PRESENTED BY RED HAT

LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.

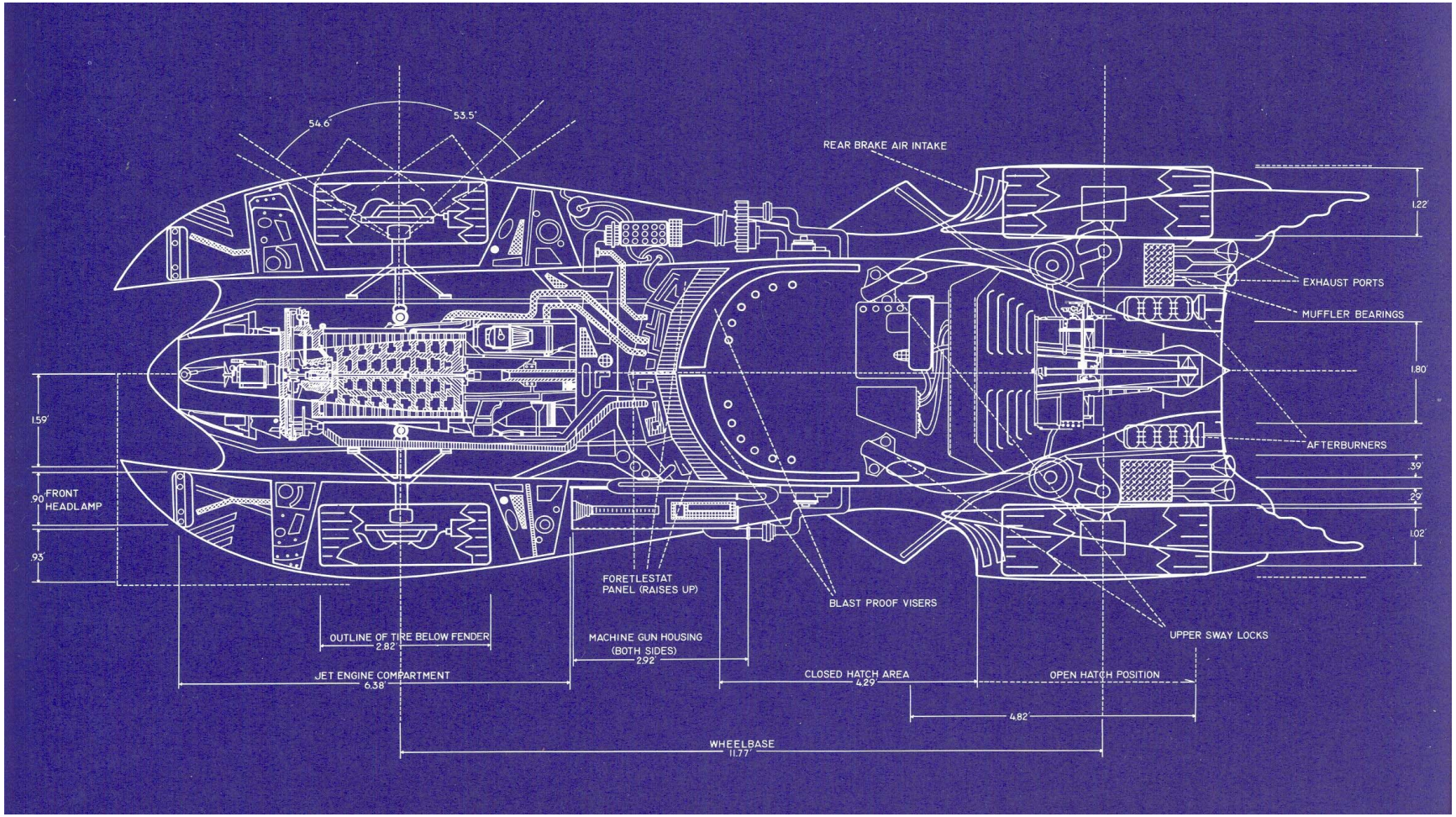
JBoss Enterprise BRMS:

Best Practices

Edson Tirelli
Principal Software Engineer, Red Hat
06.27.12

Why use best practices / blue prints?

It is not just about performance...

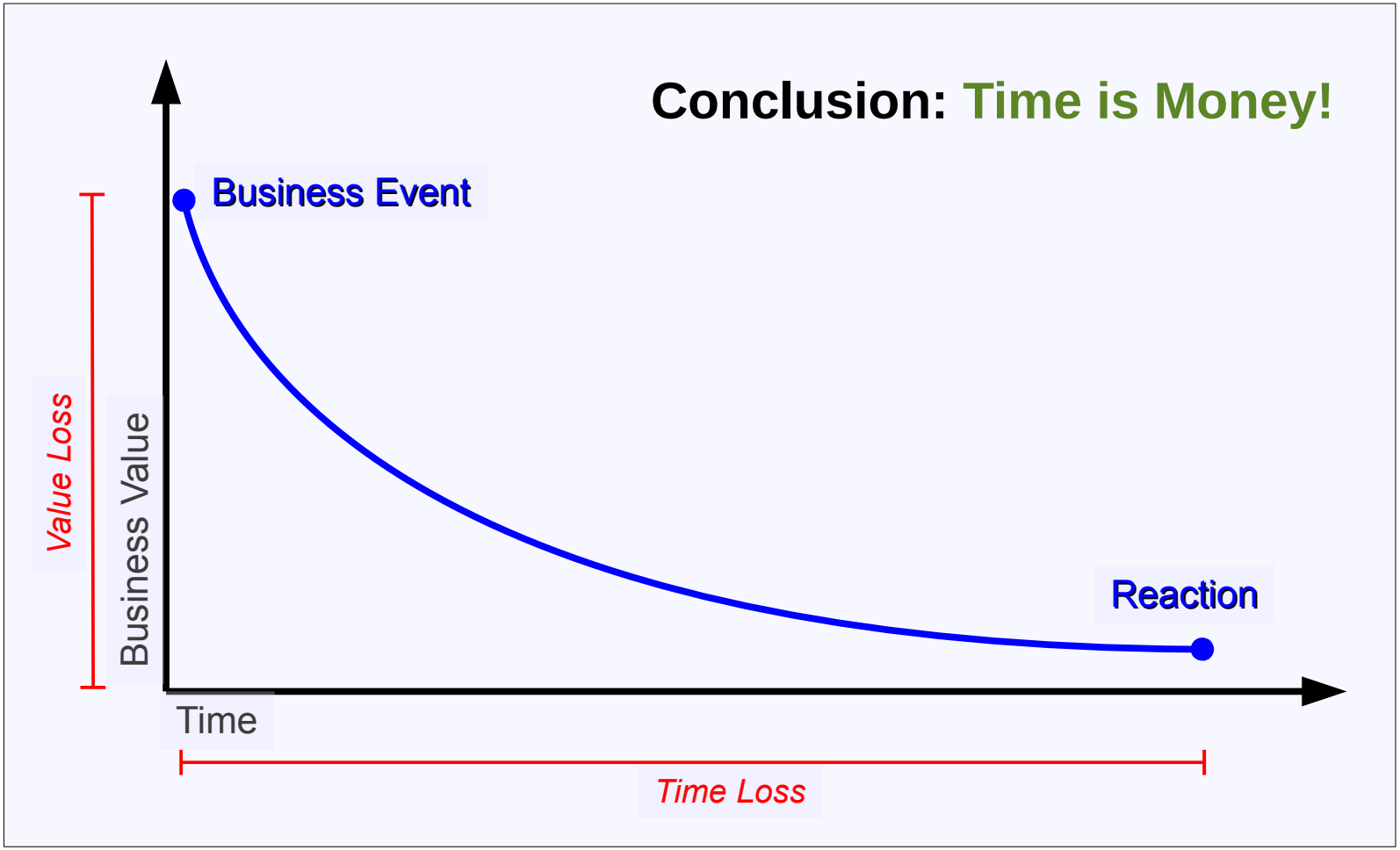


Agenda

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

BRMS adoption goals

Goal #1: Decision Automation

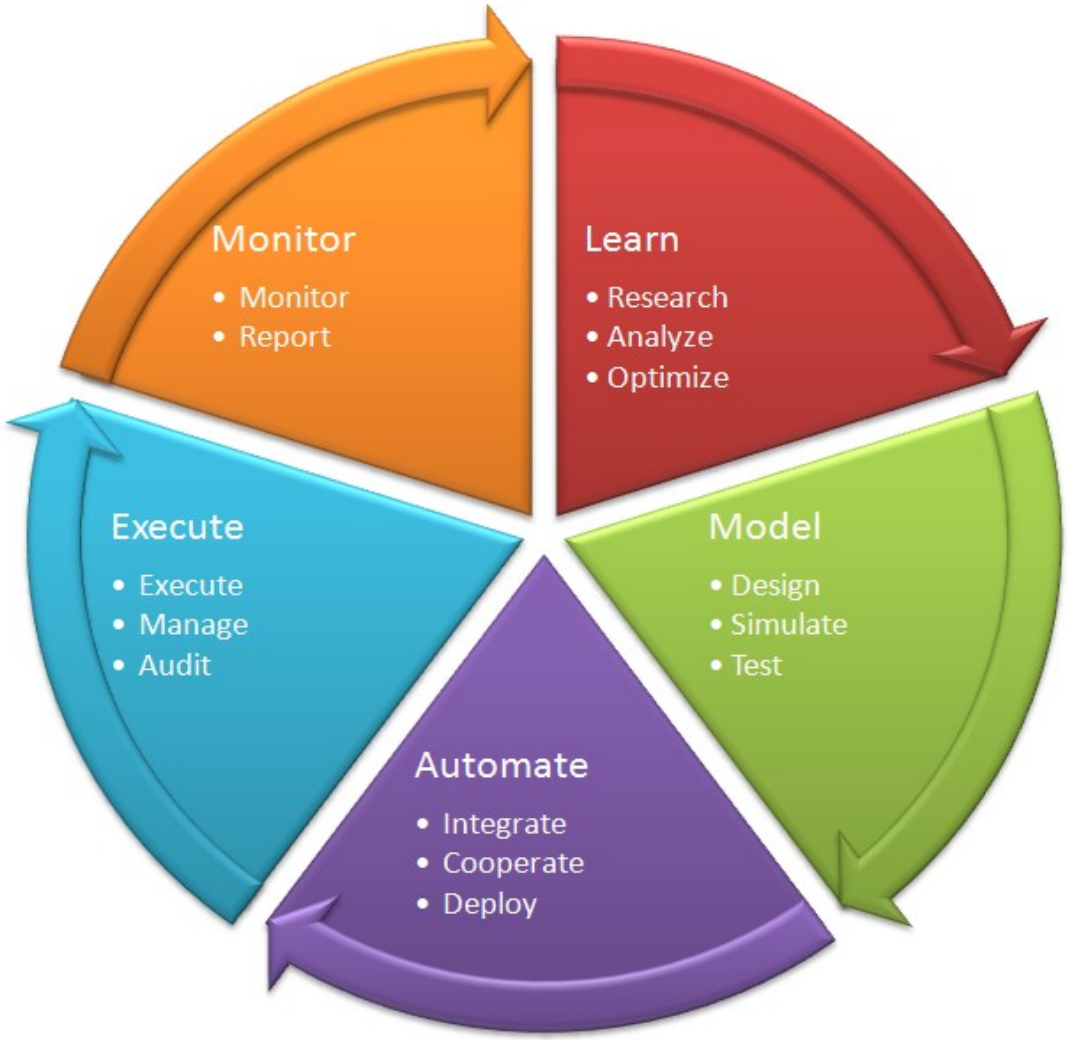


Adapted from a presentation by James Taylor, Sep/2011

Goal #1: Decision Automation

- James Taylor, CEO of DMS:
  - *75%-95% of operational day-to-day decisions can be automated*
- Benefits:
  - *improves consistency and audit-ability*
  - *reduces costs*
  - *speeds processing*
  - *focus staff's expertise*

Goal #2: Independent Lifecycle Management*



* *a.k.a.* ***Business Agility***

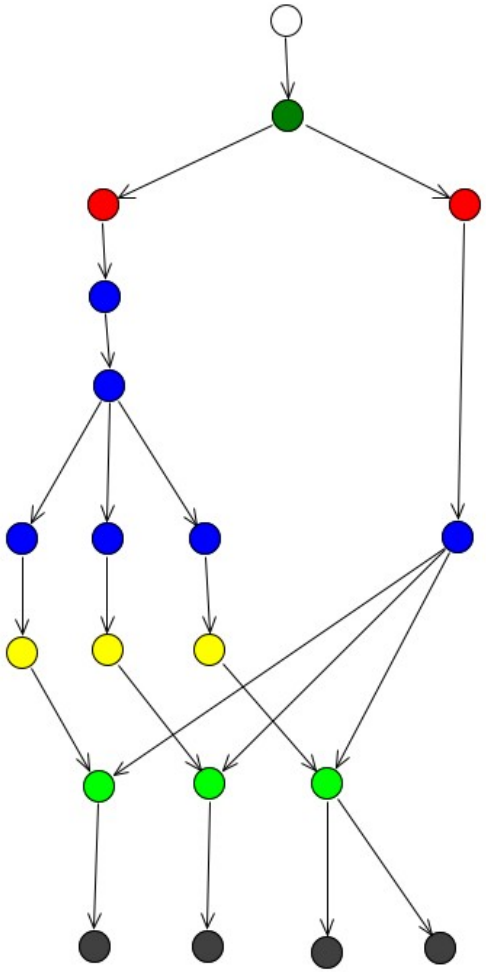
Goal #3: Expressiveness and Visibility

```
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”*

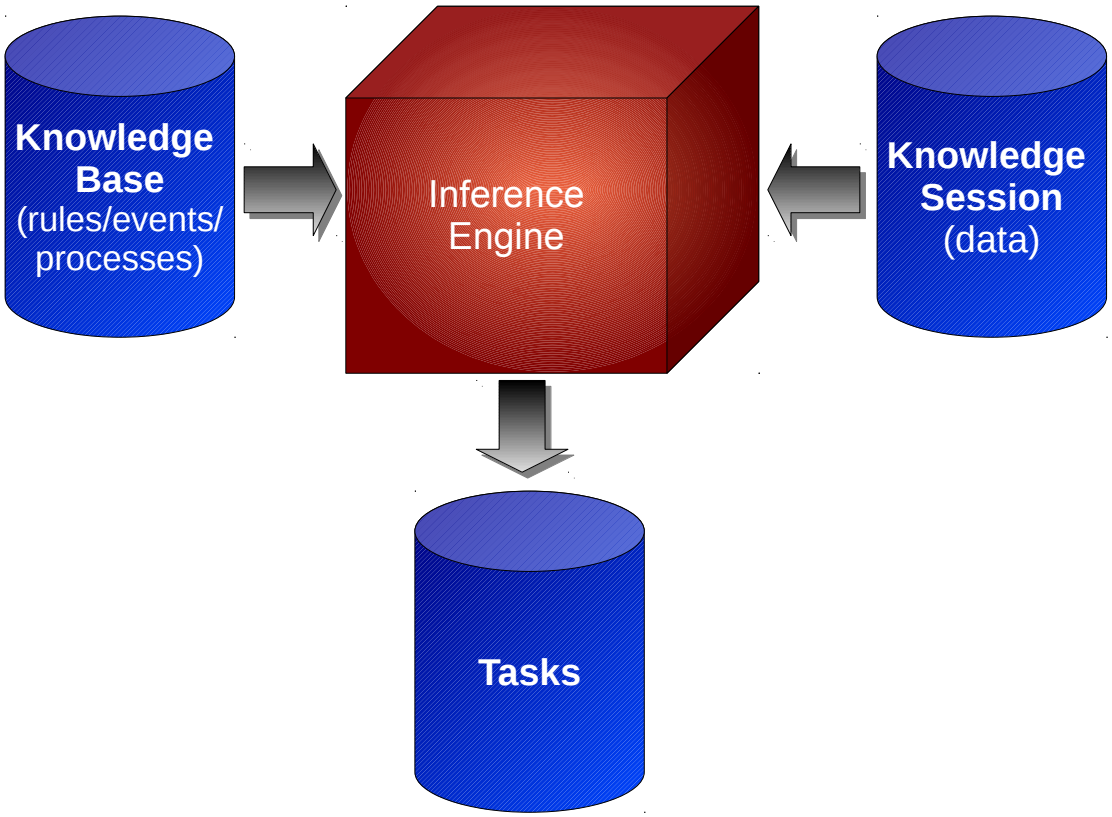
Goal #4: Performance and Scalability

- **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*

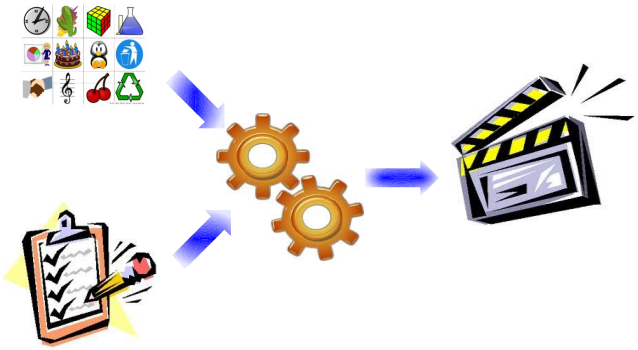


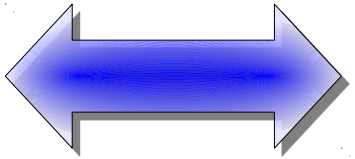
Goal #5..#7: other technical goals

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



BRMS: a simple analogy





Business Rules Management Systems (Rules)

Database Management Systems (Data)

JBoss BRMS: under the hood

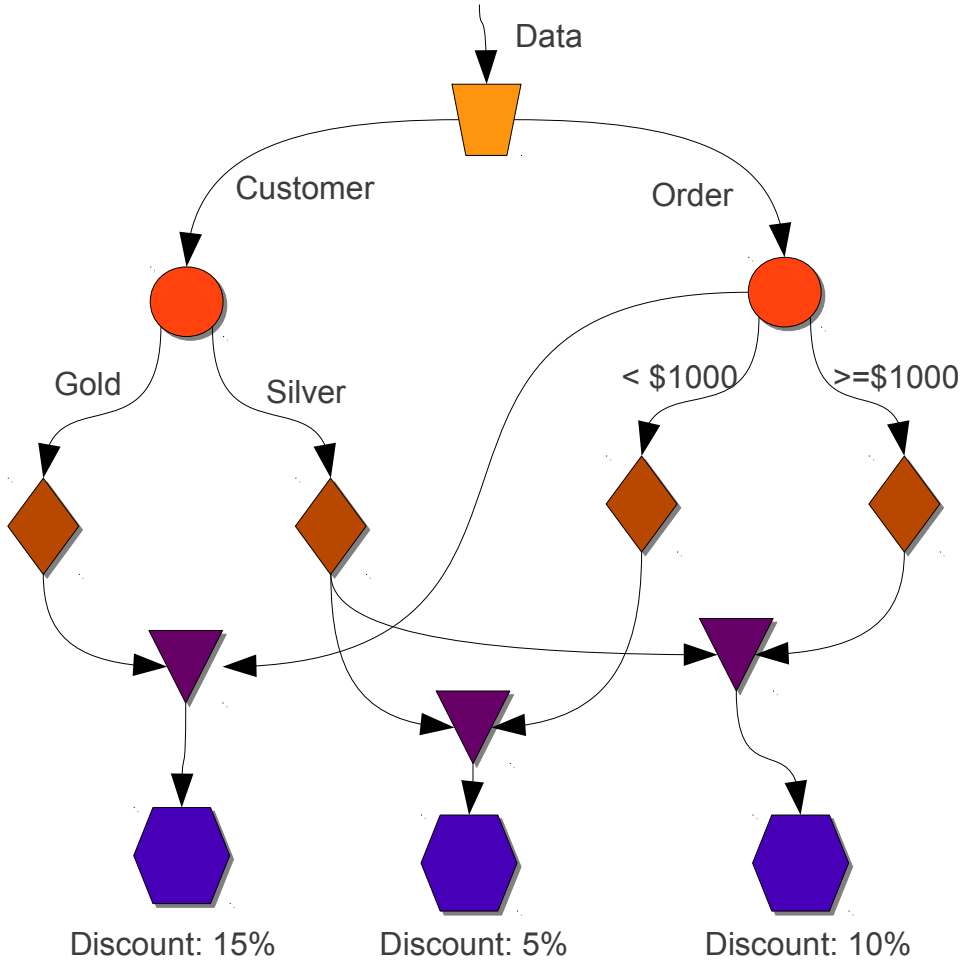
Engine's Algorithm – 30 seconds crash course

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

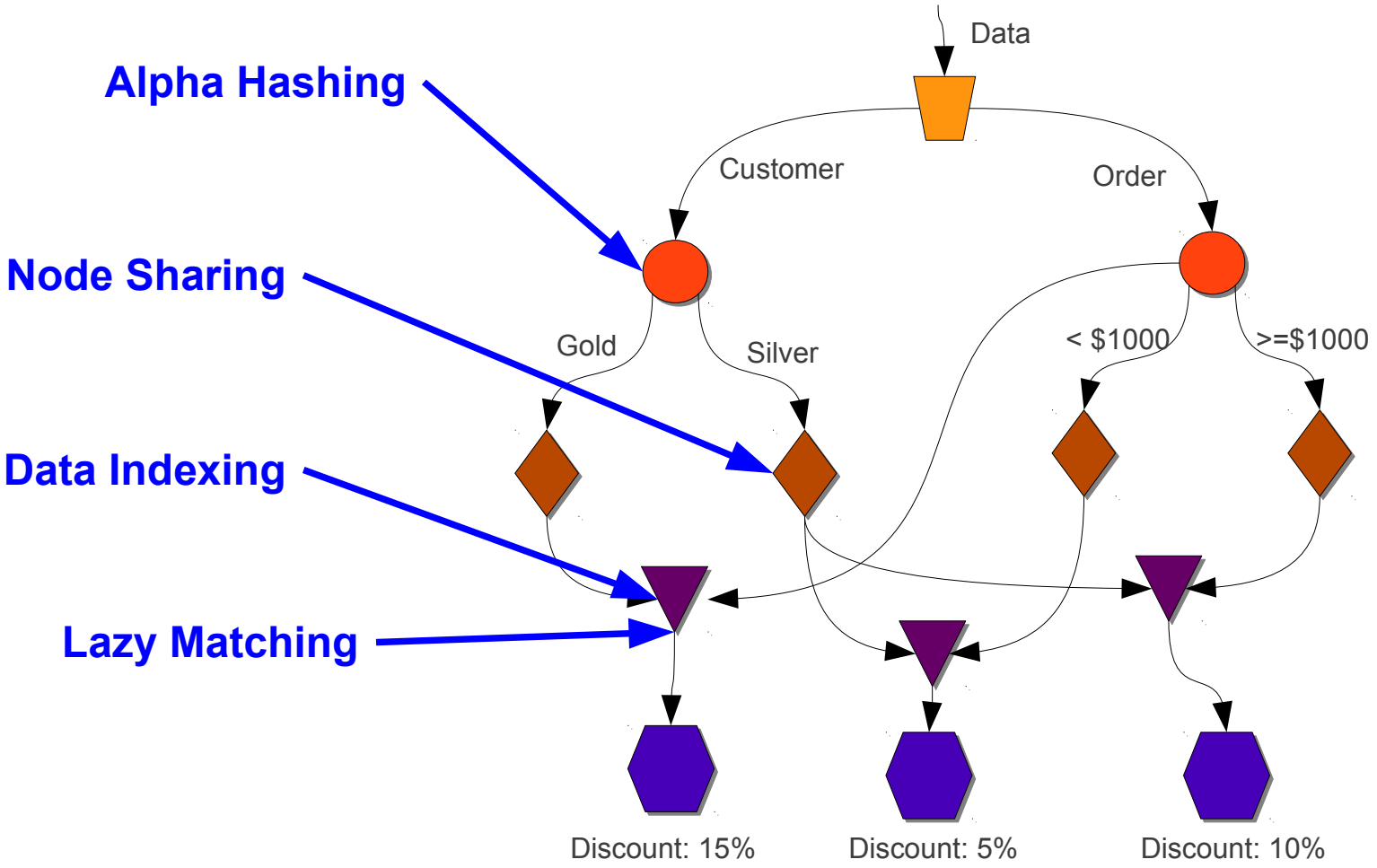


Efficient, Effective

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**

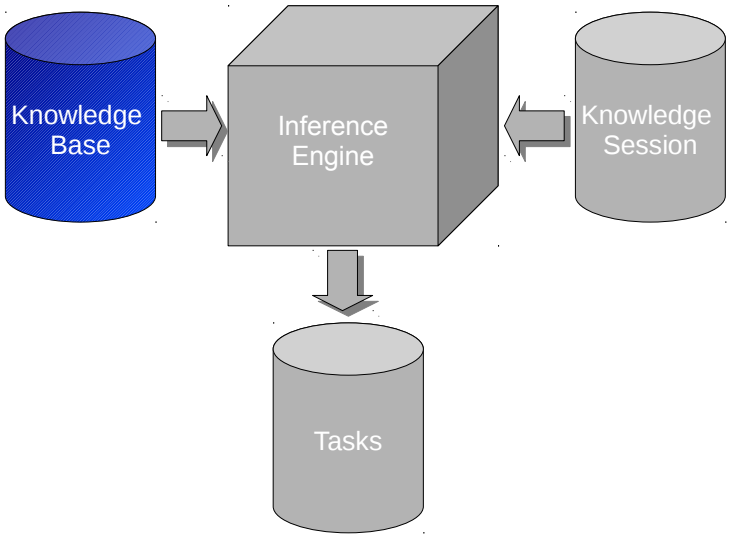
JBoss BRMS – More optimizations



JBoss BRMS – Best Practices

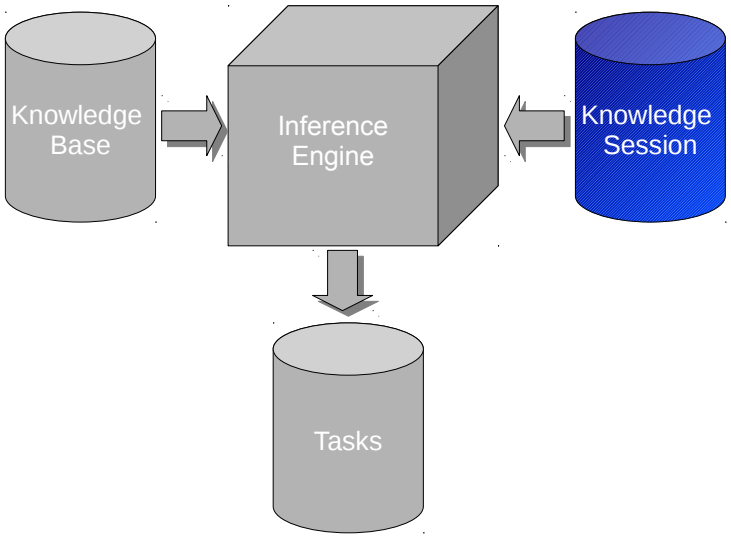
Best Practices – Architecture

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



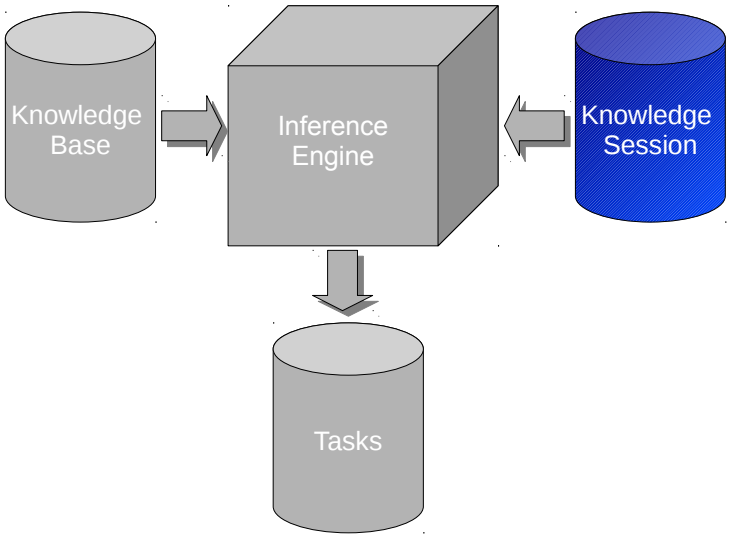
Best Practices – Architecture

- **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



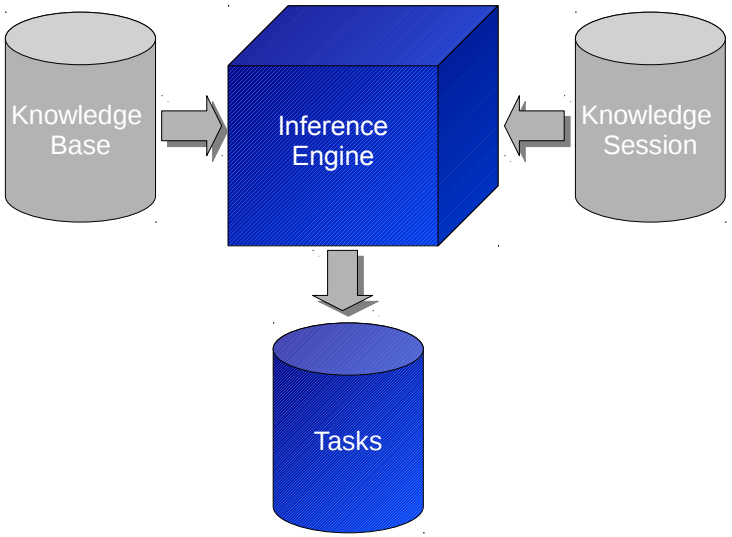
Best Practices – Architecture

- **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



Best Practices – Architecture

- **Know your runtime environment**
  - Resource pooling
  - JVM policies
  - Instrumentation

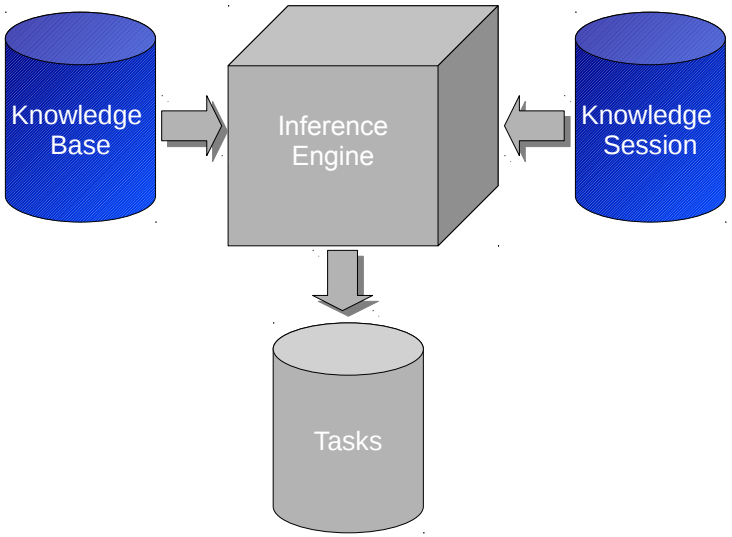


Drools
Business Logic integration Platform /

JBoss BRMS – Best Practices in Rules Authoring

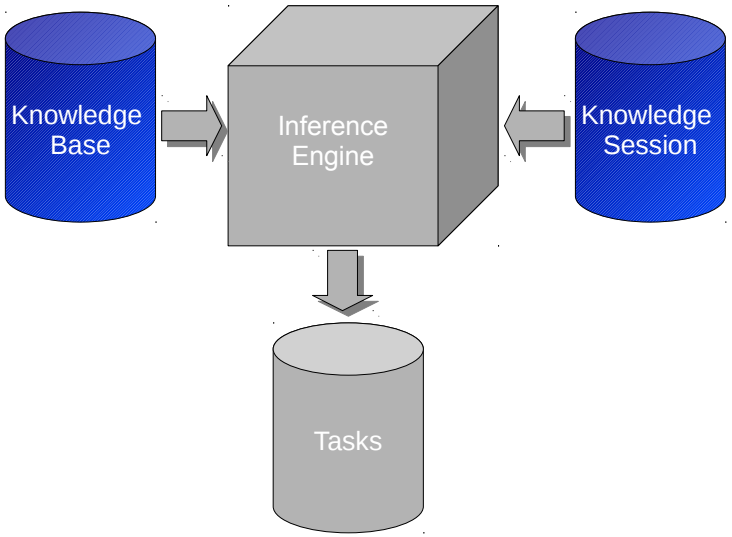
Best Practices – Rules Authoring

- **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
```

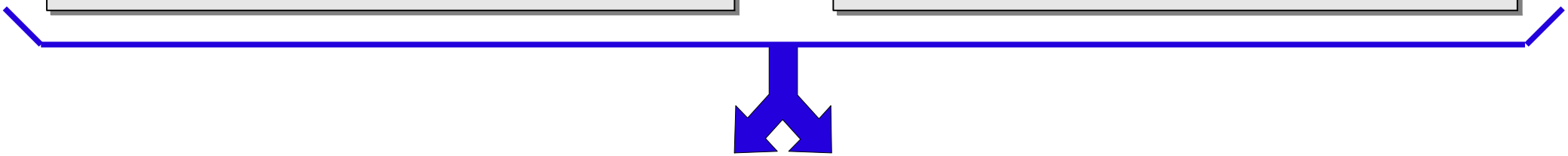
```
rule “2 – Elders can buy tickets in area A”
when
    Person age is greater or equal to 65
then
    Allow sales of area A tickets
end
```

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
```



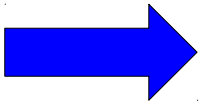
```
rule "1 – Teenagers and Elders get discount"
when
    Teenager or Elder
then
    Assign 25% ticket discount
end
```

```
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

- **Control facts should be used carefully if ever**
  - Breaks expressiveness and visibility goal
  - Breaks “one scenario – one action” best practice
  - Usually leads to micro-control and procedural code

```
rule “X”
when
    ControlFact( enabled == true )
    ...
```


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!"

Q&A

Edson Tirelli
etirelli@redhat.com

LIKE US ON FACEBOOK

www.facebook.com/redhatinc

FOLLOW US ON TWITTER

www.twitter.com/redhatsummit

TWEET ABOUT IT

#redhat

READ THE BLOG

summitblog.redhat.com

GIVE US FEEDBACK

www.redhat.com/summit/survey