PIM-to-PSM transformation with ATL

The PIM-to-PSM transformation rules were expressed in ATL format and summarized in a file transformPIMtoPSM.atl. Figure 1 shows the ATL/Eclipse configuration used for the execution of this file (which is given below the figure).

NB : PIMM is PIM Meta-model and PSMM is PSM Meta-model

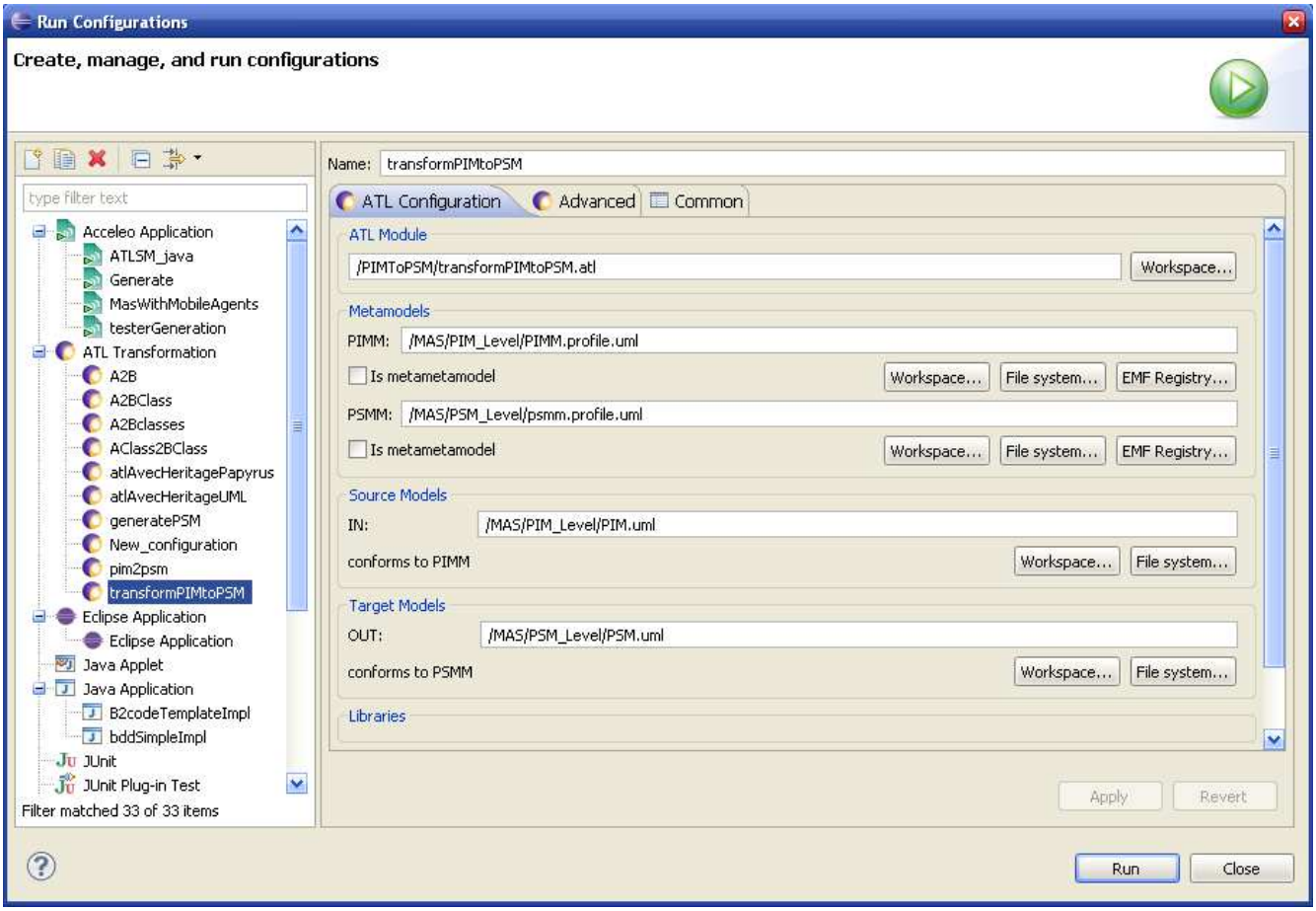

Figure 1. The ATL/Eclipse configuration used to execute the file transformPIMtoPSM.atl

---------- transformPIMtoPSM.atl file

```
-- @path PSMM=/MAS/PSM_Level/PSMM.profile.uml
-- @path PIMM=/MAS/PIM_Level/PIMM.profile.uml

module transformPIMtoPSM;
create OUT: PSMM from IN: PIMM;

helper context PIMM!Element def: hasStereotype(name: String): Boolean =
                                        not self.getAppliedStereotype(name).oclIsUndefined();
```

```
------------------------------ Model creation

helper def: model: PSMM!Model = OclUndefined;


rule models { -- Models rule transformation
from
        modelIn: PIMM!Model
to
        modelOut: PSMM!Model (
                name <- 'psm_model'
        )
do      {       -- apply PSMM on PSM
                modelOut.applyProfile(PSMM!Profile.
                        allInstancesFrom('PSMM').asSequence().first());

                thisModule.model <- modelOut; -- save a reference on modelOut

                thisModule.addDataTypesPackage(); -- add Data types
                thisModule.createRoleEnding();  -- create the Roleending role
        }
}

------------------------------ DataTypes adding

helper def: getDataType(typeName: String): PSMM!DataType =
                PSMM!DataType.allInstances()->select(t | t.name=typeName)->first();

helper def: dataTypesPackage: PSMM!Package = OclUndefined;

helper def: OclVoid: PSMM!DataType = OclUndefined;


rule addDataTypesPackage() { -- create a package for Data types
to
        ptPac: PSMM!Package (
                name <- 'DataTypes',
                nestingPackage <- thisModule.model
        ),

        stringType: PSMM!DataType (
                name <- 'String'
        ),
        integerType: PSMM!DataType (
                name <- 'Integer'
        ),
        booleanType: PSMM!DataType (
                name <- 'Boolean'
        ),
        realType: PSMM!DataType (
                name <- 'Real'
        ),
        OclAnyType: PSMM!DataType (
                name <- 'OclAny'
        )
do      {
                thisModule.dataTypesPackage <- ptPac; -- save a ref. on ptPac

                stringType.package <- thisModule.dataTypesPackage;
                integerType.package <- thisModule.dataTypesPackage;
```

```
                booleanType.package <- thisModule.dataTypesPackage;
                realType.package <- thisModule.dataTypesPackage;
                OclAnyType.package <- thisModule.dataTypesPackage;
        }
}

---------------------------- RoleEnding role creation

helper def: roleEndingInterf: PSMM!ComportmentInterface = OclUndefined;

helper def: roleEndingBehaviorInterf: PSMM!Interface = OclUndefined;


rule createRoleEnding() {  -- Create a RoleEnding interf and impl
to
        comportmentInterface: PSMM!Interface (
                name <- 'RoleEndingBehaviorInterf',
                package <- thisModule.model
        ),
        comportmentInterfaceStereotype: PSMM!ComportmentInterface (
                base_Interface <- comportmentInterface
        ),

        comportmentImplementation: PSMM!Class (
                name <- 'RoleEndingBehaviorImpl',
                ownedAttribute <- Set{},
                package <- thisModule.model
        ),
        comportmentImplementationStereotype: PSMM!ComportmentImplementation (
                standAlone <- false, -- Règle de transformation N° 5
                base_Class <- comportmentImplementation
        )
do      {       -- link interface to implementation
                comportmentInterfaceStereotype.itsImplementation <-
                        comportmentImplementationStereotype;

                -- save references on roleEndingInterf&impl
                thisModule.roleEndingInterf <- comportmentInterfaceStereotype;
                thisModule.roleEndingBehaviorInterf <- comportmentInterface;

                -- create other members: valuesToReturn, confirmEndOfRole
                thisModule.createOtherMembersOfRoleEnding(comportmentInterface,
                                                                comportmentImplementation);
        }
}


rule createOtherMembersOfRoleEnding(ownerInterfInPSM
        :PSMM!ComportmentInterface, ownerImplInPSM: PSMM!ComportmentImplementation) {
to
        valuesToReturn: PSMM!Property(
                name <- 'valuesToReturn',
                visibility <- #private,
                lower <- 0,
                upper <- -1,
                type <- thisModule.getDataType('OclAny')
        ),

        confirmEndOfRoleInterf: PSMM!Operation (
                name <- 'confirmEndOfRole',
```

```
                visibility <- #public,
                interface <- ownerInterfInPSM
        ),

        confirmEndOfRoleImpl: PSMM!Operation (
                name <- 'confirmEndOfRole',
                visibility <- #public,
                class <- ownerImplInPSM
        )
do      {
                ownerImplInPSM.ownedAttribute <- ownerImplInPSM.ownedAttribute->
                                                        including(valuesToReturn);

                -- create an Object Parameter for confirmEndOfRoleImpl&Impl
                thisModule.createObjectParameterForTheOperation(confirmEndOfRoleInterf);
                thisModule.createObjectParameterForTheOperation(confirmEndOfRoleImpl);
        }
}


rule createObjectParameterForTheOperation(op: PSMM!Operation) {
to
        paramPSM: PSMM!Parameter(
                name <- 'return',
                direction <- #return,
                type <- thisModule.getDataType('OclAny'),
                lower <- 0,
                upper <- -1
        )
do      {
                op.ownedParameter <- op.ownedParameter->including(paramPSM);
        }
}

------------------------- Agent interface to Actor Interface transformation


rule AgentToActorInterfaces { -- Agent interface (in PIM) and Actor interface (in PSM) are introduced only to
be able to write: joinGroup(Agent a), leaveGroup(Agent a), askForRoleNameRole(Agent a) in PIM and
joinGroup(Actor a), leaveGroup(Actor a), askForRoleNameRole(Actor a) in PSM

from
        agent:PIMM!Class (agent.name = 'Agent')
to
        actor:PSMM!Interface (name <- 'Actor', package <- agent.package)
}

----------------- Roles transformation – Règle de transformation N° 1


rule roles { -- Roles rule transformation
from
        role: PIMM!Class            (role.hasStereotype('PIMM_Profile::Role'))
using {
                PIM_Properties: PIMM!Property = PIMM!Property.allInstances()->
                                        asSequence()->select(p | role.ownedAttribute.includes(p));

                PIM_Operations: PIMM!Operation = PIMM!Operation.allInstances()->
                                        asSequence()->select(o | o.owner = role);
        }
```

```
to
        actor: PSMM!Interface (-- create an Actor for the role
                name <- role.name+'Role',
                package <- thisModule.model
        ),
        actorStereotype: PSMM!Actor (-- create its stereotype
                base_Interface <- actor
        ),

        comportmentInterface: PSMM!Interface (-- create an interface for role
                name <- role.name+'Role'+'BehaviorInterf',
                package <- thisModule.model
        ),
        comportmentInterfaceStereotype: PSMM!ComportmentInterface (
                base_Interface <- comportmentInterface -- its stereotype
        ),

        comportmentImplementation: PSMM!Class (-- create an implementation for the role
                name <- role.name+'Role'+'BehaviorImpl',
                ownedAttribute <- Set{},
                package <- thisModule.model
        ),
        comportmentImplementationStereotype: PSMM!ComportmentImplementation (-- create its stereotype
                standAlone <- (role.name ='BookChecker' or role.name ='ResultsDeliver'),
                -- StandAlone roles (Règle de transformation N° 5)

                -- link attribute itsStateMachine (in PSM) to attribute
                -- itsBehavior (in PIM) – Règle de transformation N° 3
                itsStateMachine <- role.getValue(role.getAppliedStereotype
                        ('PIMM_Profile::Role'), 'itsBehavior'),base_Class <-comportmentImplementation
        )
do      {       -- create other members: itsGroup, interactsWith, stop,  stopRole, become
                thisModule.createOtherMembersOfTheRole(role, comportmentInterface,
                                                                comportmentImplementation);

                -- fillfull the attribute adopts
                actorStereotype.adopts <- comportmentInterfaceStereotype;
                actorStereotype.adopts <- actorStereotype.adopts.append (thisModule.roleEndingInterf);

                -- link interface to implementation
                comportmentInterfaceStereotype.itsImplementation <-
                                                        comportmentImplementationStereotype;

                for (pim_prop in PIM_Properties){ -- copy properties of role from PIM into PSM
                        self.copyProperties(pim_prop, comportmentImplementation);
                }

                for (pim_op in PIM_Operations){ -- copy operation of role from PIM into PSM (Interf&Impl)
                        if (pim_op.visibility.toString().equalsIgnoreCase('public')) {
                                self.copyOperationsIntoInterfacePart(pim_op, comportmentInterface);
                        }
                        self.copyOperationsIntoImplementationPart(pim_op, comportmentImplementation);
                }
        }
}
rule createOtherMembersOfTheRole(pim_role: PIMM!Class, ownerInterfInPSM:
        PSMM!ComportmentInterface, ownerImplInPSM: PSMM!ComportmentImplementation) {
        -- some members are new and do not exist in the PIM
to
        itsGroup: PSMM!Property(
```

```
                name <- 'itsGroup',
                lower <- 1,
                upper <- 1,
                visibility <- #public,
                type <- thisModule.getDataType('String'),
                default <- pim_role.getValue(pim_role.getAppliedStereotype('PIMM_Profile::Role'),
                                                        'itsGroup').base_Class.name
        ),

        interactsWith: PSMM!Property(
                name <- 'interactsWith',
                lower <- 0,
                upper <- -1,
                visibility <- #public,
                type <- thisModule.getDataType('String'),
                default <- '{'
        ),

        stop: PSMM!Property(
                name <- 'stop',
                lower <- 1,
                upper <- 1,
                visibility <- #private,
                type <- thisModule.getDataType('Boolean'),
                default <- 'false'
        ),

        stopRoleInterf: PSMM!Operation (
                name <- 'stopRole',
                visibility <- #public,
                interface <- ownerInterfInPSM
        ),

        stopRoleImpl: PSMM!Operation (
                name <- 'stopRole',
                visibility <- #public,
                class <- ownerImplInPSM
        ),

        become: PSMM!Operation (-- become allows to change to RoleEnding
                name <- 'become',
                visibility <- #package,
                interface <- ownerInterfInPSM,
                ownedParameter <- Set{}
        )
do      {       -- fillfull the attribute interactsWith
                for (interactedRole in pim_role.getValue
                        (pim_role.getAppliedStereotype('PIMM_Profile::Role'),'interactsWith')) {

                        if (interactsWith.default <> '{') {
                                interactsWith.default <-
                                interactsWith.default.concat(',');
                        }

                        interactsWith.default <- interactsWith.default.concat
                                                        (interactedRole.base_Class.name);
                }
                interactsWith.default <- interactsWith.default.concat('}');

                ownerImplInPSM.ownedAttribute <- ownerImplInPSM.ownedAttribute->including(itsGroup);
```

```
                ownerImplInPSM.ownedAttribute <- ownerImplInPSM.ownedAttribute->
                                                                    including(interactsWith);
                ownerImplInPSM.ownedAttribute <- ownerImplInPSM.ownedAttribute->including(stop);

                hisModule.createParametersForBecomeOfRole(become);
        }
}


rule createParametersForBecomeOfRole(opPSM:PSMM!Operation) {
to
        paramPSM: PSMM!Parameter(
                name <- 'behavior',
                direction <- #"in",
                type <- thisModule.roleEndingBehaviorInterf,
                lower <- 1,
                upper <- 1
        ),
        paramPSMVoid: PSMM!Parameter(
                name <- 'return',
                direction <- #"return",
                type <- thisModule.OclVoid,
                lower <- 1,
                upper <- 1
        )
do      {
                opPSM.ownedParameter <- opPSM.ownedParameter
                        ->including(paramPSM);
                opPSM.ownedParameter <- opPSM.ownedParameter
                        ->including(paramPSMVoid);
        }
}

rule copyOperationsIntoInterfacePart(pim_op: PIMM!Operation, psm_comportmentInterface
                        :PSMM!Interface){
        -- copy operations from PIM into PSM (inside interfaces)
using {
                params: Set(PIMM!Parameter) = pim_op.ownedParameter->select
                                        (param | param.name.oclIsUndefined() = false);
        }
to
        psm_op: PSMM!Operation(
                name <- pim_op.name,
                interface <- psm_comportmentInterface,
                visibility <- pim_op.visibility,
                ownedParameter <- Set{}
        )
do      {
                for (param in params){
                        self.copyParameter(psm_op, param);
                }
        }
}

rule copyOperationsIntoImplementationPart(pim_op: PIMM!Operation, psm_comportmentImplementation
                        :PSMM!Class){
        -- copy operations from PIM into PSM (inside implementations)
using {
                params: Set(PIMM!Parameter) = pim_op.ownedParameter->select
                                        (param | param.name.oclIsUndefined() = false);
```

```
		}
to
		psm_op: PSMM!Operation(
			name <- pim_op.name,
			class <- psm_comportmentImplementation,
			visibility <- pim_op.visibility,
			ownedParameter <- Set{}
		)
do		{
			for (param in params){
				self.copyParameter(psm_op, param);
			}
		}
}


rule copyParameter(opPSM:PSMM!Operation, paramPIM:PIMM!Parameter){
-- copy parameters of operation from PIM into PSM
to
		paramPSM: PSMM!Parameter(
			name <- paramPIM.name,
			direction <- paramPIM.direction,
			visibility <- paramPIM.visibility,
			type <- paramPIM.type,
			lower <- paramPIM.lower,
			upper <- paramPIM.upper
		)
do		{
			opPSM.ownedParameter <- opPSM.ownedParameter->including(paramPSM);
		}
}


rule copyProperties(pim_prop: PIMM!Property, psm_Container:PSMM!Class){
-- copy properties from PIM into PSM
to
		psm_prop: PSMM!Property(
			name <- pim_prop.name,
			type <- pim_prop.type,
			lower <- pim_prop.lower,
			upper <- pim_prop.upper,
			default <- pim_prop.default,
			visibility <- pim_prop.visibility
		)
do		{
			psm_Container.ownedAttribute <- psm_Container.ownedAttribute->including(psm_prop);
		}
}



---------- Agents transformation – Règle de transformation N° 2

helper def: mobileBookSeekerAgentBehaviorInterf: PSMM!ComportmentInterface = OclUndefined;


rule agents { -- Agents rule transformation
from
		agent: PIMM!Class (agent.hasStereotype('PIMM_Profile::Agent') or
```

```
                                agent.hasStereotype('PIMM_Profile::MobileAgent'))

using {
        PIM_Properties: PIMM!Property = PIMM!Property.allInstances()->asSequence()->
                                        select(p | agent.ownedAttribute.includes(p));

        PIM_Operations: PIMM!Operation = PIMM!Operation.allInstances()
                                        ->asSequence()->select(o | o.owner = agent);
    }
to
    actor: PSMM!Interface(-- create an Actor for the agent
        name <- agent.name+'Agent',
        generalization <- ruleGeneralization,
        package <- thisModule.model
    ),
    ruleGeneralization: PSMM!Generalization (
        specific <- actor,
        general <- PIMM!Interface.allInstances()->asSequence()->select(i | i.name = 'Actor')->first()
    ),
    actorStereotype: PSMM!Actor (-- create its stereotype
        base_Interface <- actor
    ),

    comportmentInterface: PSMM!Interface(-- create an interface for agent
        name <- agent.name+'Agent'+'BehaviorInterf',
        package <- thisModule.model
    ),
    comportmentInterfaceStereotype: PSMM!ComportmentInterface (
        base_Interface <- comportmentInterface -- its stereotype
    ),

    comportmentImplementation: PSMM!Class (-- create an implementation for the agent
        name <- agent.name+'Agent'+'BehaviorImpl',
        ownedAttribute <- Set{},
        package <- thisModule.model
    ),
    comportmentImplementationStereotype: PSMM!ComportmentImplementation (-- create its stereotype
        standAlone <- true, -- Règle de transformation N° 5
        base_Class <- comportmentImplementation
    )
do  {   -- fillfull the attribute adopts
        actorStereotype.adopts <- comportmentInterfaceStereotype;

        -- link interface to implementation
        comportmentInterfaceStereotype.itsImplementation <-comportmentImplementationStereotype;

        for (pim_prop in PIM_Properties){ -- copy properties of the agent from PIM into PSM
            self.copyProperties(pim_prop, comportmentImplementation);
        }


        -- copy operation of the agent from PIM into PSM (Interf&Impl)
        for (pim_op in PIM_Operations){
            self.copyOperationsIntoInterfacePart(pim_op, comportmentInterface);
            self.copyOperationsIntoImplementationPart(pim_op, comportmentImplementation);
        }

        -- create the attribute mayPlay
        thisModule.createMayPlayAttribute(agent, comportmentImplementation);
```

```
                if (agent.hasStereotype('PIMM_Profile::MobileAgent')) {
                        thisModule.mobileBookSeekerAgentBehaviorInterf <-comportmentInterface;
                        --This serves to set the type of the parameter of become

                        -- create the become operation
                        thisModule.createBecomeOperationForMobileAgent(comportmentInterface);

                        -- link attribute itsStateMachine (in PSM) to attribute
                        -- itsBehavior (in PIM) – Règle de transformation N° 3
                        comportmentImplementationStereotype.itsStateMachine <- agent.getValue
                                (agent.getAppliedStereotype('PIMM_Profile::MobileAgent'), 'itsBehavior');
                }
                else {  -- link attribute itsStateMachine (in PSM) to attribute
                        -- itsBehavior (in PIM) – Règle de transformation N° 3
                        comportmentImplementationStereotype.itsStateMachine <-agent.getValue
                                (agent.getAppliedStereotype('PIMM_Profile::Agent'), 'itsBehavior');
                }
        }
}


rule createBecomeOperationForMobileAgent(owner_interf: PSMM!Interface) {
-- become allows the agent to loop on its principal behavior
to
        become: PSMM!Operation (
                name <- 'become',
                visibility <- #package,
                interface <- owner_interf,
                ownedParameter <- Set{}
        )
do      {
                thisModule.createParameterOfBecomeOperationForMobileAgent(become);
        }
}


rule createParameterOfBecomeOperationForMobileAgent(opPSM:PSMM!Operation) {
-- create parameters for the become operation inside an agent
to
        paramPSM: PSMM!Parameter(
                name <- 'behavior',
                direction <- #"in",
                type <- thisModule.mobileBookSeekerAgentBehaviorInterf,
                lower <- 1,
                upper <- 1
        ),
        paramPSMVoid: PSMM!Parameter(
                name <- 'return',
                direction <- #"return",
                type <- thisModule.OclVoid,
                lower <- 1,
                upper <- 1
        )
do      {
                opPSM.ownedParameter <- opPSM.ownedParameter->including(paramPSM);
                opPSM.ownedParameter <- opPSM.ownedParameter->including(paramPSMVoid);
        }
}
```

```
helper def: playedRoles: Set(PIMM!Role) = Set{};


rule createMayPlayAttribute(agent: PIMM!Class, ownerInPSM: PSMM!Class) {
-- create the payPlay attribute
to
	mayPlay_prop: PSMM!Property(
		name <- 'mayPlay',
		lower <- 1,
		upper <- -1,
		visibility <- #public,
		type <- thisModule.getDataType('String'),
		default <- '{'
	)
do	{	-- collect values of the mayPlay attribute from PIM
		if (agent.hasStereotype('PIMM_Profile::Agent')) {
			thisModule.playedRoles <- agent.getValue
				(agent.getAppliedStereotype('PIMM_Profile::Agent'), 'mayPlay');
		}
		if (agent.hasStereotype('PIMM_Profile::MobileAgent')) {
			thisModule.playedRoles <- agent.getValue
				(agent.getAppliedStereotype('PIMM_Profile::MobileAgent'), 'mayPlay');
		}

		-- fillfull the mayPlay attribute
		for (roleStereotype in thisModule.playedRoles) {
			if (mayPlay_prop.default <> '{') {
				mayPlay_prop.default <- mayPlay_prop.default.concat(',');
			}
			mayPlay_prop.default <- mayPlay_prop.default.concat
								(roleStereotype.base_Class.name);
		}
		mayPlay_prop.default <- mayPlay_prop.default.concat('}');

		ownerInPSM.ownedAttribute <- ownerInPSM.ownedAttribute->including(mayPlay_prop);
	}
}


---------- Groups transformation – Règle de transformation N° 4


rule groups { -- Groups rule transformation
from
	group: PIMM!Class (group.hasStereotype('PIMM_Profile::Group'))

using {
		PIM_Properties: PIMM!Property = PIMM!Property.allInstances()->
			asSequence()->select(p | group.ownedAttribute.includes(p));

		PIM_Operations: PIMM!Operation = PIMM!Operation.allInstances()->
			asSequence()->select(o | o.owner = group);
	}
to
	actor: PSMM!Interface(-- create an Actor for the group
		name <- group.name+'Group',
		package <- thisModule.model
	),
	actorStereotype: PSMM!Actor (-- create its stereotype
		base_Interface <- actor
```

```
		),

		comportmentInterface: PSMM!Interface (-- create an interface for the group
			name <- group.name+'Group'+'BehaviorInterf',
			package <- thisModule.model
		),
		comportmentInterfaceStereotype: PSMM!ComportmentInterface (
			base_Interface <- comportmentInterface -- its stereotype
		),

		comportmentImplementation: PSMM!Class (-- create an implementation for the group
			name <- group.name+'Group'+'BehaviorImpl',
			ownedAttribute <- Set{},
			package <- thisModule.model
		),
		comportmentImplementationStereotype: PSMM!ComportmentImplementation (-- create its stereotype
			standAlone <- false, -- Règle de transformation N° 5
			base_Class <- comportmentImplementation
		)
do		{	-- fillfull the attribute adopts
			actorStereotype.adopts <- comportmentInterfaceStereotype;

			-- link interface to implementation
			comportmentInterfaceStereotype.itsImplementation <-
					comportmentImplementationStereotype;

			for (pim_prop in PIM_Properties){ -- copy properties of the group from PIM into PSM
				self.copyProperties(pim_prop, comportmentImplementation);
			}

			-- copy operation of the group from PIM into PSM (Interf&Impl)
			for (pim_op in PIM_Operations){
				self.copyOperationsIntoInterfacePart(pim_op, comportmentInterface);
				self.copyOperationsIntoImplementationPart(pim_op, comportmentImplementation);
			}

			-- create the attributes roles of the group
			thisModule.createTheRolesAttribute(group, comportmentImplementation);
		}
}


rule createTheRolesAttribute(group_pim: PIMM!Class, owner_psm: PSMM!Class) {
-- create the attributes roles of the group
to
		propPSM: PSMM!Property(
			name <- 'roles',
			lower <- 1,
			upper <- -1,
			visibility <- #public,
			type <- thisModule.getDataType('String'),
			default <- '{'
		)
do		{	-- fillfull the attributes roles
			for (roleStereotypes in group_pim.getValue
					(group_pim.getAppliedStereotype('PIMM_Profile::Group'), 'roles')) {
				if (propPSM.default <> '{') {
					propPSM.default <- propPSM.default.concat(',');
				}
				propPSM.default <- propPSM.default.concat(roleStereotypes.base_Class.name);
```

```
                }
                propPSM.default <- propPSM.default.concat('}');

                owner_psm.ownedAttribute <- owner_psm.ownedAttribute->including(propPSM);
        }
}
```