JBPM

Rajesh Patel
Software Developer
Partner Harpoon Technologies
rpatel@harpoontech.com

St. Louis Java SIG
August 10th, 2006

Background

- Java Developer for 5 years
- 8 years in software industry.
- Mentoring Java Developers
- Developer for Procelerate Technologies
- Used JBPM to facilitate a business process

Business Process Mgmt

- Process Design
  - No more Programming
- Process Execution
- Process Monitoring
  - Performance
  - Logging
  - Manipulation

State of the Workflow

- Lots of vendors
- Products are prohibitively expensive
- Products require extensive training
- TDD is difficult

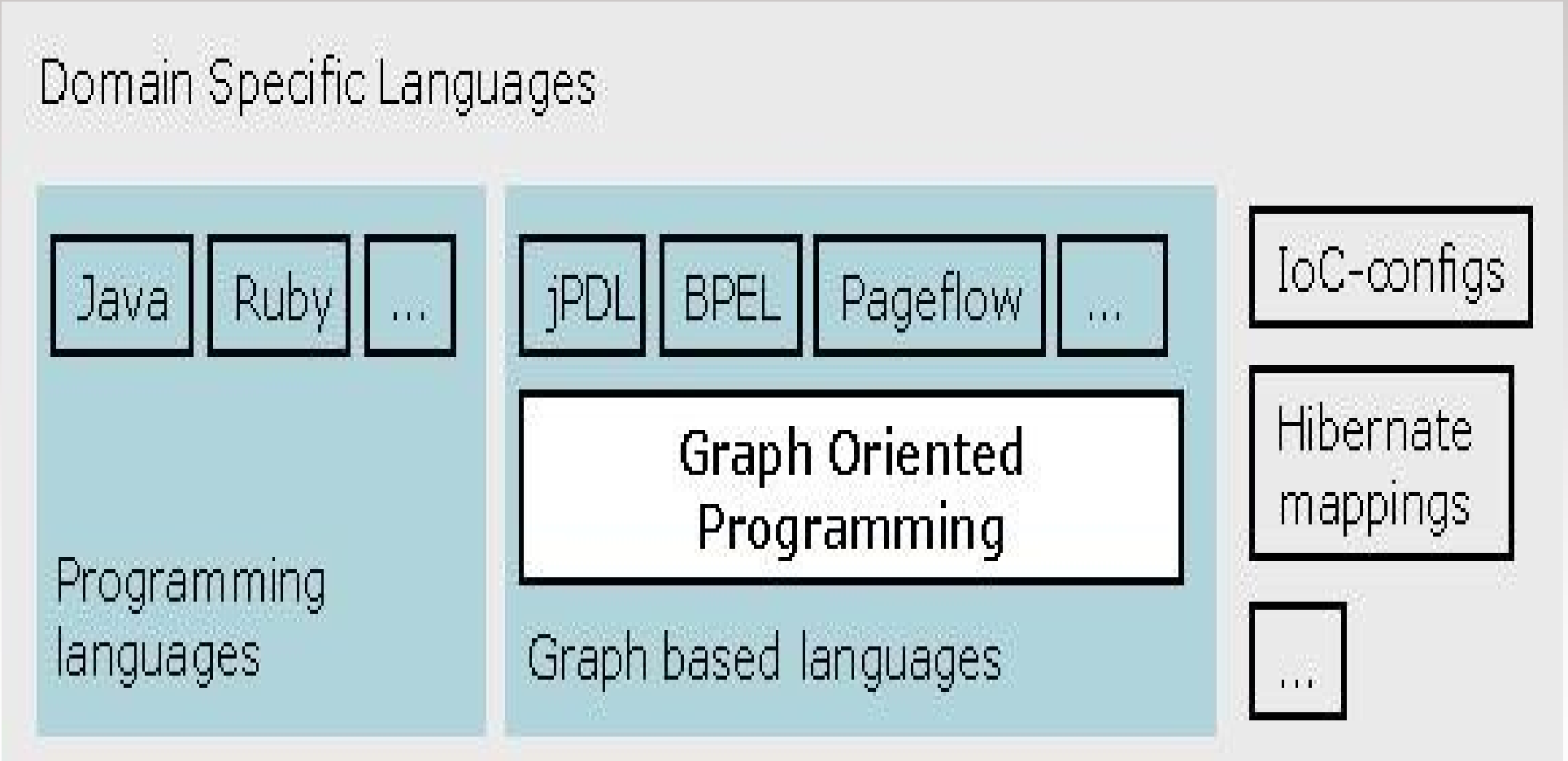
JBPM

- Free/ Open Source
- Developer Friendly
- TDD is very simple
  - Checkout JBPM test suite for proof
- Embeddable into any java application
- Clear Separation between biz analyst and dev.
- Allows a single developer to implement BPM
  - Before only choice was full enterprise BPM

Graph Oriented Programming

- Well Suited for graphical representation
- Support for Wait States

Graph Oriented Programming



JBPM DSL Support

- JPDL – Java based process language
  - Core engine based on JPDL
- BPEL – Web Service process language
  - Separate Download
  - JBPM BPEL License???
- Seam Pageflow
  -

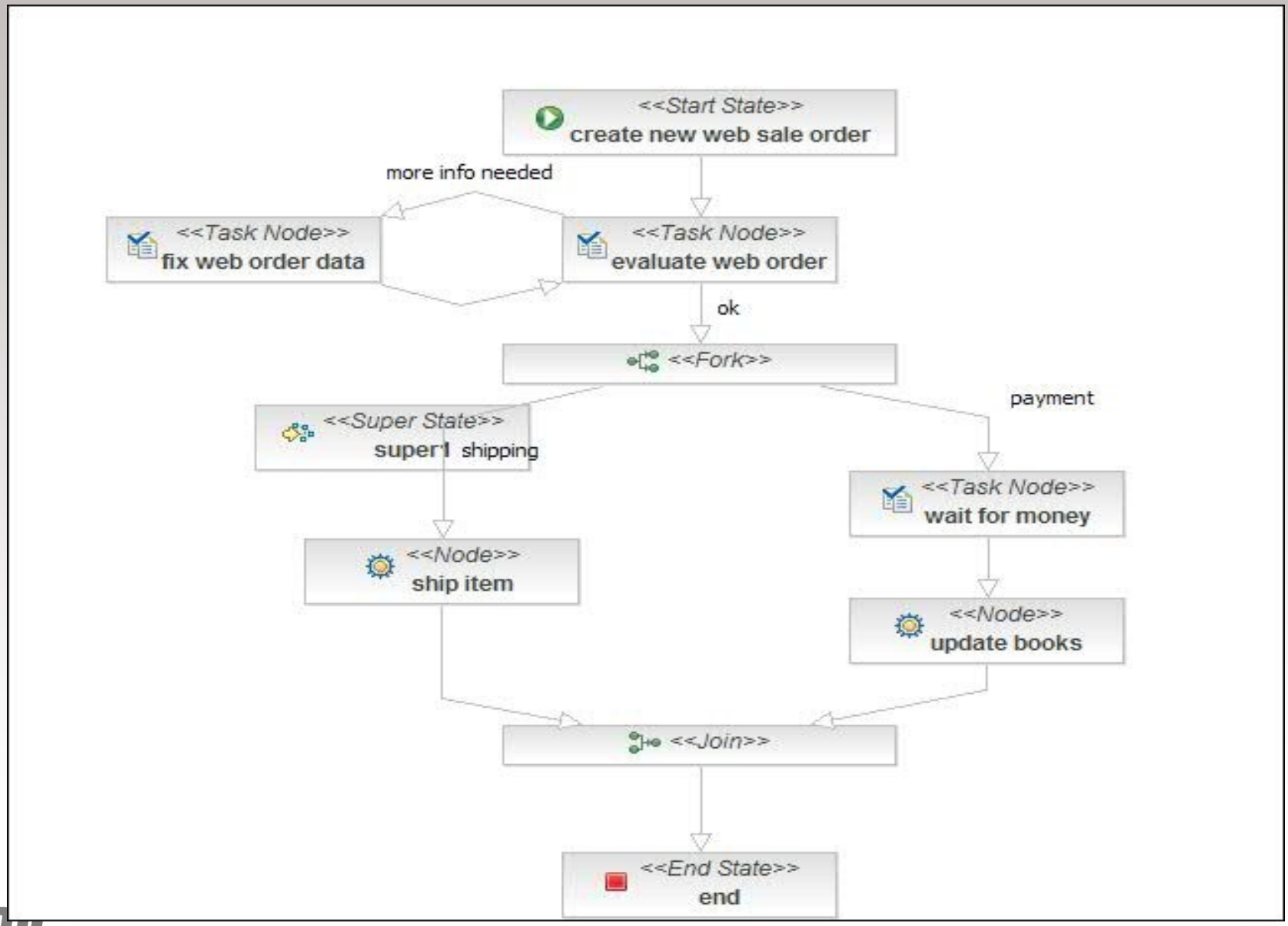
Getting Started

- JBoss IDE
- JBoss Starters Edition
- Use Jboss IDE JBPM Process Wizard

JBPM Node Types

- Start Node
- Task – Requires human intervention
- State - Wait States
- Decision – Decision based on process var
- Fork – Splits execution along multiple paths
- Join – combines multiple execution paths
- Node – Custom Code
- Transitions – Linkages between nodes
- End Node

Example Process



Task Nodes

- Represent a task to be done by a human
- Assigned to a human by a task assignment handler
- Assigned Tasks added to users task list

Sample Process Definition

```
<process-definition name="simple">
   <start-state name="start">
      transition name="toOrder" to="TakeOrder"/>
   </start-state>
   <task-node name="TakeOrder">
     <task name="TakeOrder">
       <assignment actor-id="raj"/>
     </task>
      <transition name="toEnd" to="end"/>
   </task-node>
   <end-state name="end"/>
</process-definition>
```

Node

- Allows custom java code to be executed on node entry
- Custom code must implement an ActionHandler

```
public class CreatePizzaActionHandler implements
ActionHandler {

  public void execute(ExecutionContext executionContext) {
    String item =            (String)
       executionContext.getVariable("item");
    JoptionPane.showInputDialog("Pizza has: " + item);

     executionContext.leaveNode();
  }
}
```

Node Definition

```
   <task-node name="TakeOrder" async="true">
      <task name="TakeOrder">
        <assignment actor-id="bert"/>
      </task>
       <transition name="" to="MakePizza"/></transition>
   </task-node>
   <node name="MakePizza" async="true">
       <action name="MakePizza"
class="org.jbpm.pizza.CreatePizzaActionHandler"/>
      <transition name="" to="Deliver"/></transition>
   </node>
```

Decision

- Allows graph navigation based on process vars
- Condition logic
  - JSF EL
  - Custom java code

Scripted Decision

```
<decision name="delivery">
      <transition name="bike" to="bike">
       <condition>#{address == 'waterman'}</condition>
      </transition>
      <transition name="car" to="car">
       <condition>#{1==1}</condition>
      </transition>
</decision>
```

Decision handler

```
<decision name="delivery">
   <handler
    class="org.jbpm.tutorial.action.DeliveryDecisionHandler" />
   <transition name="bike" to="bike"/>
   <transition name="car" to="car"/>
</decision>
```

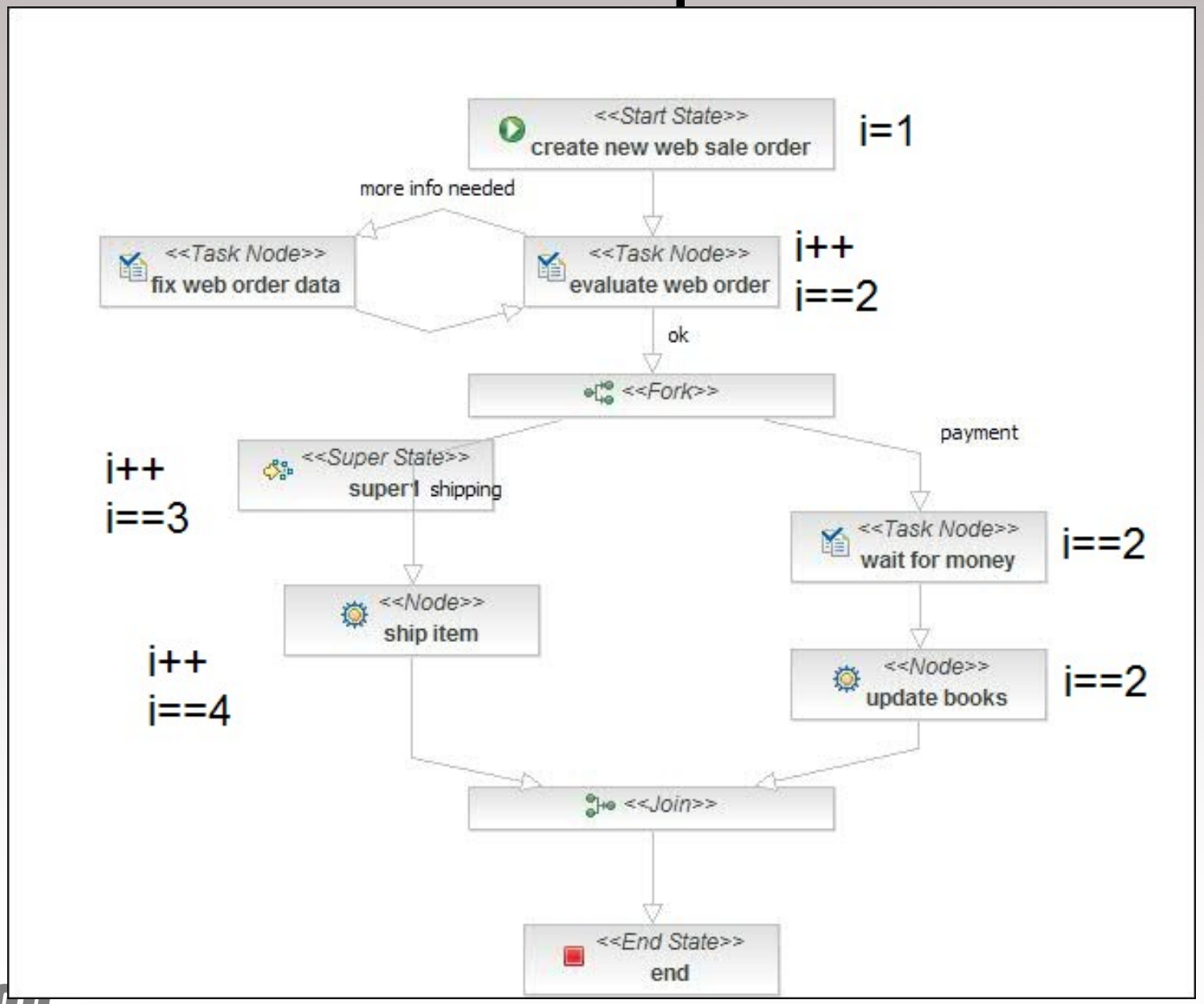
```
public  class DeliveryDecisionHandler implements DecisionHandler {
    public String decide(ExecutionContext executionContext) throws
Exception {
       String address =
(String)executionContext.getVariable("address");
       if((address.indexOf("Waterman") == -1)  &&
              (address.indexOf("waterman") == -1) ){
          return "car";
       } else{
          return "bike";
       }
    }
}
```

fork

- Allows the execution path to split

```
<fork name="salefork">
  <transition name="chicken"  to="Make chicken" />
  <transition name="mexican" to="Make mexican" />
</fork>
```

Process Scoped Varibles



join

- Used to combine execution paths
- All nodes with transitions to join node must complete

```
<join name="salejoin">
  <transition to="end" />
</join>
```

JBPM API?

Spring Integration

- Provided through spring modules project
- Allows dependency injection on actions
- https://springmodules.dev.java.net/docs/reference/0.5/html_single/#jbpm31

References

- http://www.jboss.com/pdf/jbpm_whitepaper.pdf
- http://docs.jboss.com/jbpm/v3/userguide/

Questions?

- E-mail: rpatel@rajix.com