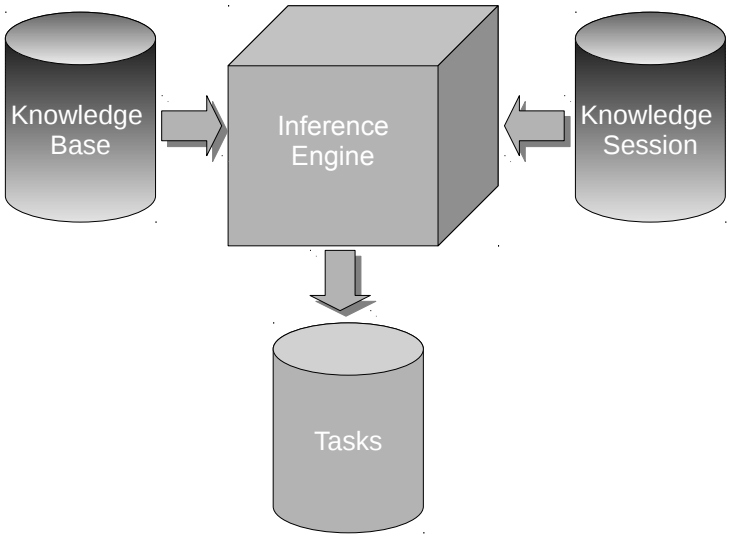
- **Quality of the data/fact model is directly proportional to the performance and maintainability of the rules using it**
  - Think about the DBMS analogy
  - Flatter models improve performance
  - Smaller classes help avoiding recursions



JBoss BRMS – Best Practices in Rules Authoring

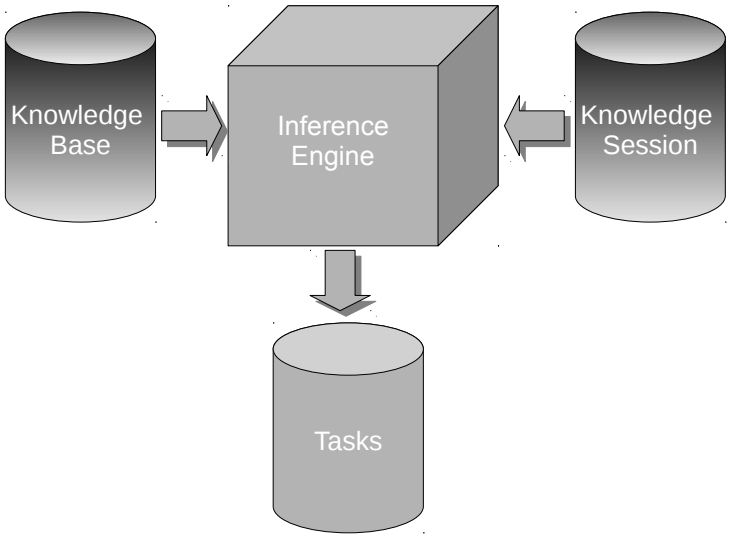
Best Practices – Rules Authoring

- **Don't try to micro-control rules execution**
  - Use the Conflict Resolution Strategies instead
    - Salience
    - Agenda groups
    - Ruleflow / Processes
    - Dynamic Enablement



Best Practices – Rules Authoring

- **Don't overload rules**
  - Each rule should describe one and only one scenario→action mapping
  - The engine will optimize shared conditions
  - The engine supports inference



~~Best~~ Bad Practices – Rules Authoring

```
rule “1 – Teenagers and Elders get Discount”
when
    Person age is between 16 and 18 or Person age is greater or equal to 65
then
    Assign 25% ticket discount
end
```

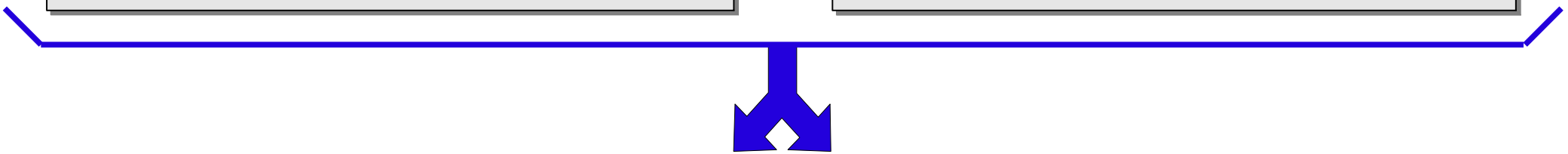
```
rule “2 – Elders can buy tickets in area A”
when
    Person age is greater or equal to 65
then
    Allow sales of area A tickets
end
```

Rules are being overloaded with multiple concepts, increasing maintenance and testing costs.

Best Practices – Rules Authoring

```
rule "0.a – Teenagers are 16-18"
when
    Person age is between 16 and 18
then
    Assert: the person is a Teenager
end
```

```
rule "0.b – Elders are older than 65"
when
    Person is older than 65
then
    Assert: the person is an Elder
end
```



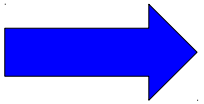
```
rule "1 – Teenagers and Elders get discount"
when
    Teenager or Elder
then
    Assign 25% ticket discount
end
```

```
rule "2 – Elders can buy tickets in area A"
when
    Elder
then
    Allow sales of area A tickets
end
```


Best Practices – Rules Authoring

- **One calculation (accumulate) per rule**
  - Accumulates have O(n) performance
  - Sequences of accumulates have $O(n^m)$ performance
    - n = number of matching facts
    - m = number of accumulates

```
rule “Sum debits and credits”
when
    accumulate( Debit( $d : amount ),
                $debits: sum( $d ) )
    accumulate( Credit( $c : amount),
                $credits: sum( $c ) )
then ...
```



```
rule “Sum debits”
when
    accumulate( Debit( $d : amount ),
                $debits: sum( $d ) )
then ...
```

```
rule “Sum credits”
when
    accumulate( Credit( $c : amount ),
                $credits: sum( $c ) )
then ...
```


Best Practices – Rules Authoring

- **Rules vs Queries**

	Rules	Queries
Control	Invoked by the engine	Invoked by the application
Parameters	Don't support parameters	Support parameters
Results	Execute actions	Return results

```
rule “Approve VIP customers”
when
    $c : Customer( type == “VIP” )
then
    insert( new Approved( $c ) );
end
```

```
query “Get customers by type”( $type )
when
    $c : Customer( type == $type )
end
```

“Use the right tool for the right job!”

Best Practices – Rules Authoring

- **Declared Types**
  - Facts used only by the rules
  - Facts that change frequently with the rules
- **POJOs**
  - Facts shared by both rules and application
    - No data copy – very efficient
  - Easier to integrate, easier to test
- **When in doubt, use POJOs**

"Use the right tool for the right job!"

- JBoss Enterprise BRMS Best Practices Guide
  http://www.redhat.com/promo/integrated_enterprise/brms-best-practices-form.html
- JBoss Enterprise BRMS Best Practices, Edson Tirelli,
  http://www.redhat.com/summit/sessions/jboss.html
- JBoss BRMS,
  http://www.redhat.com/products/jbossenterprisemiddleware/business-rules/
- BRMS Best Practices Process Initialization Layer,
  http://howtojboss.com/2012/08/15/brms-best-practices-process-initialization-layer/

goto; conference aarhus

Questions?

INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

gotocon.com