

- » BOOST PERFORMANCE
- » REDUCE COST
- » INCREASE AGILITY
- » ENHANCE CRM
- » SHORTEN TIME TO MARKET
- » DRIVE INNOVATION
- » IMPROVE EFFICIENCY
- » INCREASE ADAPTIVITY
- » ENABLE BUSINESS TRANSPARENCY
- » ENSURE REGULATORY COMPLIANCE



CONSULTING > SOLUTIONS > OUTSOURCING

jBPM

Practical opensource workflow and BPM

Jos Dirksen

Ede, 15th of June, 2006

Agenda



1. **Introduction** (+/- 3 min)
2. Basic concepts and ideas behind jBPM (+/- 22 min)
3. Advanced jBPM topics (+/- 20 min)
4. Summary and question time (+/- 5 min)

- BPEL is a clumsy way to support Business Process Management -

Tom Baeyens, Lead Developer



What is BPM, and why do we need it?

Workflow: *The automation of a business process, during which documents (information, issues, tasks, work orders, bug/defect reports etc.) are passed from one state to another for action, according to a set of rules defined by your workflow scheme.*

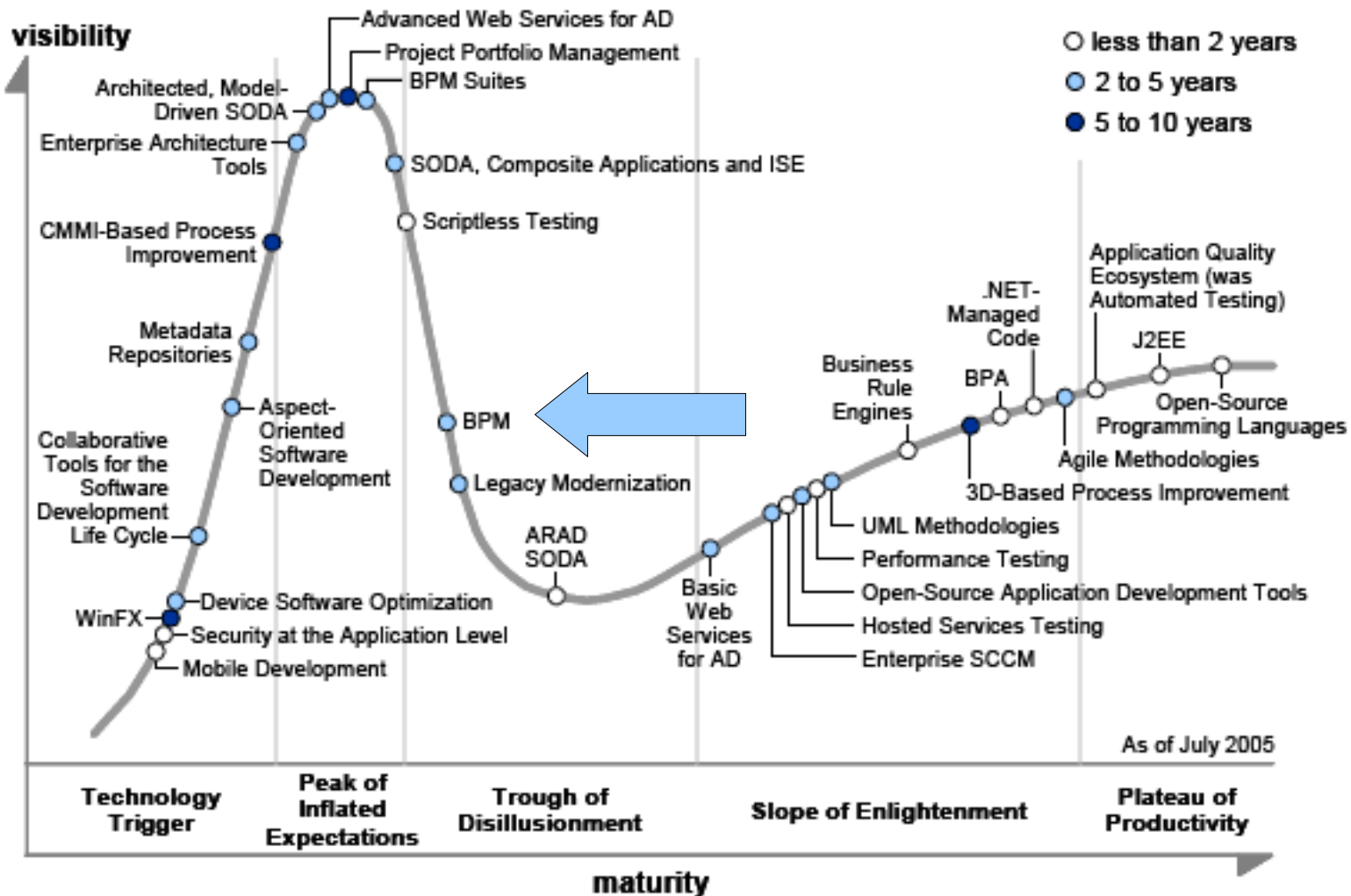
BPM: *The term Business Process Management (or BPM) refers to a set of activities which organizations can perform to either optimize their business processes or adapt them to new organizational needs.*

- Long running processes:
 - Wait states
 - Execution needs to be persisted
 - How do you persist a java thread?

- Graphical representation:
 - Common language
 - Easy translation from requirements to design

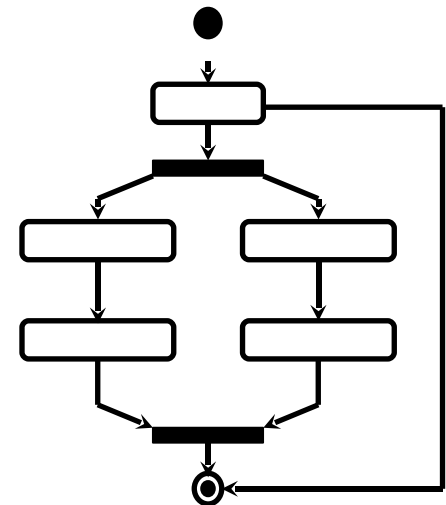
Where is BPM at the moment?

Figure 1. Hype Cycle for Application Development, 2005



What is jBPM?

- jBPM: Java Business Process Management
- Started in March 2002 by Tom Baeyens
- December 2003: release 1.0; current version: 3.1.1 (released 13/04/2006)
- October 2004, acquired by JBoss
- Features:
 - Based on *Graph Oriented Programming (GOP)*
 - Supports Task Management
 - Clean java integration
 - Timer service
 - Runs on any DB, any application server
 - Can be run standalone, embedded, webapp or .ear



Agenda



1. Introduction (+/- 3 min)
2. **Basic concepts and ideas behind jBPM (+/- 22 min)**
3. Advanced jBPM topics (+/- 20 min)
4. Summary and question time (+/- 5 min)

- BPEL is a clumsy way to support Business Process Management -

Tom Baeyens, Lead Developer



GOP: Nodes and transitions

- Based on the execution of directed graphs
- Doesn't focus on a specific execution language
- A graph has `Nodes` and `Transitions`



directed graph

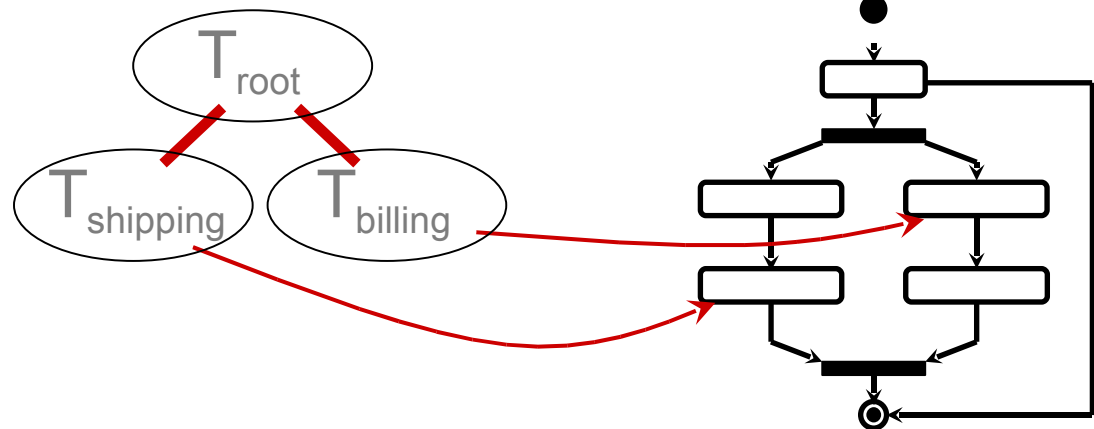
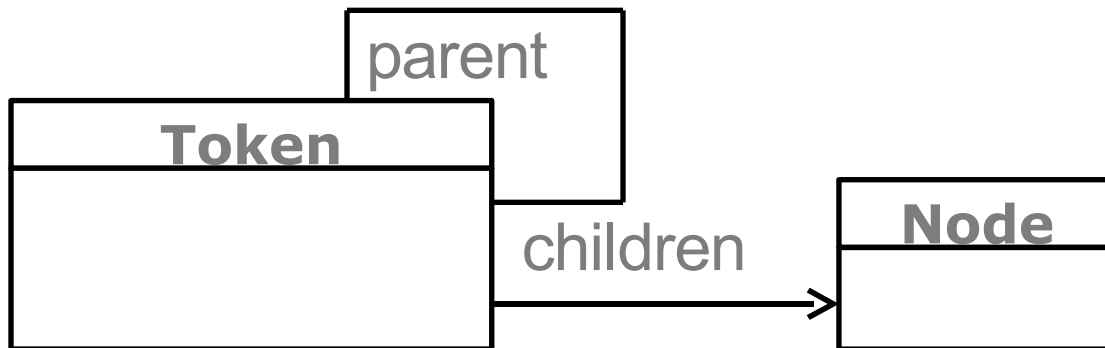


- A graph in which each graph edge is replaced by a directed graph edge, also called a digraph -

- **Transition:**
 - a node has leaving and arriving transitions
- **Node:**
 - A node is a command and has an execute method
 - Can be a 'Wait state' or a piece of Java code

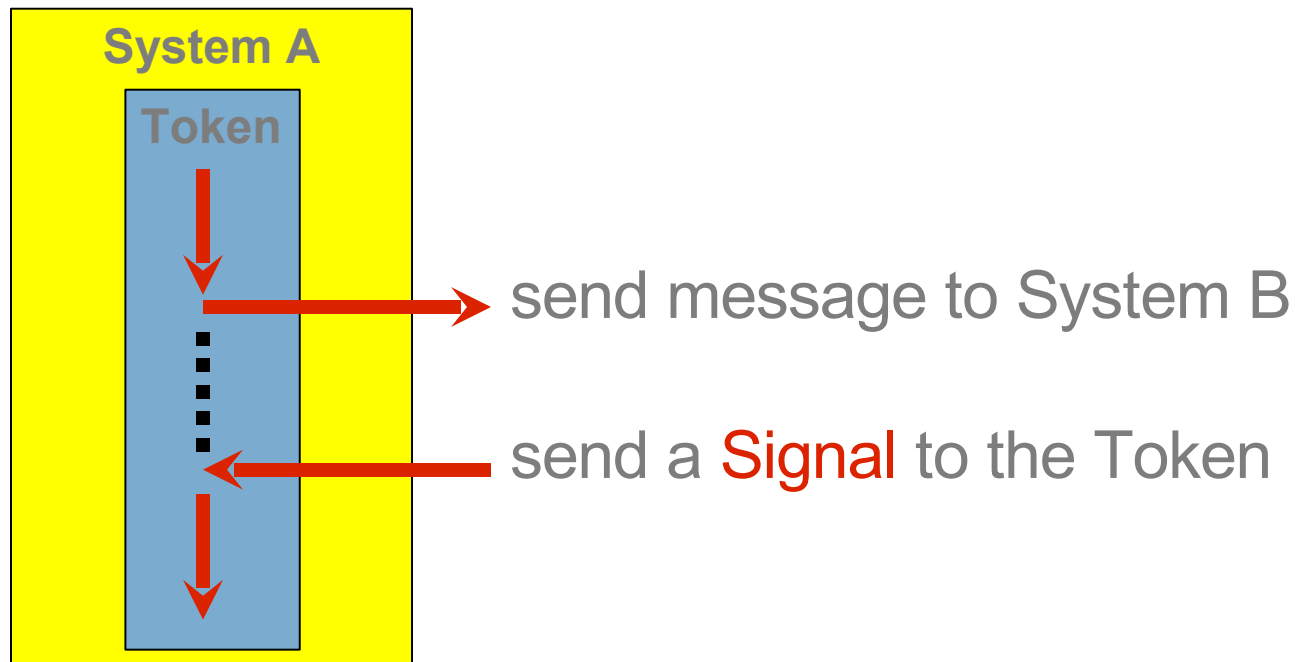
GOP: Token

- **Token:** a pointer to where you are in the proces
- Modelled as normal Java classes.



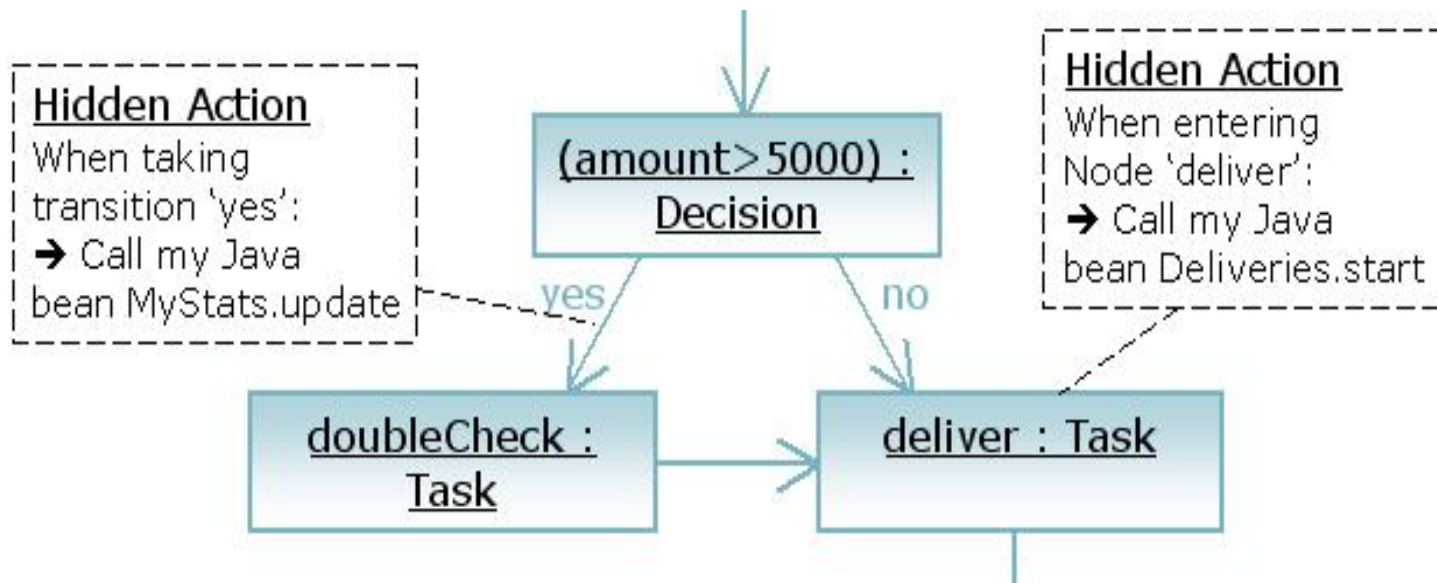
GOP: Signalling a Token

- How to continue from a wait state?



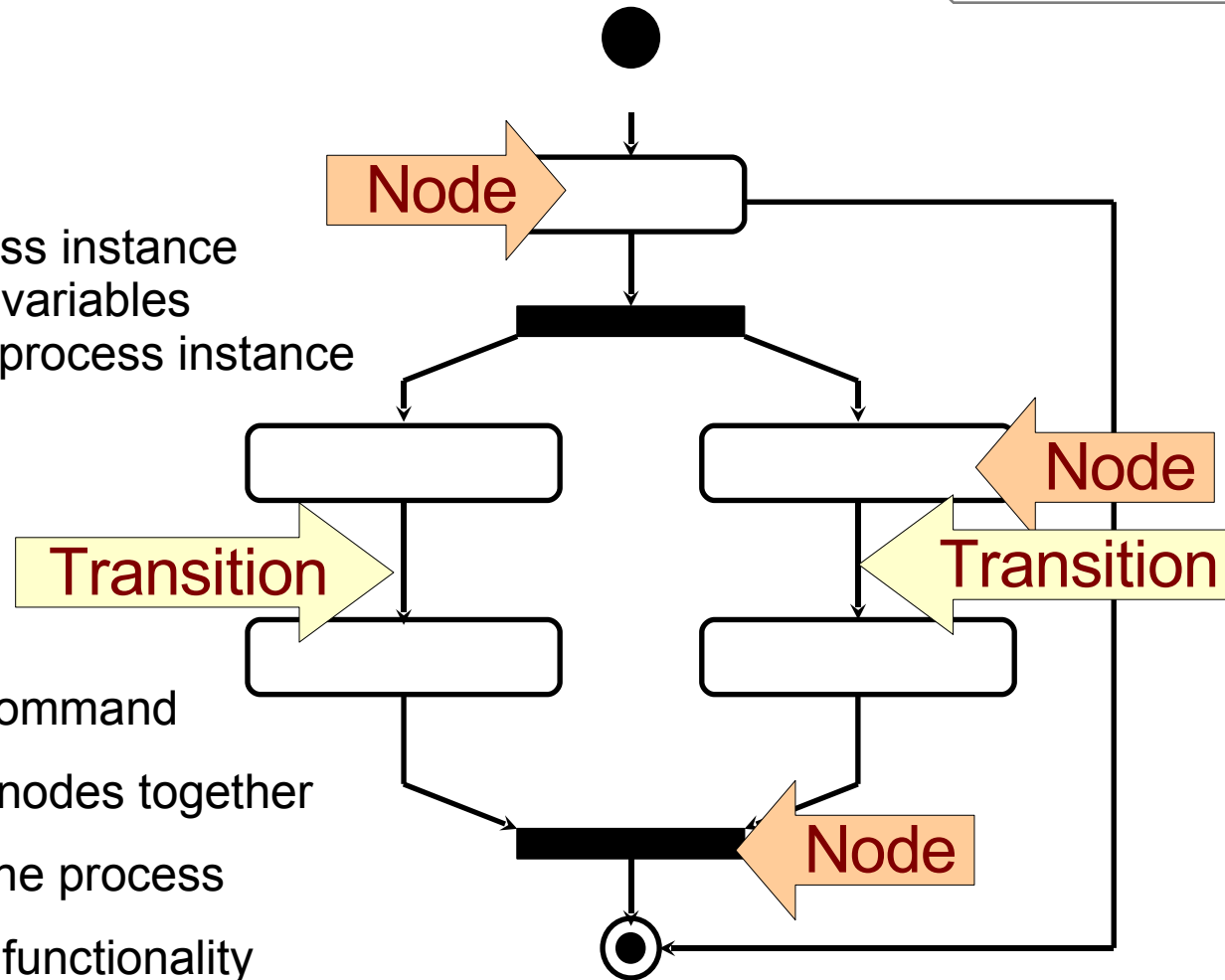
GOP: Actions

- **Action:**
 - node-enter, node-leave, before-signal, after-signal



GOP: Process Context and summary

- **Context:**
 - Scope is a single process instance
 - Read / Write access to variables
 - Can be accessed from process instance



- **Node:** execute a certain command
- **Transition:** connects two nodes together
- **Token:** where are you in the process
- **Action:** execute technical functionality

Demo 1. Implement a basic business process

- Company sells party articles for certain sporting events.
- Based on the outcome of certain matches, extra party articles are ordered
- If the match is lost, everything is sold with a very big discount.



Shows: tasks, decisions, action handlers, users, token



Agenda








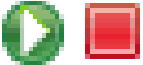


1. Introduction (+/- 3 min)
2. Basic concepts and ideas behind jBPM (+/- 22 min)
3. **Advanced jBPM topics (+/- 20 min)**
4. Summary and question time (+/- 5 min)

- BPEL is a clumsy way to support Business Process Management -

Tom Baeyens, Lead Developer



All the available nodes

-  **Decision:** models a decision made by the process
-  **Fork and Join:**
 - Fork splits the current path of execution.
 - Join puts them back together.
 - Allows for multiple concurrently executing tokens
-  **Node:** if you want to create nodes based on your custom code
-  **Task Node:** used for human actions (more on this later...)
-  **Process State:** allows state composition
-  **Start and Stop State:** start and end point of the process
-  **State:** basic wait state, used for communicating to external systems
-  **Super State:** can be used to group states together

Tasks and assigning them to users

➤ **Task:**

- Is a wait-state for a human action.
- Multiple tasks per node are allowed.
- Path of execution (token) normally can't leave until task is complete
 - Can be customized: *last, last-wait, first, first-wait, unsynchronized, never*

➤ **Assignment:**

- Specify who will execute this task:
- Can use the Identity component: eg. user(Jos)--> group(admin)

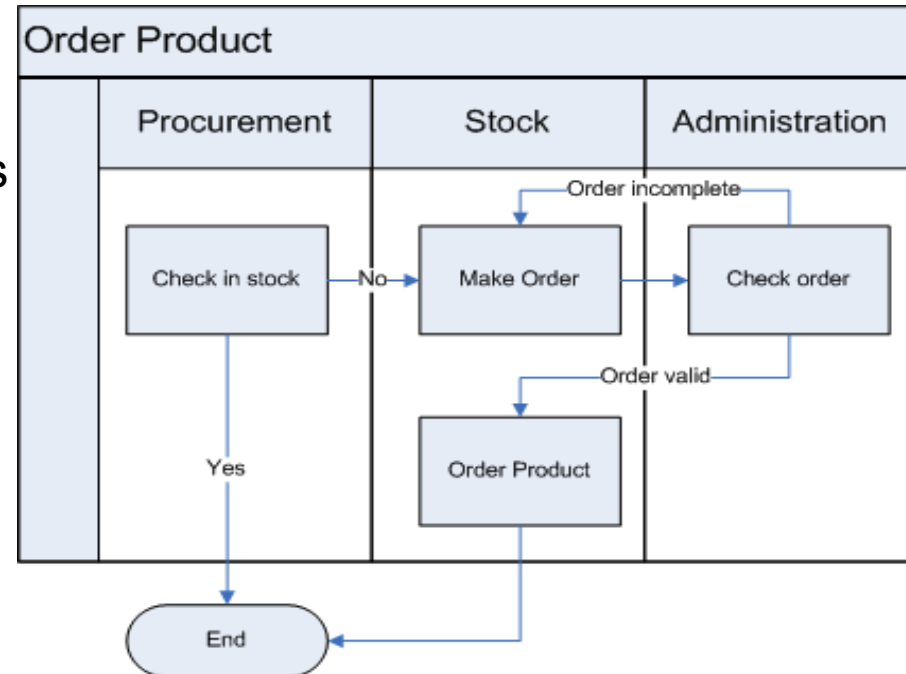
```
<task name="Enter the match results">
  <controller>
    <variable name="weWon"
              access="read,write,required" />
  </controller>
  <assignment expression="user(Jos)" />
</task>
```

Swimlanes

- A swimlane is a process role.
- Indicates that multiple tasks in the process must be done by the same user

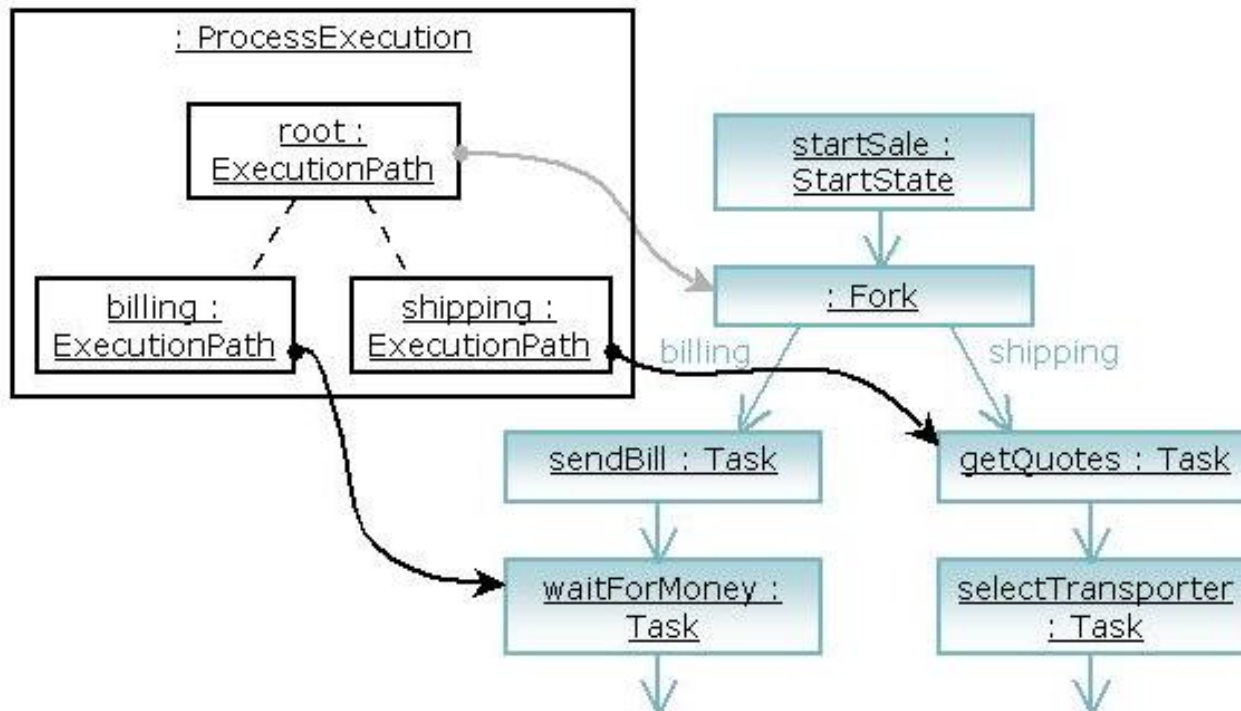
```
<swimlane name="Stock">
  <assignment
    expression="user (Jos) " />
</swimlane>
```

```
<task name="make order"
  swimlane="Stock">
  <controller>
    ...
  </controller>
</task>
```



Fork and Join

- Allows for concurrent executions
- A concurrent execution does **NOT** imply a multithreaded environment.
- State transitions are instantaneously. The process will always be in a stable state



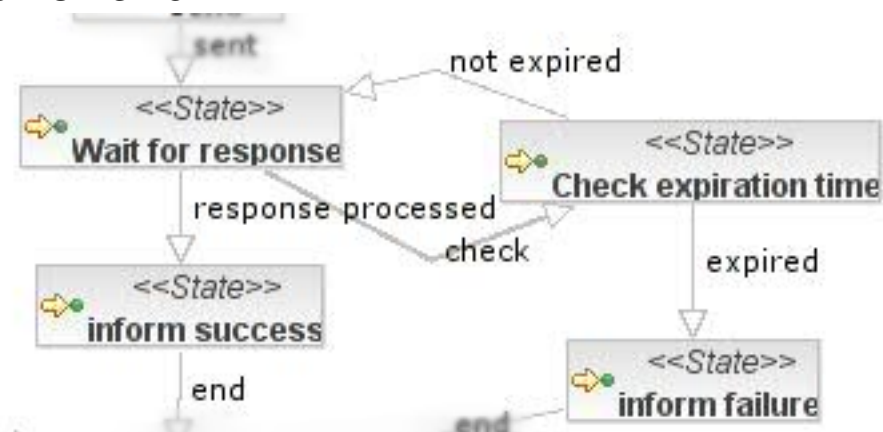
Timers and schedulers

- Some nodes have a certain time within they need to be executed.
- jBPM provides a timer construction to deal with this.
- Timers are executed by a separate Scheduler thread.

```

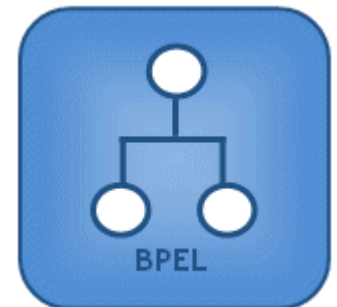
<state name='catch crooks'>
  <timer name='remnder'
    duedate='3 business hours'
    repeat='10 business minutes'
    transition='time-out-transition' >
    <action class='action'/>
  </timer>
</state>

```



BPEL and jPDL: Short summary of BPEL

- Business Process Execution Language
- Standardization at OASIS
- XML language for describing long-running interactions of web services
- Standard language for integrating Web Services
- Focuses on orchestration *across* platforms
- Of course requires services to be exposed as Web Services
 - IBM & BEA are working on BPELJ for integration with J2EE
- Support from IBM, BEA, Microsoft, Sun and more...



BPEL and jPDL: main differences

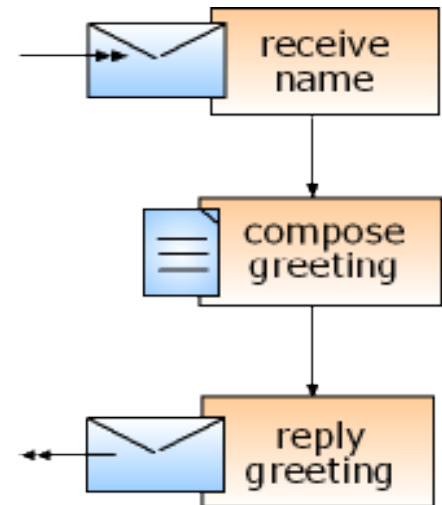
- Fixed set of XML constructs **versus** set of nodes that allow custom functionality
- BPEL can declare links (**AND / OR**) between parallel executing processes
- jPDL variable context is based on POJOs, BPEL's one is focussed on XPath
- Interaction with the process: BPEL uses webservices, jBPM uses jBPM API
- jPDL calls services using ActionClasses in Java, BPEL uses webservices.

- Which one to chose:
 - **BPEL**: heterogenous systems, XML based messages, Interoperability
 - **jPDL**: Rich workflow constructs, Java based environments, Lightweight

- Whatever you chose, jBPM supports both -

Demo 2: Using BPEL with jBPM

- At the moment this is a bit hard to do!
 - 1) Start with a bpel file. `hello.bpel`
 - 2) Package and deploy the BPEL process.
 - 3) Generate and create the service side bindings.
 - 4) Generate and create the client side bindings.
 - 5) Generate and deploy the enterprise application
 - 6) Finally run a JUnit test to check the process



- jBPM-BPEL provides a good introduction to BPEL-

Agenda



1. Introduction (+/- 3 min)
2. Basic concepts and ideas behind jBPM (+/- 22 min)
3. Advanced jBPM topics (+/- 20 min)
4. **Summary and question time (+/- 5 min)**

- BPEL is a clumsy way to support Business Process Management -

Tom Bayens, Lead Developer



- **Solid foundations:**
 - Graph Oriented Programming
 - Task management
 - Lightweight java integration
 - Timers and schedulers
 - Graphical development environment

- **Easy integration:**
 - Uses hibernate to integrate with different DBs
 - Spring support through spring modules (<https://springmodules.dev.java.net/>)
 - Works with POJOs and JEE
 - Can be deployed as standalone, Web application or .ear

- **Flexible:**
 - Support multiple definition languages: jPDL and BPEL
 - Easy to extend and add custom functionality

Questions ?



- » BOOST PERFORMANCE
- » REDUCE COST
- » INCREASE AGILITY
- » ENHANCE CRM
- » SHORTEN TIME TO MARKET
- » DRIVE INNOVATION
- » IMPROVE EFFICIENCY
- » INCREASE ADAPTIVITY
- » ENABLE BUSINESS TRANSPARENCY
- » ENSURE REGULATORY COMPLIANCE

Atos TM Origin

CONSULTING > SOLUTIONS > OUTSOURCING

More information:

Jos Dirksen
jos.dirksen@atosorigin.com

<http://www.jbpm.org>