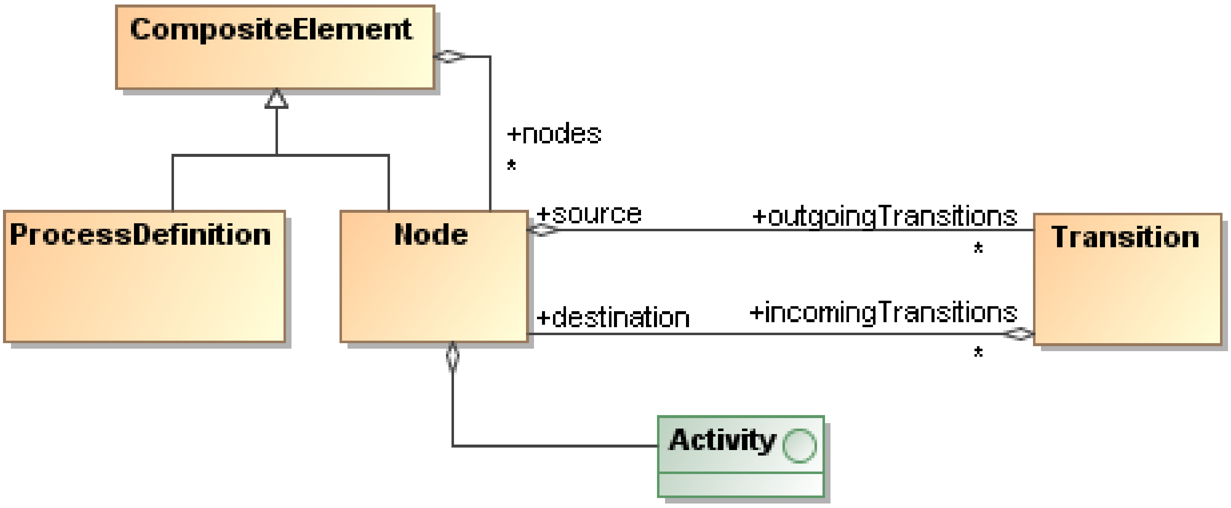
The Process Virtual Machine versus jPDL

The Process Virtual Machine

The Process Virtual Machine is the Java framework for building executable process graphs. It defines the classes for creating a graph structure (**`ProcessDefinition`**) and it defines the classes that represent the runtime execution state (**`Execution`**) of one execution of such a process definition.

Some process languages are based on free graph process models, while others are based on a composite structure. The process structure of the Process Virtual Machine supports both.

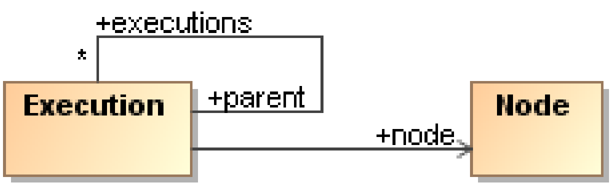


Each node has an **`Activity`** associated to it. That activity represents the runtime behavior of the node. The process structure delegates the runtime execution of the node to the activity.

```
public interface Activity {
  void execute(ActivityExecution execution) throws Exception;
}
```

Because the process model supports composition and graph based process structures, any process construct in any process language can be translated to a node and activity without compilation. As an alternative, processes might also be translated into another process structure. For example, a composite process might always be translated into an executable graph structure. This is how we implemented BPEL on jBPM 3. Though in those cases, the direct relation between the process construct and the activity implementation is lost.

An **`Execution`** represents one path of execution. An execution maintains a pointer to the current node in the process graph. To represent concurrent paths of execution, the execution has a list of child executions that results into a tree structure. This tree structure boils down to the minimal complexity if you have just one path of execution cause you're working with an execution directly. And it can scale to complex scenarios to handle all forms of concurrency in case that is needed.



As you might have concluded, this model is based on interpretation of the process graph at runtime. The activity can invoke methods on the execution (ActivityExecution) to take transitions or to execute child nodes. Then the executions will interpret the process structure and propagate the execution to the next node and then delegate to that activity. Interpretation is not the only form of making executable processes. Alternatively, a process definition in a given process language could be compiled into e.g. plain Java code. That approach could be faster, but as long as the process definition that is interpreted is kept in memory, the interpretation approach is not significantly slower. This interpretation approach is much more flexible then a compilation approach.

Another advantage of interpretation is that the actual implementation of the execution can be customized. Each process language might overwrite and customize the methods that perform the actual process interpretation like the take transition and execute child node methods.

jPDL

jPDL is a process language implementation based on the Process Virtual Machine that exposes the Process Virtual Machine concepts directly. Hence it fits nicely onto the PVM. jPDL defines a number of concrete process language constructs, an XML schema and a process archive format to express processes. Typically the jPDL process constructs have a tight relation to Java. As Java is taken as the hosting environment for the Process Virtual Machine and the jPDL process language.

This close relation to Java provides makes it easy for developers to leverage jPDL for state management and perform the rest of the work in a project in Java. Many process languages don't have such a gradual flexibility. With jPDL, you can start with managing the state of an overall process. Then, as we include more node types out of the box, developers can leverage Java capabilities in an easy way. But in each process, there are situations where more complex things need to be done then what is convenient to do with process constructs in the process itself. In those cases, developers need to be able to link programming logic into the process easily. In that respect, Java is a much more powerful environment to link to then e.g. a WSDL service environment like BPEL does. With XML technologies, information processing is much harder.