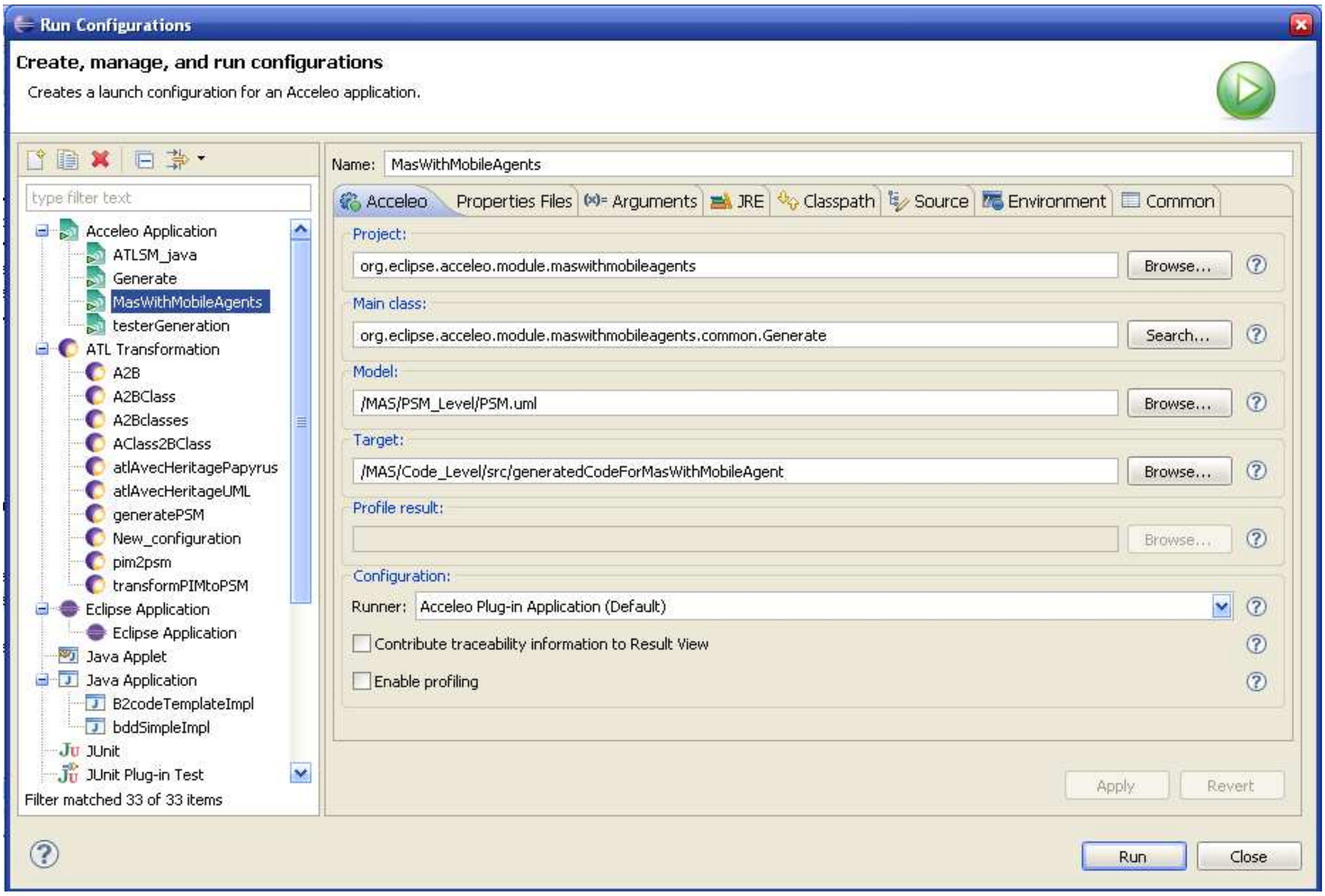
JavAct code generation with Acceleo

The rules for code generation from a PSM to JavAct have been written using Acceleo and summarized in a file generate.mtl. Figure 1 shows the configuration of Acceleo/Eclipse used for the execution of this file (which is given below the figure).



Run Configurations

Create, manage, and run configurations

Creates a launch configuration for an Acceleo application.

Name: MasWithMobileAgents

Project: org.eclipse.acceleo.module.maswithmobileagents

Main class: org.eclipse.acceleo.module.maswithmobileagents.common.Generate

Model: /MAS/PSM_Level/PSM.uml

Target: /MAS/Code_Level/src/generatedCodeForMasWithMobileAgent

Profile result:

Configuration:

Runner: Acceleo Plug-in Application (Default)

□ Contribute traceability information to Result View

□ Enable profiling

Filter matched 33 of 33 items

Figure 1. The Acceleo/Eclipse configuration used to execute the file generate.mtl

```
---------------------------- generate.mtl file

[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/uml2/4.0.0/UML',
'http://www.eclipse.org/uml2/4.0.0/UML/Profile/L2', 'http://www.eclipse.org/uml2/4.0.0/UML/Profile/L3',
'http://www.eclipse.org/emf/2002/Ecore')]


[comment: checks if "umlElement" is stereotyped with <<stereotypeName>>/]
[query private isStereotypedWith(umlElement: Element, stereotypeName: String): Boolean =
   if umlElement.getAppliedStereotype(stereotypeName) <> null then
      true
   else
      false
   endif/]
[comment: retrieves the value of the property "propertyName" from a stereotype <<stereotypeName>>
if it is applied to "umlElement", null otherwise/]
[query private getTaggedValue(umlElement: Element, stereotypeName: String, propertyName: String):
OclAny =
   if umlElement.getAppliedStereotype(stereotypeName) <> null then
      umlElement.getValue(umlElement.getAppliedStereotype(stereotypeName), propertyName)
   else
      null
   endif/]


[comment: this template consider classes stereotyped with <<DomainConcept>> and
<<ComportmentIpmlementation>>, then applies the generation rule: 4 or 5 /]
[template public generateElement(aClass: Class)]

[comment @main /] [comment @main indicates that this template is an entry point/]

[if not (aClass.oclIsKindOf(StateMachine) or aClass.oclIsKindOf(Activity))]

            [if aClass.isStereotypedWith('PSMM_Profile::ComportmentImplementation')]

                        [comment: generates classes from aClass into Implementations.java/]
                        [generateElement(aClass, 'Implementations.java')/]

            [/if]

[/if]

[/template]


[comment: this template generate classes from ones stereotyped with <<DomainConcept>> and
<<ComportmentImplementation>> int the PSM. the generation rules: 4 and 5 are applied/]
[template public generateElement(aClass: Class, fileName: String)]

[comment @main /] [comment @main indicates that this template is an entry point/]

[file (fileName, true, 'UTF-8')]

[comment: this section adds (#import DomainTypes.java and class GoFailed) once in Implementations.java/]
[if (fileName = 'Implementations.java' and aClass.getModel().getKeywords()->
                                                                          excludes('importAddedInImpl,'))]

     [if aClass.getModel().addKeyword('importAddedInImpl,')]

['#import DomainTypes.java;'/]

// GoFailed is a class used inside the migration method to test if the move has successed or failed
class GoFailed implements HookInterface {
            Boolean moved;
            // Constructor
            public GoFailed (Boolean moved) {
                        this.moved = moved;
            }
            public void resume (GoException e) {
                        moved = new Boolean(false);
                        System.out.println("\nGo failed!!!!!!!!!!!!!!! "); // message displayed if the go command failed
            }
}
     [/if]
[/if]
```

```
[aClass.visibility/][if (aClass.isAbstract)] abstract[/if][if (aClass.isLeaf)] final[/if]
class [aClass.name.toUpperFirst()/]

[if aClass.isStereotypedWith('PSMM_Profile::ComportmentImplementation')]  extends
            [(aClass.name.first(aClass.name.size()-4)+'InterfQuasiBehavior').toUpperFirst()/]

     [if aClass.getTaggedValue('PSMM_Profile::ComportmentImplementation',
                                                       'standAlone').oclAsType(Boolean)]
            implements StandAlone
     [/if]

[elseif (aClass.isStereotypedWith('PIMM_Profile::DomainConcept') and
            aClass.getTaggedValue('PIMM_Profile::DomainConcept', 'transferable').oclAsType(Boolean))
            or aClass.isStereotypedWith('PIMM_Profile::Site')]
            implements Serializable
[/if] {

[for (p: Property | aClass.attribute) separator('\n')] [comment: generates the attributes/]
     [if p.visibility<>'package'][p.visibility/] [/if]
     [if p.upper=-1]Vector[elseif (p.type.name <> 'OclAny')][p.type.name/][else]Object[/if] [p.name/]
     [if not p.default.oclIsUndefined()] = [p.default/][/if];
[/for]

[if aClass.isStereotypedWith('PIMM_Profile::DomainConcept')] [comment: add a comment at the end of
DomainTypes/]

            // add constructor and methods manually
[/if]

[if aClass.name = 'RoleEndingBehaviorImpl'] [comment: application of the generation rule 5.4/]

            // Constructor
            public RoleEndingBehaviorImpl(Vector valuesToReturn) {
                        this.valuesToReturn = valuesToReturn;
            }

[elseif aClass.name.endsWith('GroupBehaviorImpl')] [comment: application of the generation rule 5.3/]

            private Vector agents; // to contain the list of agents that have joined the group

            // Constructeur
            public [aClass.name/]() {
                        this.agents = new Vector();
            }

[elseif aClass.name.endsWith('RoleBehaviorImpl')] [comment: application of the generation rule 5.1/]

            private Vector valuesToReturn; /* allow the role to return the updated values of attributes belonging to
                                                  the agent which play it */


     [let interactedRoles:String = aClass.attribute->select(a|a.name='interactsWith').default
                                                                                   ->asSequence()->first()]

          [for (interactedRole:String |interactedRoles.substring(2, interactedRoles.size()-1).tokenize(','))
                                                                                         separator ('\n')]
            private Actor playerOf[interactedRole/]Role; /* the agent playing the role ([interactedRole/]) with
                                                              which this role ([aClass.name/]) interacts*/
          [/for]
```

```
    [/let]

          private TypeToSpecify owner; /* the owner is the agent playing this role. Its type will be specified
                                        manually. It allows the role to manipulate the attributes of its owner */

          // Constructor
          public [aClass.name/](TypeToSpecify owner) { // the type of owner is specified manually
                    this.owner = owner;
                    this.stop = false;
                    valuesToReturn = new Vector();
          }

[elseif aClass.name.endsWith('AgentBehaviorImpl')] [comment: application of the generation rule 5.2/]

          private Actor group; // to indicate the group to join

    [let playedRoles:String = aClass.attribute->select(a|a.name='mayPlay').default->asSequence()->first()]

        [for (playedRole:String |playedRoles.substring(2, playedRoles.size()-1).tokenize(',')) separator ('\n')]
          private Actor [playedRole.toLowerFirst()/]RoleActor; /* to point to the actor representing the played
                                                                      role ([playedRole/]) */
        [/for]

    [/let]

          // Constructor
          public [aClass.name/]() {
                    /* any global variable must receive its value here. Indeed, as this agent move, this manner
                    avoids to lost values of its global variables   */
          }

[/if]

[for (o: Operation | aClass.ownedOperation) separator('\n')] [comment: generates the operations/]
    [if o.visibility.toString()<>'package'][o.visibility.toString()/][/if]
    [if(o.getType().oclIsUndefined())]void
    [else][for (p: Parameter | o.ownedParameter->select(pp: Parameter |
                                                            pp.name.equalsIgnoreCase('return')))]
                [if p.upper=-1]Vector[elseif (p.type.name <>'OclAny')][p.type.name/][else]Object[/if]
          [/for]
    [/if]
    [o.name/] ( [for (p: Parameter | o.ownedParameter->select(pp: Parameter | not pp.name.oclIsUndefined()
                              and not pp.name.equalsIgnoreCase('return'))) separator(' ,')]
                        [if p.upper=-1]Vector[elseif (p.type.name <> 'OclAny')][p.type.name/][else]Object[/if]
                        [p.name/]
                    [/for]) {


    [if o.name.equalsIgnoreCase('stopRole') and aClass.name.endsWith('RoleBehaviorImpl')]
    [comment: application of the generation rule 5.1/]

                    stop = true;

        [if not aClass.getTaggedValue('PSMM_Profile::ComportmentImplementation',
                              'standAlone').oclAsType(Boolean)]
                    // terminates
                    become(new RoleEndingBehaviorImpl(this.valuesToReturn));
        [/if]
    [elseif o.name='joinGroup' and aClass.name.endsWith('GroupBehaviorImpl')]
```

```
                        [comment: application of the generation rule 5.3/]
              // add the agent to the group
              agents.addElement(a);

              // complete the content manually
   [elseif o.name='leaveGroup' and aClass.name.endsWith('GroupBehaviorImpl')]
                        [comment: application of the generation rule 5.3/]
              // delete the agent from the group
              agents.removeElementAt(agents.indexOf(a));

              // complete the content manually
  [elseif o.name='confirmEndOfRole' and aClass.name='RoleEndingBehaviorImpl']
                       [comment: application of the generation rule 5.1/]
              suicide();
              return valuesToReturn;
  [else]
              // TODO should be implemented
   [/if]
        }
[/for]

[if aClass.isStereotypedWith('PSMM_Profile::ComportmentImplementation') and
aClass.getTaggedValue('PSMM_Profile::ComportmentImplementation', 'standAlone').oclAsType(Boolean)]

       public void run() {// the method run exist in any class which implement StandAlone

  [if aClass.name.endsWith('AgentBehaviorImpl')] [comment: application of the generation rule 5.2/]

              Vector returnedValues; // values returned by played roles
  [/if]

              // TODO should be completed from the associated stateMachine
  [if aClass.name.endsWith('RoleBehaviorImpl') and aClass.getTaggedValue
              ('PSMM_Profile::ComportmentImplementation', 'standAlone').oclAsType(Boolean)]
              [comment: application of the generation rule 5.1/]

              // terminates
              become(new RoleEndingBehaviorImpl(this.valuesToReturn));
  [/if]
        }
[/if]
}

[/file]

[comment: this section calls the template which treat stateMachines/]
[if aClass.isStereotypedWith('PSMM_Profile::ComportmentImplementation') and aClass.getTaggedValue
       ('PSMM_Profile::ComportmentImplementation', 'itsStateMachine').oclIsTypeOf(StateMachine)]

       [generateElement(aClass.getTaggedValue('PSMM_Profile::ComportmentImplementation',
              'itsStateMachine').oclAsType(StateMachine), 'principal_SM.txt')/]
[/if]



[comment: this section generates domain's types whith which attributes and parameters are typed in the PSM/]
[for (t: Type | aClass.attribute.type->asSequence()->union(aClass.ownedOperation.ownedParameter.type->
              asSequence())->asSet()) separator(',')]

  [if t.isStereotypedWith('PIMM_Profile::DomainConcept') or
```

```
                                                  t.isStereotypedWith('PIMM_Profile::Site')]
        [if t.getModel().getKeywords()->excludes(t.name+',')]
                [t.getModel().addKeyword(t.name+',')/]
                [generateElement(t.oclAsType(Class),'DomainTypes.java')/]
        [/if]
    [/if]
[/for

[/template]



[comment: this template generate interfaces from the ones stereotyped with <<Actor>> or
<<ComportmentInterface>> int the PSM. the generation rules: 1 and 2 are applied/]
[template public generateElement(aInterface: Interface)]

[comment @main /] [comment @main indicates that this template is an entry point/]

[file ('Interfaces.java', true, 'UTF-8')]

[comment: this section adds (#import DomainTypes.java;) once in Interfaces.java/]
[if (aInterface.getModel().getKeywords()->excludes('importAddedInInterf,'))]

    [if aInterface.getModel().addKeyword('importAddedInInterf,')]
          ['#import DomainTypes.java;'/]
    [/if]

[/if]

[if aInterface.isStereotypedWith('PSMM_Profile::Actor.') or
                              aInterface.isStereotypedWith('PSMM_Profile::ComportmentInterface')]

    [if (aInterface.isLeaf)]final [/if]
    interface [aInterface.name.toUpperFirst()/]
    extends
    [if aInterface.isStereotypedWith('PSMM_Profile::ComportmentInterface')] BehaviorProfile
    [else] [aInterface.name.toUpperFirst()+'BehaviorInterf,'/]
        [if aInterface.name.endsWith('Role')][' RoleEndingBehaviorInterf,'/][/if]
        [' ActorProfile'/]
    [/if] {

    [for (p: Property | aInterface.attribute) separator('\n')]
        [if p.visibility.toString()<>'package'][p.visibility/] [/if]

        [if p.upperValue.stringValue().equalsIgnoreCase('*')]Vector
        [elseif (p.type.name <> 'OclAny')][p.type.name/]
        [else]Object
        [/if] [p.name/];
    [/for]

    [for (o: Operation | aInterface.ownedOperation) separator('\n')]
        [if o.visibility.toString()<>'package'][o.visibility/] [/if]

        [if(o.getType().oclIsUndefined())]void
        [else][for (p: Parameter | o.ownedParameter->select(pp: Parameter |
                                                    pp.name.equalsIgnoreCase('return')))]
                    [if p.upperValue.stringValue().equalsIgnoreCase('*')]Vector
                    [elseif (p.type.name <> 'OclAny')][p.type.name/]
                    [else]Object
                    [/if]
                    [/for]
            [/if]

            [o.name/] ( [for (p: Parameter | o.ownedParameter->select(pp: Parameter | not
                                            pp.name.oclIsUndefined() and not pp.name.equalsIgnoreCase('return'))) separator(' ,')]
                                        [if p.upperValue.stringValue().equalsIgnoreCase('*')]Vector
                                        [elseif (p.type.name <> 'OclAny')][p.type.name/]
                                        [else]Object
                                        [/if]
                                        [p.name/]
                                [/for]);
        [/for]
}
[/if]

[/file]

[/template]


[comment: checks if a transition (in a stateMachine) has a trigger/]
[query private hasTrigger(tr: Transition): Boolean =
        if tr.ownedElement->select(e|e.oclIsKindOf(Trigger))->asSet()->isEmpty() then
                false
        else
                true
        endif/]


[comment: get the trigger of a transiton/]
[query private getTrigger(tr: Transition): Trigger =
        tr.ownedElement->select(e|e.oclIsKindOf(Trigger))->asSequence()->first().oclAsType(Trigger)/]


[comment: this template take a stateMachine in XMI format and generate it in a textual form
containing necessar details to be easily used per à java program/]
[template public generateElement(aSM: StateMachine, fileName: String)]

[comment @main /] [comment @main indicates that this template is an entry point/]

[file (fileName, true, 'UTF-8')] [comment: principal_SM.txt contains SMs of agents and roles and
secondary_SM.txt contains SMs of Migration and AfterMigration actions/]

StateMachine: [aSM.name/]

[for (r: Region | aSM.region) separator ('\n')]
        region: [r.name/]

            states:
            [if r.subvertex->asSet()->isEmpty()]notSpecified
            [else][for (s:Vertex | r.subvertex) separator (', ')][s.name/][/for]
            [/if]

            transitions:
            [if r.transition->asSet()->isEmpty()]notSpecified
            [else]
                [for (t:Transition | r.transition) separator ('\n\n')]
                    tr: [t.name/]
                        from: [t.source.name/] to: [t.target.name/]
```

```
[if not t.ownedElement->asSet()->isEmpty()]

    trigger's event:
        [if not t.hasTrigger()]not specified
        [else][t.getTrigger().event.name/] [comment: name of the event/]
            stereotype: [t.getTrigger().event.oclAsType(SignalEvent).getAppliedStereotypes()
                             ->asSequence()->first().name/]

                [if t.getTrigger().event.isStereotypedWith('PIMM_Profile::ReceiveMessage')]
               itsPreviousSend:
                    [if not t.getTrigger().event.getTaggedValue('PIMM_Profile::
                             ReceiveMessage', 'itsPreviousSend').oclAsType(PIMM_Profile::
                             ReceiveMessage).base_SignalEvent.oclIsUndefined()]

                             [t.getTrigger().event.getTaggedValue('PIMM_Profile::
                             ReceiveMessage', 'itsPreviousSend').oclAsType(PIMM_Profile::
                             SendMessage).base_SendSignalAction.name/]
                   [else]not specified
                   [/if]
               [/if]
               signal: [t.getTrigger().event.oclAsType(SignalEvent).signal.name/]
               [for (p: Property | t.getTrigger().event.oclAsType(SignalEvent).signal.attribute)
                                                                             separator('\n')]
                        [p.name/]: [p.type.name/]{[p.lower/], [p.upper/]}
                [/for]
        [/if]

    guard: [if t.guard.oclIsUndefined()]not specified
            [else][t.guard.specification.name/]
            [/if]

    effect: [if t.effect.oclIsUndefined()]not specified
            [else]
                  name: [t.effect.name/]
                  [for (e: Element | t.effect.ownedElement) separator('')]

                        [if e.oclIsKindOf(Action)]
                          action: [e.oclAsType(Action).name/]
                          stereotype: [e.oclAsType(Action).getAppliedStereotypes()
                                                          ->asSequence()->first().name/]
                             [if e.isStereotypedWith('PIMM_Profile::SendMessage')]
                             itsPreviousReceive:
                                   [if not e.getTaggedValue('PIMM_Profile::SendMessage',
                                   'itsPreviousReceive').oclAsType(PIMM_Profile::ReceiveMess
                                   age).base_SignalEvent.oclIsUndefined()]

                                       [e.getTaggedValue('PIMM_Profile::SendMessage','itsPrevi
                                       ousReceive').oclAsType(PIMM_Profile::ReceiveMessage).
                                       base_SignalEvent.name/]

                                   [else]not specified
                                   [/if]
                             signal:
                                   [e.oclAsType(SendSignalAction).signal.name/]
                                   [e.oclAsType(SendSignalAction).target.name/]:
                                   [e.oclAsType(SendSignalAction).target.type.name/]
                              [/if]
                        [/if]

                        [if e.oclIsKindOf(StateMachine)]
```

```
                                [generateElement(e.oclAsType(StateMachine),
                                'secondary_SM.txt')/]
                            [/if]

                        [/for]
                    [/if]
             [/if]
        [/for]
     [/if]

[/for]

[/file]
[/template]
```