

1. ATL Transformation Example: Maven → Ant

The Ant to Maven example describes a transformation from a file in Ant to a file in Maven (which is an extension of Ant).

1.1. Transformation overview

The aim of this transformation is to generate a file for the build tool Ant starting from files corresponding to the build tool Maven.

Here an example of files in Maven:

```
<project id="gs-example" name="gs-example">
  <build>
    <defaultGoal>build</defaultGoal>
    <sourceDirectory>.</sourceDirectory>
  </build>
</project>
```

Figure 1. project.xml

```
<project xmlns:ant="jelly:ant" default="build">
  <ant:path id="classpath">
    <ant:fileset dir="${jwsdp.home}/common/lib">
      <ant:include name="*.jar"/>
    </ant:fileset>
  </ant:path>
  <ant:property name="example" value="GSApp"/>
  <ant:property name="path" value="/${example}"/>
  <ant:property name="build"
    value="${jwsdp.home}/docs/tutorial/examples/${example}/build"/>
  <ant:property name="url" value="http://localhost:8080/manager"/>
  <ant:property file="build.properties"/>
  <ant:property file="${user.home}/build.properties"/>
  <ant:taskdef name="install" classname="org.apache.catalina.ant.InstallTask"/>
  <ant:taskdef name="reload" classname="org.apache.catalina.ant.ReloadTask"/>
  <ant:taskdef name="remove" classname="org.apache.catalina.ant.RemoveTask"/>
  <goal name="init">
    <ant:tstamp/>
  </goal>
  <goal name="prepare">
    <attainGoal name="init"/>
    <ant:mkdir ant:dir="${build}"/>
    <ant:mkdir ant:dir="${build}/WEB-INF"/>
    <ant:mkdir ant:dir="${build}/WEB-INF/classes"/>
  </goal>
  <goal name="install">
    <attainGoal name="build"/>
    <install url="${url}" username="${username}" password="${password}"
      path="${path}" war="file:${build}"/>
  </goal>
  <goal name="reload">
    <attainGoal name="build"/>
    <reload url="${url}" username="${username}" password="${password}"
      path="${path}"/>
  </goal>
</project>
```

```

<goal name="remove">
  <remove url="${url}" username="${username}" password="${password}"
                                             path="${path}" />
</goal>
<goal name="build">
  <attainGoal name="prepare" />
  <ant:javac srcdir="src" destdir="${build}/WEB-INF/classes">
    <ant:include name="**/*.java" />
    <ant:classpath refid="classpath" />
  </ant:javac>
  <ant:copy todir="${build}/WEB-INF">
    <ant:fileset dir="web/WEB-INF">
      <ant:include name="web.xml" />
    </ant:fileset>
  </ant:copy>
  <ant:copy todir="${build}">
    <ant:fileset dir="web">
      <ant:include name="*.html" />
      <ant:include name="*.jsp" />
      <ant:include name="*.gif" />
    </ant:fileset>
  </ant:copy>
</goal>
</project>

```

Figure 2. maven.xml

```

<project name="gs-example" default="build" basedir=". ">
  <target name="init">
    <tstamp/>
  </target>


  <property name="example" value="GSApp" />
  <property name="path" value="/${example}" />
  <property name="build"
    value="${jwsdp.home}/docs/tutorial/examples/${example}/build" />
  <property name="url" value="http://localhost:8080/manager" />
  <property file="build.properties" />
  <property file="${user.home}/build.properties" />

  <path id="classpath">
    <fileset dir="${jwsdp.home}/common/lib">
      <include name="*.jar" />
    </fileset>
  </path>
  <taskdef name="install" classname="org.apache.catalina.ant.InstallTask" />
  <taskdef name="reload" classname="org.apache.catalina.ant.ReloadTask" />
  <taskdef name="remove" classname="org.apache.catalina.ant.RemoveTask" />

  <target name="prepare" depends="init" description="Create build directories.">
    <mkdir dir="${build}" />
    <mkdir dir="${build}/WEB-INF" />
    <mkdir dir="${build}/WEB-INF/classes" />
  </target>

  <target name="install" description="Install Web application" depends="build">
    <install url="${url}" username="${username}" password="${password}"
      path="${path}" war="file:${build}" />
  </target>

```

	ATL TRANSFORMATION EXAMPLE	
	Maven to Ant	Date 05/08/2005

```

<target name="reload" description="Reload Web application" depends="build">
  <reload url="${url}" username="${username}" password="${password}"
                                             path="${path}" />
</target>

<target name="remove" description="Remove Web application">
  <remove url="${url}" username="${username}"
          password="${password}" path="${path}" />
</target>

<target name="build" depends="prepare"
          description="Compile app Java files and copy HTML and JSP pages" >
  <javac srcdir="src" destdir="${build}/WEB-INF/classes">
    <include name="**/*.java" />
    <classpath refid="classpath"/>
  </javac>
  <copy todir="${build}/WEB-INF">
    <fileset dir="web/WEB-INF" >
      <include name="web.xml" />
    </fileset>
  </copy>
  <copy todir="${build}">
    <fileset dir="web">
      <include name="*.html" />
      <include name="*.jsp" />
      <include name="*.gif" />
    </fileset>
  </copy>
</target>
</project>

```

Figure 3. Corresponding file in Ant

This transformation is divided into several parts:

- the injector to obtain files in xmi-format corresponding to the Maven Metamodel;
- the transformation from the Maven to the Ant Metamodel;
- the extractor to obtain the two files in xml-format corresponding to Ant.

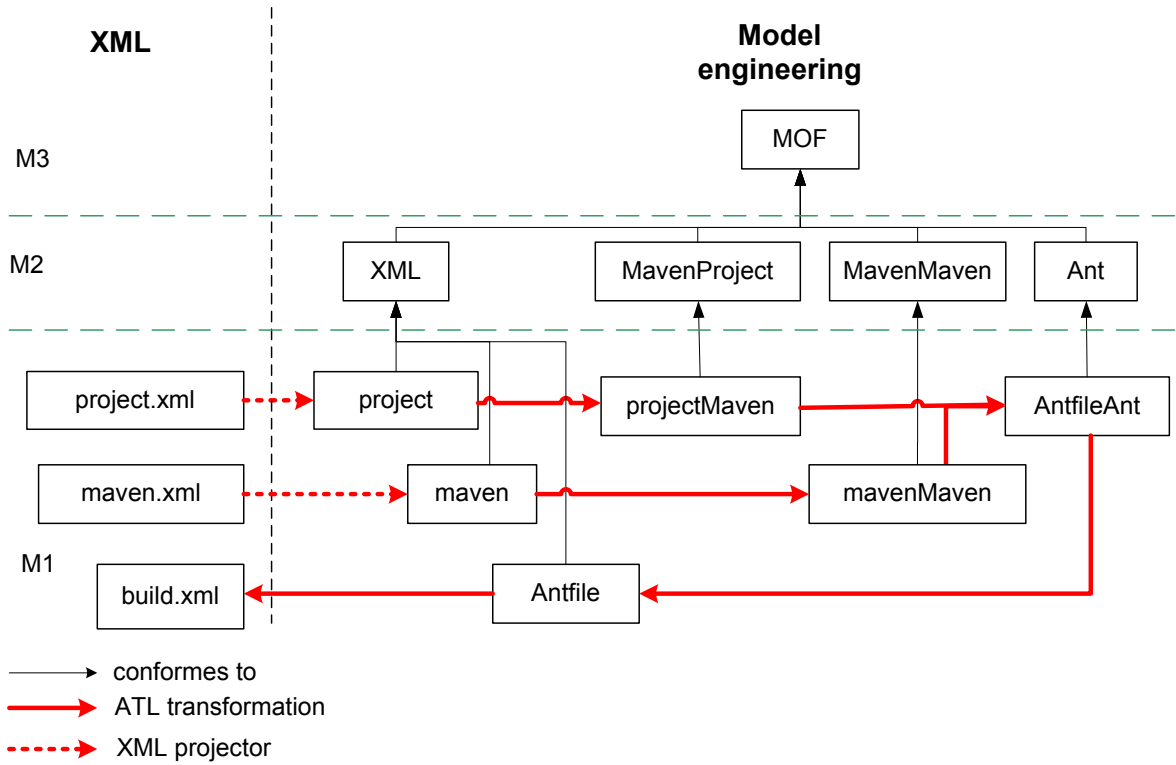


Figure 4. Transformation overview

1.2. Metamodels

1.2.1. Maven Metamodels

Maven needs two XML-based files:

- project.xml, the Maven project descriptor: this file contains the basic project configuration for maven (project name, developers, urls, dependencies, etc);
- maven.xml, the Maven configuration for defining build goals: this file contains the default maven goals for the project, plus added pre-post operations to be performed.

1.2.1.1. Metamodel for the file project.xml

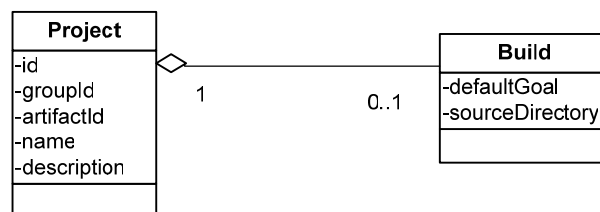


Figure 5. Metamodel of the file project.xml

A Maven project (for the file project.xml) is modeled by a Project element. A Project element is defined with the attributes id, groupId, artifactId, name, basedir description (all of these attributes are optional).

It can contain a Build element which indicates the source directory and the goal which is started by default.

It can contain others elements (like the list of developer), but these information are not deductible from an Ant file.

1.2.1.2. Metamodel for the file maven.xml

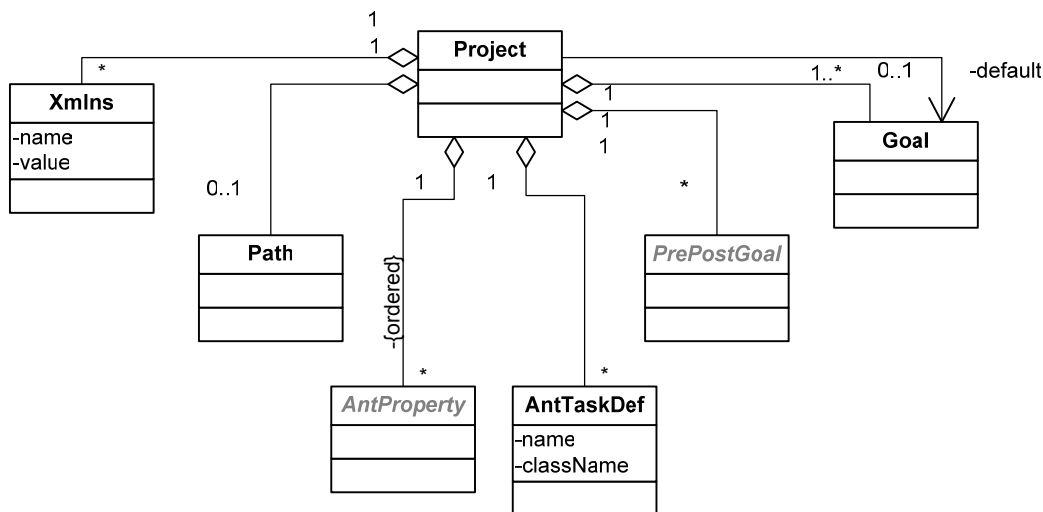


Figure 6. General Metamodel of the file maven.xml

A Maven project (for the file maven.xml) is modeled by a Project element. A Project element contains a set of Xmlns elements, an ordered set of AntProperty elements, a set of AntTaskDef elements, a set of PrePostGoal and at least one Goal element.

This project shows also the goal to start by default. But generally this information appears in the other file project.xml.

The Xmlns element represents an attribute starting with 'xmlns:' in the project tag.

The Path (and others patterns), AntProperty and AntTaskDef elements have the same definition that Path, Property and TaskDef elements in Ant (presented in Ant Metamodel).

1.2.1.2.1. Goals

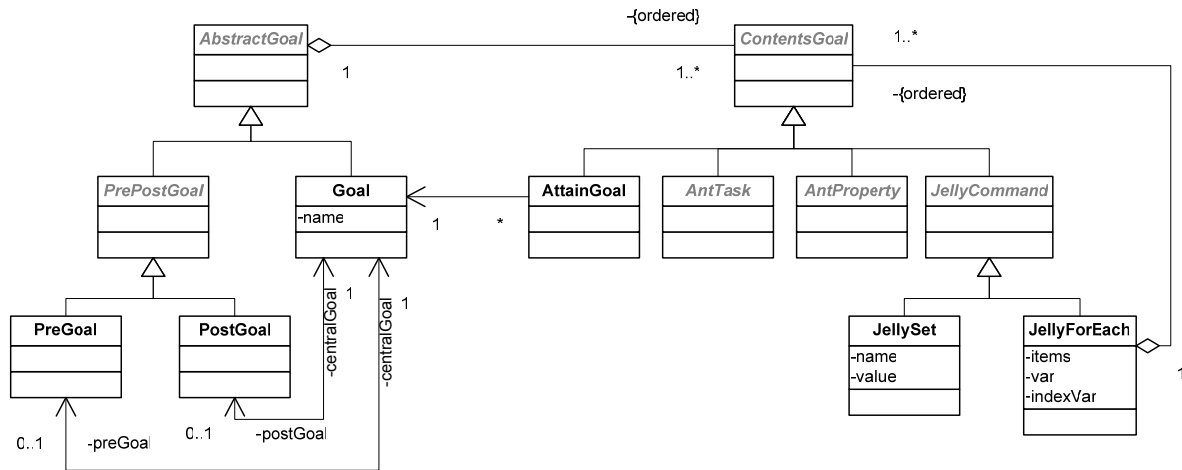


Figure 7. Goals description

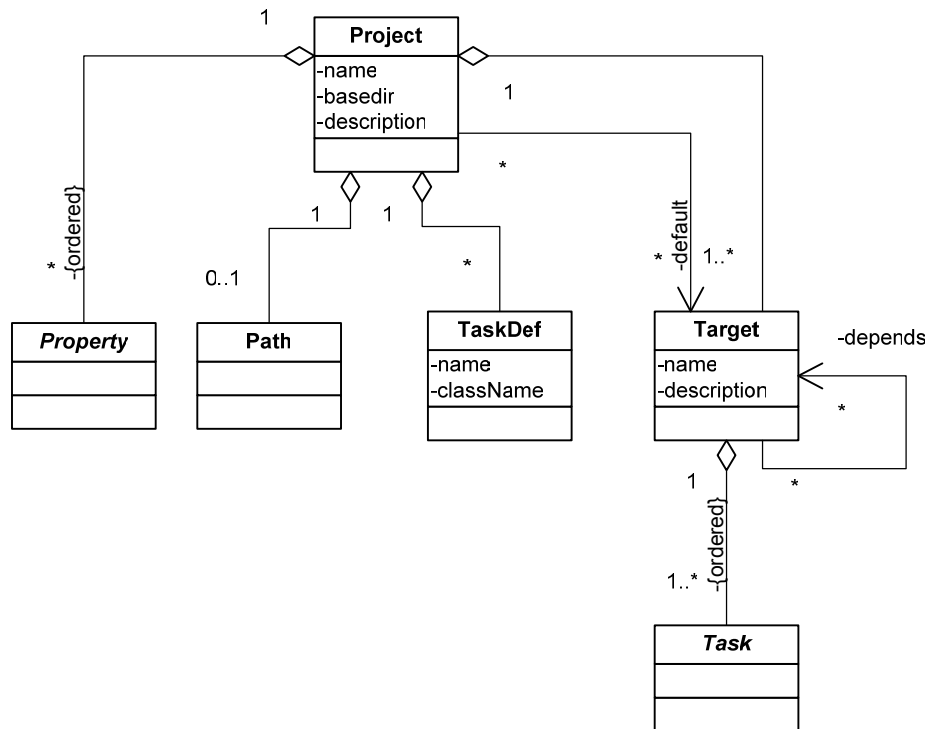
An AbstractGoal element contains a list of executions.

The PreGoal element instructs Maven to execute the defined tasks in the preGoal before achieving the central goal. The PostGoal is executed after the specified goal.

AntTask and AntProperty elements are identical to Task and Property elements presented in Ant Metamodel.

The AttainGoal element indicates which goal must be started.

Maven can use the jelly language, represented by the JellyCommand element. The JellySet element allows giving a value to a variable. The JellyForEach element allows making a loop. This last element can not be used in this transformation.

1.2.2. Ant Metamodel

Figure 8. General Metamodel of Ant

An Ant project is modeled by a Project element. A Project element project is defined with the attributes name, basedir and description (this last attribute is optional). It contains a set of properties, a path (optional), a set of TaskDef element and at least one Target element.

A Taskdef allows adding a task definition to the current project.

A Target element is an ordered set of tasks which must be executed. It can have dependencies on other targets.

1.2.2.1. Properties

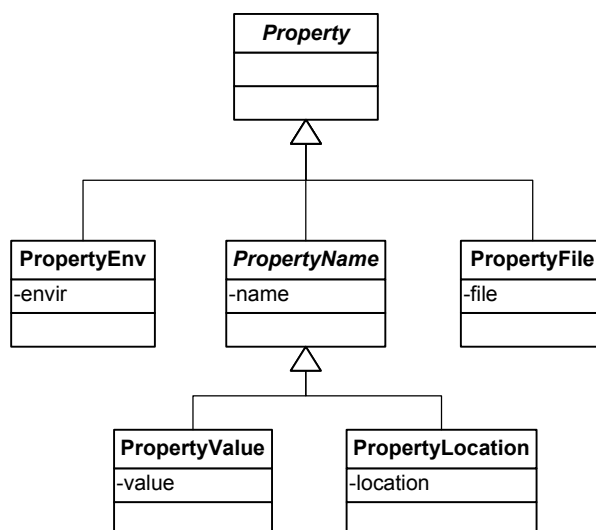


Figure 9. A few ways to define a Property

All these properties correspond to the tag 'property'.

This Metamodel allows setting various kinds of Properties:

- By supplying both the *name* and *value* attribute;
- By supplying both the *name* and *location* attribute;
- By setting the *file* attribute with the filename of the property file to load;
- By setting the *environment* attribute with a prefix to use.

1.2.2.2. Tasks

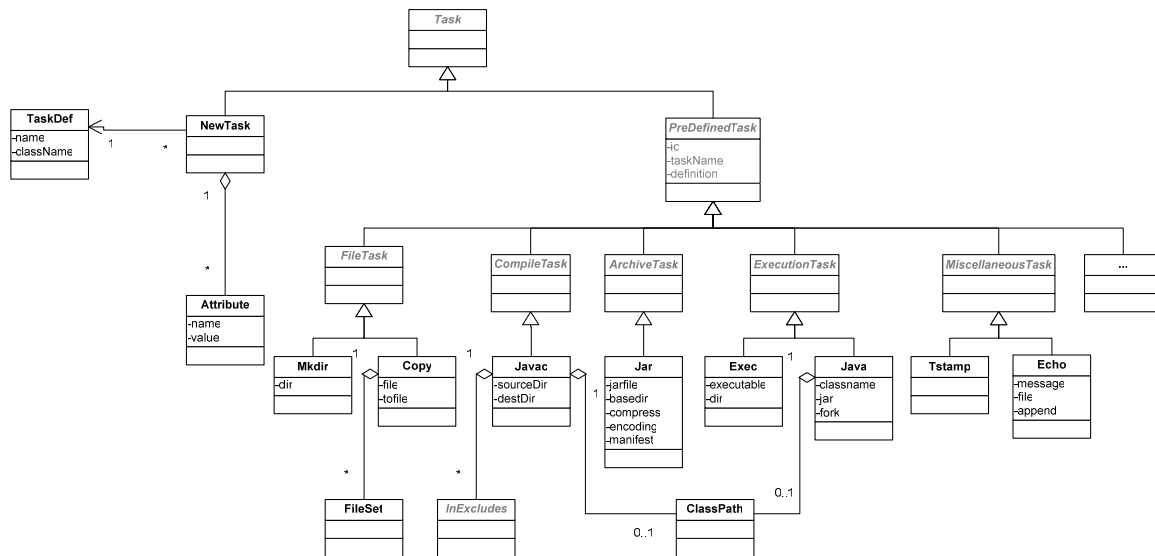


Figure 10. A few tasks

There are two types of Task:

- The tasks defined by the user. Its name is found thanks to the definition given in the TaskDef element which represents the definition of this task;
- The pre-defined tasks. There is only a sample of tasks in this Metamodel and their attributes are not all represented.

Some pre-defined tasks need a pattern (e.g. FileSet, InExcludes or ClassPath).

1.2.2.3. Pattern

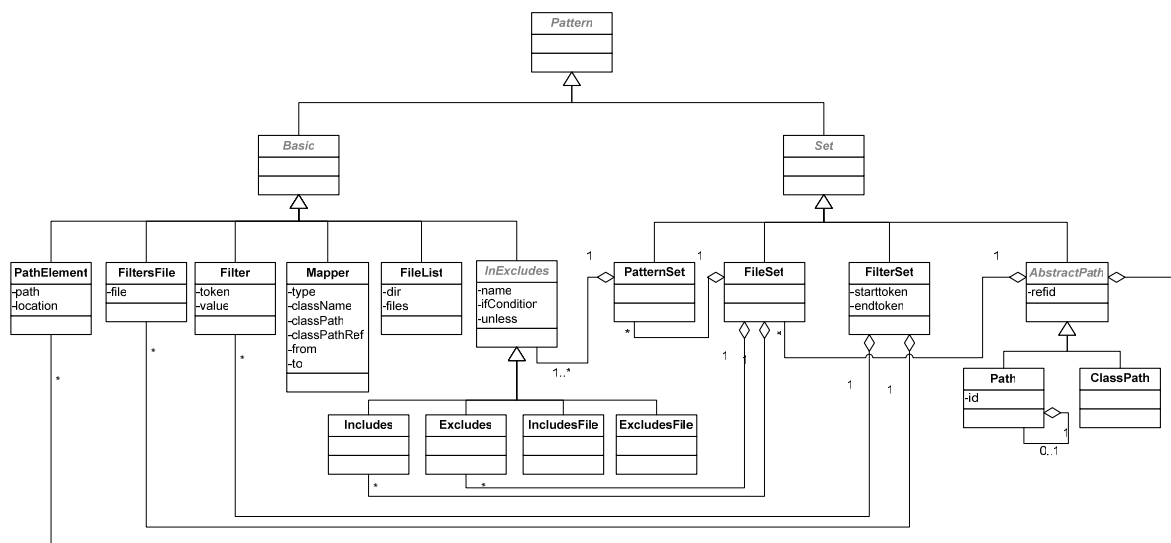



Figure 11. Metamodel of Pattern

	ATL TRANSFORMATION EXAMPLE	
	Maven to Ant	Date 05/08/2005

1.3. Injector

It creates two files corresponding to MavenProject and MavenMaven Metamodels from two files corresponding to the XML Metamodel (one representing project.xml and the other representing maven.xml). The files maven.xml and project.xml are together used in Maven, that is why their transformation (which are independent each other) appears in the same file.

1.3.1. Rules specification

These are the rules to transform two XML Models to a MavenMaven and a MavenProject Model:

- For the Root from the XMLProject Model, a Project element is created for the model corresponding to the MavenProject Metamodel,
- For an Element which name is 'build', a Build element is created,
- For the Root from the XMLMaven Model, a Project element is created for the model corresponding to the MavenMaven Metamodel,
- For an Element which name is 'property', a test on existence on its attribute must be done:
 - If this element has an attribute named 'location', a PropertyLocation element is created,
 - If this element has an attribute named 'value', a PropertyValue element is created,
 - ...
- Etc.

1.3.2. ATL Code

This ATL code for the XML to Maven transformation consists of 6 helpers and 34 rules (one rule per element in Ant Metamodel) for the MavenMaven Metamodel and 6 helpers and 2 rules concerning the MavenProject Metamodel.


1.3.2.1. Concerning MavenProject Metamodel

The `getAttribute` helper is useful for all elements having Attribute children. Its rule consists in returning the value of an attribute whose name is given in parameter. It returns "" if the required attribute does not exist. This helper uses `testAttribute` helper which indicates whether the attribute given in parameter exists (as children for the analysed element), and `getAttrVal` helper which returns the value of an attribute.

The `getText` helper returns the value of a Text belonging to an Element whose name is given in parameter, it returns "" if this Element does not exist. It uses the `testElement` helper to indicate if this Element exists and the `getTextAux` which returns the value of the Text (without test of existence).

The rule `XMLProjectRoot2MavenProjectProject` allocates a Project element.

The rule `Build` allocates a Build element.

	ATL TRANSFORMATION EXAMPLE	
	Maven to Ant	Date 05/08/2005

1.3.2.2. Concerning MavenMaven Metamodel

The getAttribute, testAttribute and getAttrVal helpers have the same rules as those presented above.

The detXmInS helper returns the value of the namespace: it removes the prefix 'xmInS:'.

The getXmInS helper returns the value of the prefix corresponding to the library whose name is given in parameter. This helper uses getXmInSAux helper which returns the name of the attribute which indicates the library given in parameter. The testXmInSAux helper indicates if this name exists.

The rule XMLMavenRoot2MavenMavenProject allocates a Project element.

The rule Goal allocates a Goal element.

...

For the rule XMLMavenRoot2MavenMavenProject, the reference 'xmInS' needs all Attribute whose name begins with 'xmInS:'. A test on the size of this name must be executed before the test on the begin of the word to not declench error (if an Attribute whose name has a size smaller than 6: the size of the word 'xmInS:') exists:

```

xmInS <- i.children ->
    select(d | if d.oclIsKindOf(XMLMaven!Attribute)
            and d.name.size()>5 then
                d.name.substring(1,6)='xmInS:'
            else
                false
            endif
    ),

```

For the rule XMLMavenRoot2MavenMavenProject, the reference 'default' need an Element named 'goal' whose value of the Attribute named 'name' has the same value as that given in the Attribute of name 'default':

```

default <- XMLMaven!Element.allInstances() ->
    select(d | d.name = 'goal'
            and d.getAttribute('name')=i.getAttribute('default'))->
    first(),

```

This value can be null.

For the rule XmInS, a test on the size of the name of the Element must be executed before the test on the begin of the word:

```

rule XmInS{
    from i : XMLMaven!Attribute(
        if i.parent.name='project'
        and i.parent.oclIsKindOf(XMLMaven!Root)
        and i.name.size()>5 then
            i.name.substring(1, 6) = 'xmInS:'
        else
            false
        endif
    )
    to o : MavenMaven!XmInS(...)
}

```

For the rules PreGoal and PostGoal (or AttainGoal), it is in the same way that the reference centralGoal (or attainGoal) is determined:

```

centralGoal <- XMLMaven!Element.allInstances() ->

```

```
select(d|d.name='goal' and
      d.getAttribute('name')=i.getAttribute('name')) ->
      first(),
```

For a rule indicating an execution included in a library, the getXmlns helper is used:

```
rule AntTaskDef{
  from i : XMLMaven!Element(
    i.name = thisModule.getXmlns('jelly:ant')+ 'taskdef'
  ) to o : MavenMaven!AntTaskDef(...)
}
```

Concerning the rule NewTask, a test is done on the existence of this new Task, that is to say that an Element named 'taskdef' must have the same value (in the Attribute named 'name') as the name of this Element. To find the reference for taskName, a research on all the elements and a selection on the name are done.


```
rule NewTask{
  from i : XMLMaven!Element(
    not(XMLMaven!Element.allInstances() ->
      select(d|d.name = thisModule.getXmlns('jelly:ant')+ 'taskdef'
        and d.getAttribute('name')=i.name) ->
        isEmpty())
  )
  to o : MavenMaven!NewTask(
    taskName <- XMLMaven!Element.allInstances() ->
      select(d|d.name = thisModule.getXmlns('jelly:ant')+ 'taskdef'
        and d.getAttribute('name')=i.name) ->
        first(),
    attributes <- i.children ->
      select(d|d.ocIsKindOf(XMLMaven!Attribute))
  )
}
```

Concerning the rule Attribut, a test is done on the existence of this new Task on the parent, that is to say that an Element named 'taskdef' must have the same value (in the Attribute named 'name') as the name of the parent of this Element.

```
rule Attribut{
  from i : XMLMaven!Attribute(
    not(XMLMaven!Element.allInstances() ->
      select(d | d.name = thisModule.getXmlns('jelly:ant')+ 'taskdef'
        and d.getAttribute('name')=i.parent.name) ->
        isEmpty())
  )
  to o : MavenMaven!Attribut(
    name <- i.name,
    value<- i.value
  )
}
```

1.4. Transformation from Maven to Ant

This transformation has two files in entry (one representing the file maven.xml and the other project.xml) and it creates a corresponding file in Ant.

	ATL TRANSFORMATION EXAMPLE	
	Maven to Ant	Date 05/08/2005

1.4.1. Rules Specification

These are the rules to transform a Maven model to Ant model:

- For a Project element (corresponding to the MavenMaven Metamodel), a Project element is created but information are extracted from the Project element of the model corresponding to the MavenProject Metamodel.
- For a Goal element, a Target element is created; the executions belonging to its eventual PreGoal and PostGoal are inserted in this same element.
- For a jellySet element, a PropertyValue element is created. It can generate errors because a value in a jelly command can be changed but not a value in a Property in Ant.
- For all properties, tasks and pattern, the elements are simply copied: for an AntPropertyValue element, a PropertyValue element is created, etc.

1.4.2. ATL Code

This ATL code for the Maven to Ant transformation consists of 30 rules and 4 helpers.

The `getAllTasks` helper returns a sequence of Tasks concerning a Goal Element. It inserts also those which are in its possible preGoal element at the beginning, and it inserts at the end those which are in its possible postGoal. It uses `getTasksAux` which returns Tasks elements belonging to an AbstractGoal element.

The `getAllAttainGoal` helper has the same principle that the `getAllTasks` helper but concerning the AttainGoal elements. It uses `getAttainGoalAux` helper which returns AttainGoal elements belonging to an AbstractGoal element.

The rule `MavenProjects2AntProject` allocates a Project element. It needs the Project element from the MavenMaven Metamodel (in the `from`) and the Project element from MavenProject Metamodel (mp in the `using`):

- The attributes name, basedir, description and the reference default are determined from the model corresponding to the MavenProject Metamodel;
- The others references are determined from the model corresponding to the MavenMaven Metamodel.

All JellySet and AntProperty elements allocates each one a Property element which is located directly in the Project whatever their place in the maven project (it can be in a Project element or in a Goal element). It is the same case for the AntTaskDef element.

```

rule MavenProjects2AntProject{
  from mm : MavenMaven!Project
  using{
    -- to have the second file in entry
    mp : MavenProject!Project =
      MavenProject!Project.allInstances()->
        asSequence()->
          first();
    -- to obtain all properties (JellySet and AntProperties)
    allJellySets : Sequence(MavenMaven!JellySet) =
      MavenMaven!JellySet.allInstances()->
        asSequence();
  }
}

```

```
allProperties : Sequence(MavenMaven!AntProperty) =
    MavenMaven!AntProperty.allInstances() ->
    asSequence();
-- to obtain all taskdef (even those which are inside a goal)
allTaskDefs : Sequence(MavenMaven!AntTaskDef) =
    MavenMaven!AntTaskDef.allInstances() ->
    asSequence();
}
to a : Ant!Project(
    name <- mp.name,
    basedir <- mp.build.sourceDirectory,
    default <- MavenMaven!Goal.allInstances()->
        select (e|e.name=mp.build.defaultGoal) ->
        first();,
    -- if there are several properties or jellySet with the same value,
    -- there are all represented
    properties <- Sequence{allProperties,allJellySets},
    path <- mm.path,
    taskdef <- allTaskDefs,
    targets <- mm.goals,
    description <- mp.description
)
}
```

The rule MavenGoal2AntTarget allocates a Target element:

- its dependencies are deductible thanks to the getAllAttainGoal helper (which gives the dependencies from its possible PreGoal, then whose from the Goal and whose from its possible PostGoal);
- Its tasks are deductible thanks to the getAllTasks helper.

```
rule MavenGoal2AntTarget{
    from mm : MavenMaven!Goal
    to a : Ant!Target(
        name <- mm.name,
        depends <- mm.getAllAttainGoal(),
        tasks <- mm.getAllTasks()
    )
}
```

All the others rules are simple copies of property, task or pattern.


1.5. Extractor

It creates a file corresponding to XML Metamodel from the obtained file in Ant Metamodel.

1.5.1. Rules specification

These are the rules to transform an Ant Model to a XML Model:

- For the Project, a Root element is created,
- For a Comment element, an Element which name is 'comment' is created,
- Etc.

	ATL TRANSFORMATION EXAMPLE	
	Maven to Ant	Date 05/08/2005

1.5.2. ATL Code

This ATL code for the Ant to XML transformation consists of 1 helper and 24 rules.

The concat helper allows concatenating a sequence of string given in parameter. Two elements are separated by a comma. This helper is useful for the attribute depends of a target.

The rule Project2Root creates a Root element for the projects having an attribute named description:

```
rule Project2Root{
  from i : Ant!Project(
    if i.description.oclIsUndefined()
    then false
    else not(i.description='')
    endif
  )
  to o : XML!Root(...)
}
```

The 'if then else' instruction is used: when the first test failed, the second is not executed.

There is another rule Project2RootWithoutDescription for the project not having description. Thus, there is no Attribute element named 'description' which has no value.

There is a rule for each element.

I. Maven Metamodel in KM3

I.1 Project.xml

```
1 package MavenProject {
2
3   -- @comments represents the current project
4   class Project{
5     attribute id [0-1] : String;
6     attribute groupId [0-1] : String;
7     attribute artifactId [0-1] : String;
8     attribute name [0-1] : String;
9     attribute description [0-1] : String;
10    reference build [0-1] container : Build;
11  }
12
13  -- @comments represents the tag 'build'
14  -- containing the informations required to build the project
15  class Build{
16    attribute defaultGoal [0-1] : String;
17    attribute sourceDirectory : String;
18    attribute unitTestSourceDirectory [0-1] : String;
19    reference uniTest [*] : Resource;
20    reference resources [*] : Resource;
21  }
22 }
23 package PrimitiveTypes{
24   datatype String;
25 }
```


I.2 Maven.xml

```
1 package MavenMaven {
2   -- @begin project
3   -- @comments central element of the file
4   class Project {
5     reference xmlns [*] container : Xmlns;
6     reference "default" [0-1] : Goal;
7     reference path [0-1] container : Path;
8     reference properties [*] ordered container : AntProperty;
9     reference taskdefs [*] container : AntTaskDef;
10    reference prePostGoals [*] container : PrePostGoal;
11    reference goals [1-*] container : Goal;
12  }
13  -- @end project
14
15  class Xmlns {
16    attribute name: String;
17    attribute value : String;
18  }
19
20  -- @begin antProperty
21  -- @comments represents the tag 'property': the properties for a project
22  abstract class AntProperty extends ContentsGoal{}
23
24  abstract class AntPropertyName extends AntProperty{
25    attribute name : String;
26  }
27  -- @comments represents a property to set a value
28  class AntPropertyValue extends AntPropertyName{
29    attribute value : String;
30  }
31  -- @comments represents a property set
32  --to the absolute filename of the given file
33  class AntPropertyLocation extends AntPropertyName{
34    attribute location : String;
35  }
36  -- @comments represents a property file to load
37  class AntPropertyFile extends AntProperty{
38    attribute file : String;
39  }
40  -- @comments represents a property retrieving environment variables
41  class AntPropertyEnv extends AntProperty{
42    attribute environment : String;
43  }
44  -- @end antProperty
45
46  -- @begin jellyCommands
47  abstract class JellyCommand extends ContentsGoal{}
48
49  -- @comments The set tag sets the jelly variable named by the var
50  -- attribute to the value given by the value attribute.
51  -- @comments Unlike Ant properties, Jelly variables can be changed
52  -- once they have been given a value
53  class JellySet extends JellyCommand{
54    attribute var : String;
55    attribute value : String;
56  }
```


```
57
58 class JellyForEach extends JellyCommand{
59     attribute items : String;
60     attribute var : String;
61     attribute indexVar : String;
62     reference contents ordered container : ContentsGoal;
63 }
64 -- @end jellyCommands
65
66 -- @begin goals
67 -- @comments represents a set of tasks which must be executed
68 abstract class AbstractGoal{
69     reference contentsGoal [1-*] ordered container : ContentsGoal;
70 }
71
72 abstract class ContentsGoal{}
73
74 class AttainGoal extends ContentsGoal{
75     reference attainGoal : Goal;
76 }
77
78 -- @comments represent extensions of a goal
79 abstract class PrePostGoal extends AbstractGoal{}
80
81 class PreGoal extends PrePostGoal{
82     reference centralGoal : Goal oppositeOf preGoal;
83 }
84
85 class PostGoal extends PrePostGoal{
86     reference centralGoal : Goal oppositeOf postGoal;
87 }
88
89 -- @comments represents a goal
90 class Goal extends AbstractGoal{
91     attribute name : String;
92     reference preGoal [0-1] : PreGoal oppositeOf centralGoal;
93     reference postGoal [0-1] : PostGoal oppositeOf centralGoal;
94 }
95 -- @end goals
96
97 -- @begin pattern
98 -- @comments represents complex parameters for some tasks
99 abstract class Pattern{}
100
101 -- @begin basicPattern
102 -- @comments represents a basic parameter(no children)
103 abstract class Basic extends Pattern{}
104
105 -- @comments represents the tag 'mapper' (mapping file names)
106 class Mapper extends Basic{
107     attribute type [0-1] : String;
108     attribute classname [0-1] : String;
109     attribute classpath [0-1] : String;
110     attribute classpathref [0-1] : String;
111     attribute from [0-1] : String;
112     attribute to [0-1] : String;
113 }
114
115 -- @comments represents the tag 'include','exclude',
116 -- 'includeFile' and 'excludeFile'(including or excluding files)
117 abstract class InExcludes extends Basic{
118     attribute name : String;
```

```
119     attribute ifCondition [0-1] : String;
120     attribute unless [0-1] : String;
121 }
122
123 class Includes extends InExcludes{}
124 class Excludes extends InExcludes{}
125 class IncludesFile extends InExcludes{}
126 class ExcludesFile extends InExcludes{}
127
128 -- @comments represents lists of files
129 class FileList extends Basic{
130     attribute dir : String;
131     attribute files : String;
132 }
133
134 -- @comments represents a filter: to replace a token value
135 class Filter extends Basic{
136     attribute token : String;
137     attribute value : String;
138 }
139
140 -- @comments represents the tag filtersfile:
141 -- to load a file containing name value pairs
142 class FiltersFile extends Basic{
143     attribute file : String;
144 }
145
146 -- @comments represents the tag 'pathelement'
147 class PathElement extends Basic{
148     attribute path : String;
149     attribute location : String;
150 }
151 -- @end basicPattern
152
153 -- @begin setPattern
154 -- @comments represents set parameters
155 abstract class Set extends Pattern{}
156
157 -- @comments represents the tag 'patternset'
158 class PatternSet extends Set{
159     reference inexcludes [1-*] container : InExcludes;
160 }
161
162 -- @comments represents the tag 'fileset' representing a group of files
163 class FileSet extends Set{
164     attribute dir : String;
165     reference patternset [*] container : PatternSet;
166     reference include [*] container : Includes;
167     reference exclude [*] container : Excludes;
168 }
169
170 -- @comments represents the tag 'filterset'
171 -- representing a group of filters
172 class FilterSet extends Set{
173     attribute starttoken [0-1] : String;
174     attribute endtoken [0-1] : String;
175     reference filter [*] container : Filter;
176     reference filtersfile [*] container : FiltersFile;
177 }
178
179 -- @comments represents the tag 'path'
180 class Path extends Set{
```

```
181     attribute id : String;
182     attribute refid [0-1] : String;
183     reference path [0-1] container : Path;
184     reference pathElement [*] container : PathElement;
185     reference fileset [*] container : FileSet;
186 }
187
188 -- @comments represents the tag 'classpath'
189 class ClassPath extends Set{
190     attribute refid : String;
191     reference pathElement [*] container : PathElement;
192     reference fileset [*] container : FileSet;
193 }
194 -- @end setPattern
195 -- @end pattern
196
197 -- @begin antTasks
198 -- @comments represents a piece of code
199 abstract class Task extends ContentsGoal{}
200
201 -- @begin newTask
202 -- @comments represents a task defined by the user
203 class AntTaskDef extends ContentsGoal{
204     attribute name : String;
205     attribute classname : String;
206 }
207
208 -- @comments represents a call of a task created by the user
209 class NewTask extends Task {
210     reference taskName : AntTaskDef;
211     reference attributes[*] container : Attribut;
212 }
213
214 -- @comments represents a attribute used in a new task
215 class Attribut{
216     attribute name : String;
217     attribute value : String;
218 }
219 -- @end newTask
220
221 -- @begin predefinedTasks
222 -- @comments represents predefined tasks
223 abstract class PreDefinedTask extends Task{
224     attribute id [0-1] : String;
225     attribute taskname [0-1] : String;
226     attribute description [0-1] : String;
227 }
228
229 -- @begin executionTasks
230 abstract class ExecutionTask extends PreDefinedTask{}
231
232 -- @comments represents the tag 'exec': execute a system command
233 class Exec extends ExecutionTask{
234     attribute executable : String;
235     attribute dir : String;
236 }
237
238 -- @comments represents the tag 'java': execute a java class
239 class Java extends ExecutionTask{
240     attribute classname : String;
241     attribute jar [0-1] : String;
242     attribute fork [0-1] : String;
```

```
243     reference classPath [0-1] container : ClassPath;
244 }
245 -- @end executionTasks
246
247 -- @begin miscellaneousTasks
248 abstract class MiscellaneousTask extends PreDefinedTask{}
249
250 -- @comments represents the tag 'echo':
251 -- echoes text to System.out or to a file
252 class Echo extends MiscellaneousTask{
253     attribute message : String;
254     attribute file [0-1] : String;
255     attribute append [0-1] : String;
256 }
257
258 -- @comments represents the tag 'tstamp' : set the tstamp
259 class Tstamp extends MiscellaneousTask{
260     reference format[*] container : FormatTstamp;
261 }
262
263 class FormatTstamp{
264     attribute property : String;
265     attribute pattern : String;
266     attribute offset [0-1] : String;
267     attribute unit [0-1] : String;
268     attribute locale [0-1] : String;
269 }
270 -- @end miscellaneousTasks
271
272 -- @begin compileTasks
273 abstract class CompileTask extends PreDefinedTask{}
274
275 -- @comments represents the tag 'javac':
276 -- compiles the specified source file(s)
277 class Javac extends CompileTask{
278     attribute srcdir : String;
279     attribute destdir [0-1]: String;
280     attribute debug [0-1] : String;
281     attribute fork [0-1] : String;
282     attribute optimize [0-1] : String;
283     attribute deprecation [0-1] : String;
284     reference inExcludes[*] container : InExcludes;
285     reference classPath [0-1] container : ClassPath;
286 }
287 -- @end compileTasks
288
289 -- @begin documentationTasks
290 abstract class DocumentationTask extends PreDefinedTask{}
291
292 class Javadoc extends DocumentationTask{
293     attribute sourcepath : String;
294     attribute destdir : String;
295     attribute packagenames : String;
296     attribute defaultexcludes : String;
297     attribute author : String;
298     attribute version : String;
299     attribute use : String;
300     attribute windowtitle : String;
301 }
302 -- @end documentationTasks
303
304 -- @begin archiveTasks
```

```
305     abstract class ArchiveTask extends PreDefinedTask{}
306
307     -- @comments represents the tag 'jar': jars a set of files
308     class Jar extends ArchiveTask{
309         attribute jarfile : String;
310         attribute basedir [0-1] : String;
311         attribute compress [0-1] : String;
312         attribute encoding [0-1] : String;
313         attribute manifest [0-1] : String;
314     }
315     -- @end archiveTasks
316
317     -- @begin fileTasks
318     abstract class FileTask extends PreDefinedTask{}
319
320     -- @comments represents the tag 'mkdir': creates a directory
321     class Mkdir extends FileTask{
322         attribute dir : String;
323     }
324
325     -- @comments represents the tag 'copy':
326     -- copies a file or Fileset to a new file or directory
327     class Copy extends FileTask{
328         attribute file [0-1] : String;
329         attribute presserelastmodified [0-1] : String;
330         attribute tofile [0-1] : String;
331         attribute todir [0-1] : String;
332         attribute overwrite [0-1] : String;
333         attribute filtering [0-1] : String;
334         attribute flatten [0-1] : String;
335         attribute includeEmptyDirs [0-1] : String;
336         reference fileset [0-1] container : FileSet;
337         reference filterset [0-1] container : FilterSet;
338         reference mapper [0-1] container : Mapper;
339     }
340
341     -- @comments represents the tag 'delete':
342     -- deletes either a single file, all files and sub-directories
343     -- in a specified directory, or a set of files specified by one
344     -- or more FileSets
345     class Delete extends FileTask{
346         attribute file [0-1] : String;
347         attribute dir [0-1] : String;
348         attribute verbose [0-1] : String;
349         attribute quiet [0-1] : String;
350         attribute failonerror [0-1] : String;
351         attribute includeEmptyDirs [0-1] : String;
352         attribute includes [0-1] : String;
353         attribute includesfile [0-1] : String;
354         attribute excludes [0-1] : String;
355         attribute excludesfile [0-1] : String;
356         attribute defaultexcludes [0-1] : String;
357     }
358     -- @end fileTasks
359
360     -- @begin executionTasks
361     abstract class ExecutionTask extends PreDefinedTask{}
362
363     -- @comments represents the tag 'exec': executes a system command
364     class Exec extends ExecutionTask{
365         attribute executable : String;
366         attribute dir : String;
```

	ATL TRANSFORMATION EXAMPLE	
	Maven to Ant	Date 05/08/2005

```
367     }
368     -- @end executionTasks
369     -- @end antTasks
370     }
371     package PrimitiveTypes{
372     datatype String ;
373     }
```

II. Ant Metamodel in KM3

```
1 package Ant{
2   -- @begin central element
3   class Project{
4     attribute name [0-1] : String;
5     attribute basedir [0-1] : String;
6     attribute description [0-1] : String;
7     reference "default" : Target;
8     reference path [0-1] container : Path;
9     reference properties [*] ordered container : Property;
10    reference taskdef [*] container : TaskDef;
11    reference targets [1-*] ordered container : Target;
12  }
13  -- @end central element
14
15
16  -- @begin property
17  -- @comments represents the properties for a project
18  abstract class Property {}
19
20  class PropertyName extends Property{
21    attribute name : String;
22  }
23
24  -- @comments represents a property to set a value
25  class PropertyValue extends PropertyName{
26    attribute value : String;
27  }
28
29  -- @comments represents a property set to the absolute filename
30  -- of the given file
31  class PropertyLocation extends PropertyName{
32    attribute location : String;
33  }
34
35  -- @comments represents a property file to load
36  class PropertyFile extends Property{
37    attribute file : String;
38  }
39
40  -- @comments represents a property retrieving environment variables
41  class PropertyEnv extends Property{
42    attribute environment : String;
43  }
44  -- @end property
45
46
47  -- @begin target
48  -- @comments represents a set of tasks which must be executed
49  class Target{
50    attribute name : String;
51    attribute description[0-1] : String;
52    attribute unless [0-1] : String;
53    attribute ifCondition [0-1] : String;
54    reference depends [*] : Target;
55    reference tasks [*] ordered container : Task oppositeOf target;
56  }
57  -- @end target
58
```



```
59
60 -- @begin pattern
61 -- @comments represents complex parameters for some tasks
62 abstract class Pattern{}
63
64 -- @begin basicPattern
65 -- @comments represents a basic parameter (no children)
66 abstract class Basic extends Pattern{}
67
68 -- @comments represents the tag 'mapper' (mapping file names)
69 class Mapper extends Basic{
70     attribute type [0-1] : String;
71     attribute classname [0-1] : String;
72     attribute classpath [0-1] : String;
73     attribute classpathref [0-1] : String;
74     attribute from [0-1] : String;
75     attribute to [0-1] : String;
76 }
77
78 -- @comments represents the tag 'include','exclude',
79 -- 'includeFile' and 'excludeFile'(including or excluding files)
80 abstract class InExcludes extends Basic{
81     attribute name : String;
82     attribute ifCondition [0-1] : String;
83     attribute unless [0-1] : String;
84 }
85
86 class Includes extends InExcludes{}
87 class Excludes extends InExcludes{}
88 class IncludesFile extends InExcludes{}
89 class ExcludesFile extends InExcludes{}
90
91 -- @comments represents lists of files
92 class FileList extends Basic{
93     attribute dir : String;
94     attribute files : String;
95 }
96
97 -- @comments represents a filter : to replace a token value
98 class Filter extends Basic{
99     attribute token : String;
100    attribute value : String;
101 }
102
103 -- @comments represents the tag filtersfile:
104 -- to load a file containing name value pairs
105 class FiltersFile extends Basic{
106     attribute file : String;
107 }
108
109 -- @comments represents the tag pathelement
110 class PathElement extends Basic{
111     attribute path : String;
112     attribute location : String;
113 }
114 -- @end basicPattern
115 -- @begin setPattern
116 -- @comments represents set parameters
117 abstract class Set extends Pattern{}
118
119 -- @comments represents the tag 'patternset'
120 class PatternSet extends Set{
```

```
121     reference inexcludes [1-]* container : InExcludes;
122 }
123
124 -- @comments represents the tag 'fileset' representing a group of files
125 class FileSet extends Set{
126     attribute dir : String;
127     reference patternset [*] container : PatternSet;
128     reference include [*] container : Includes;
129     reference exclude [*] container : Excludes;
130 }
131
132 -- @comments represents the tag 'filterset'
133 -- representing a group of filters
134 class FilterSet extends Set{
135     attribute starttoken [0-1] : String;
136     attribute endtoken [0-1] : String;
137     reference filter [*] container : Filter;
138     reference filtersfile [*] container : FiltersFile;
139 }
140
141 abstract class AbstractPath extends Set{
142     attribute refid [0-1] : String;
143     reference pathElement [*] container : PathElement;
144     reference fileset [*] container : FileSet;
145 }
146
147 -- @comments represents the tag 'path'
148 class Path extends AbstractPath{
149     attribute id : String;
150     reference path [0-1] container : Path;
151 }
152
153 -- @comments represents the tag 'classpath'
154 class ClassPath extends AbstractPath{
155 }
156 -- @begin setPattern
157 -- @end pattern
158
159 -- @begin task
160 -- @comments represents a piece of code
161 abstract class Task{
162     reference target : Target oppositeOf tasks;
163 }
164 -- @begin newTask
165 -- @comments represents a task defined by the user
166 class TaskDef{
167     attribute name : String;
168     attribute classname : String;
169 }
170
171 -- @comments represents a call of a task created by the user
172 class NewTask extends Task {
173     reference taskName : TaskDef;
174     reference attributes[*] container : Attribut;
175 }
176
177 -- @comments represents a attribute used in a new task
178 class Attribut{
179     attribute name : String;
180     attribute value : String;
181 }
182 -- @end newTask
```

```
183
184 -- @begin predefinedTasks
185 -- @comments represents predefined tasks
186 abstract class PreDefinedTask extends Task{
187     attribute id [0-1] : String;
188     attribute taskname [0-1] : String;
189     attribute description [0-1] : String;
190 }
191
192 -- @begin executionTasks
193 abstract class ExecutionTask extends PreDefinedTask{}
194
195 -- @comments represents the tag 'exec': execute a system command
196 class Exec extends ExecutionTask{
197     attribute executable : String;
198     attribute dir : String;
199 }
200
201 -- @comments represents the tag 'java': execute a java class
202 class Java extends ExecutionTask{
203     attribute classname : String;
204     attribute jar [0-1] : String;
205     attribute fork [0-1] : String;
206     reference classPath [0-1] container : ClassPath;
207 }
208 -- @end executionTasks
209
210
211 -- @begin miscellaneousTasks
212 abstract class MiscellaneousTask extends PreDefinedTask{}
213
214 -- @comments represents the tag 'echo':
215 -- echoes text to System.out or to a file
216 class Echo extends MiscellaneousTask{
217     attribute message : String;
218     attribute file [0-1] : String;
219     attribute append [0-1] : String;
220 }
221
222 -- @comments represents the tag 'tstamp': set the tstamp
223 class Tstamp extends MiscellaneousTask{
224     reference format[*] container : FormatTstamp;
225 }
226
227 class FormatTstamp{
228     attribute property : String;
229     attribute pattern : String;
230     attribute offset [0-1] : String;
231     attribute unit [0-1] : String;
232     attribute locale [0-1] : String;
233 }
234 -- @end miscellaneousTasks
235
236 -- @begin compileTasks
237 abstract class CompileTask extends PreDefinedTask{}
238
239 -- @comments represents the tag 'javac':
240 -- compiles the specified source file(s)
241 class Javac extends CompileTask{
242     attribute srcdir : String;
243     attribute destdir [0-1]: String;
244     attribute debug [0-1] : String;
```

```
245     attribute fork [0-1] : String;
246     attribute optimize [0-1] : String;
247     attribute deprecation [0-1] : String;
248     reference inExcludes[*] container : InExcludes;
249     reference classPath [0-1] container : ClassPath;
250 }
251 -- @end compileTasks
252
253 -- @begin documentationTasks
254 abstract class DocumentationTask extends PreDefinedTask{}
255
256 class Javadoc extends DocumentationTask{
257     attribute sourcepath : String;
258     attribute destdir : String;
259     attribute packagenames : String;
260     attribute defaultexcludes : String;
261     attribute author : String;
262     attribute version : String;
263     attribute use : String;
264     attribute windowtitle : String;
265 }
266 -- @end documentationTasks
267
268 -- @begin archiveTasks
269 abstract class ArchiveTask extends PreDefinedTask{}
270
271 -- @comments represents the tag 'jar': jars a set of files
272 class Jar extends ArchiveTask{
273     attribute jarfile : String;
274     attribute basedir [0-1] : String;
275     attribute compress [0-1] : String;
276     attribute encoding [0-1] : String;
277     attribute manifest [0-1] : String;
278 }
279 -- @end archiveTasks
280
281 -- @begin fileTasks
282 abstract class FileTask extends PreDefinedTask{}
283
284 -- @comments represents the tag 'mkdir': creates a directory
285 class Mkdir extends FileTask{
286     attribute dir : String;
287 }
288
289 -- @comments represents the tag 'copy':
290 -- copies a file or Fileset to a new file or directory
291 class Copy extends FileTask{
292     attribute file [0-1] : String;
293     attribute preservelastmodified [0-1] : String;
294     attribute tofile [0-1] : String;
295     attribute todir [0-1] : String;
296     attribute overwrite [0-1] : String;
297     attribute filtering [0-1] : String;
298     attribute flatten [0-1] : String;
299     attribute includeEmptyDirs [0-1] : String;
300     reference fileset [0-1] container : FileSet;
301     reference filterset [0-1] container : FilterSet;
302     reference mapper [0-1] container : Mapper;
303 }
304
305 -- @comments represents the tag 'delete':
306 -- deletes either a single file,
```

```
307 -- all files and sub-directories in a specified directory,
308 -- or a set of files specified by one or more FileSets
309 class Delete extends FileTask{
310     attribute file [0-1] : String;
311     attribute dir [0-1] : String;
312     attribute verbose [0-1] : String;
313     attribute quiet [0-1] : String;
314     attribute failonerror [0-1] : String;
315     attribute includeEmptyDirs [0-1] : String;
316     attribute includes [0-1] : String;
317     attribute includesfile [0-1] : String;
318     attribute excludes [0-1] : String;
319     attribute excludesfile [0-1] : String;
320     attribute defaultexcludes [0-1] : String;
321 }
322 -- @end fileTasks
323
324 -- @begin executionTasks
325 abstract class ExecutionTask extends PreDefinedTask{}
326
327 -- @comments represents the tag 'exec': executes a system command
328 class Exec extends ExecutionTask{
329     attribute executable : String;
330     attribute dir : String;
331 }
332 -- @end executionTasks
333 -- @end task
334 }
335
336 package PrimitiveTypes{
337     datatype String;
338 }
```

III. XML2Maven.atl file

```
1  module XML2Maven;
2  create OutMaven : MavenMaven, OutProject : MavenProject
3      from XML1 : XMLMaven, XML2 : XMLProject;
4
5  -- concerning the file representing maven.xml
6  -- helper : returns the value of the attribute 'name' of an element
7  -- the value must exist
8  helper context XMLMaven!Element def: getAttrVal(name: String): String =
9      self.children->
10         select(c | c.ocIsKindOf(XMLMaven!Attribute) and c.name = name)->
11             first().value;
12
13 -- helper : returns true if the attribute 'name' of an element has a value
14 helper context XMLMaven!Element def: testAttribute(name: String): Boolean =
15     not (self.children ->
16         select(d | d.ocIsKindOf(XMLMaven!Attribute) and d.name = name)->
17             first().ocIsUndefined());
18
19 -- helper : returns a value of the attribute 'name' of an element
20 -- returns '' if this attribute do not exist
21 helper context XMLMaven!Element def:getAttribute(name : String):String =
22     if (self.testAttribute(name))
23         then self.getAttrVal(name)
24         else ''
25     endif;
26
27
28 -----
29 -- concerning XmlIns
30
31 -- helper detXmlns: returns the value of the namespace:
32 -- it removes the prefix 'xmlns:'
33 helper context XMLMaven!Attribute def:detXmlns():String =
34     if self.name.size(>6
35         then self.name.substring(7,self.name.size())
36         else ''
37     endif;
38
39 helper def:testXmlnsAux(name: String): Boolean =
40     not (XMLMaven!Attribute.allInstances() ->
41         select(e|e.value=name)-> first().ocIsUndefined());
42
43 -- helper getXmlnsAux: returns the name of the attribute
44 -- whose value is given in parameter
45 helper def:getXmlnsAux(name: String): String =
46     if thisModule.testXmlnsAux(name)then
47         XMLMaven!Attribute.allInstances() ->
48         select(e|e.value=name)->first().name
49     else ''
50     endif;
51
52 -- helper getXmlns: returns the value of the prefix corresponding
53 -- to the library whose name is given in parameter
54 helper def:getXmlns(name: String): String =
55     let completeValue: String = thisModule.getXmlnsAux(name)in
56     if completeValue.size(>6
57         then completeValue.substring(7,completeValue.size())+';'
58         else ''
```

```
59     endif;
60     -----
61     -- central rule for MavenMaven
62     rule XMLMavenRoot2MavenMavenProject{
63         from i : XMLMaven!Root(
64             i.name = 'project'
65         )
66         to o : MavenMaven!Project(
67             xmlns <- i.children ->
68             select(d | if d.ocIsKindOf(XMLMaven!Attribute) then
69                 d.name.substring(1,6)='xmlns:'
70             else
71                 false
72             endif
73             ),
74             default <- XMLMaven!Element.allInstances() ->
75             select(d | d.name = 'goal'
76                 and d.getAttribute('name')=i.getAttribute('default'))->
77                 first(),
78             path <- i.children ->
79                 select(d | d.ocIsKindOf(XMLMaven!Element)
80                     and (d.name = thisModule.getXmlns('jelly:ant')+'path'))->
81                     first(),
82             properties <- i.children ->
83                 select(d | d.ocIsKindOf(XMLMaven!Element)
84                     and (d.name = thisModule.getXmlns('jelly:ant')+'property')),
85             taskdefs <- i.children ->
86                 select(d | d.ocIsKindOf(XMLMaven!Element)
87                     and (d.name = thisModule.getXmlns('jelly:ant')+'taskdef')),
88             prePostGoals <- i.children ->
89                 select(d | d.ocIsKindOf(XMLMaven!Element)
90                     and (d.name = 'preGoal' or d.name='postGoal')),
91             goals <- i.children ->
92                 select(d | d.ocIsKindOf(XMLMaven!Element)
93                     and d.name = 'goal')
94         )
95     }
96
97
98     rule Xmlns{
99         from i : XMLMaven!Attribute(
100             if i.parent.name='project'
101             and i.parent.ocIsKindOf(XMLMaven!Root)
102             and i.name.size()>5 then
103                 i.name.substring(1, 6) = 'xmlns:'
104             else
105                 false
106             endif
107         )
108         to o : MavenMaven!Xmlns(
109             name <- i.detXmlns(),
110             value <- i.value
111         )
112     }
113
114     -- properties
115     rule PropertyLocation{
116         from i : XMLMaven!Element(
117             i.name = thisModule.getXmlns('jelly:ant')+'property'
118             and i.testAttribute('location')
119         )
120         to o : MavenMaven!AntPropertyLocation(
```

```
121     name <- i.getAttribute('name'),
122     location <- i.getAttribute('location')
123   )
124 }
125
126 rule PropertyValue{
127   from i : XMLMaven!Element(
128     i.name = thisModule.getXmlns('jelly:ant')+'property'
129     and i.testAttribute('value')
130   )
131   to o : MavenMaven!AntPropertyValue(
132     name <- i.getAttribute('name'),
133     value <- i.getAttribute('value')
134   )
135 }
136 rule PropertyFile{
137   from i : XMLMaven!Element(
138     i.name = thisModule.getXmlns('jelly:ant')+'property'
139     and i.testAttribute('file')
140   )
141   to o : MavenMaven!AntPropertyFile(
142     file <- i.getAttribute('file')
143   )
144 }
145
146 rule PropertyEnv{
147   from i : XMLMaven!Element(
148     i.name = thisModule.getXmlns('jelly:ant')+'property'
149     and i.testAttribute('environment')
150   )
151   to o : MavenMaven!AntPropertyEnv(
152     environment <- i.getAttribute('environment')
153   )
154 }
155
156 rule JellySet{
157   from i : XMLMaven!Element(
158     i.name = thisModule.getXmlns('jelly:core')+'set'
159   )
160   to o : MavenMaven!JellySet(
161     var <- i.getAttribute('var'),
162     value <- i.getAttribute('value')
163   )
164 }
165
166 rule Goal{
167   from i : XMLMaven!Element(
168     i.name = 'goal'
169   )
170   to o : MavenMaven!Goal(
171     name <- i.getAttribute('name'),
172     contentsGoal <- i.children ->
173     select(d | d.ocIsKindOf(XMLMaven!Element))
174   )
175 }
176
177 rule PreGoal{
178   from i : XMLMaven!Element(
179     i.name = 'preGoal'
180   )
181   to o : MavenMaven!PreGoal(
182     centralGoal <- XMLMaven!Element.allInstances() ->
```



```
183         select(d|d.name='goal' and d.getAttribute('name')=i.getAttribute('name'))
184     ->
185         first(),
186     contentsGoal <- i.children ->
187         select(d | d.ocIsKindOf(XMLMaven!Element))
188     )
189 }
190
191 rule PostGoal{
192     from i : XMLMaven!Element(
193         i.name = 'postGoal'
194     )
195     to o : MavenMaven!PostGoal(
196         centralGoal <- XMLMaven!Element.allInstances() ->
197             select(d|d.name='goal'
198                 and d.getAttribute('name')=i.getAttribute('name')) ->
199                 first(),
200         contentsGoal <- i.children ->
201             select(d | d.ocIsKindOf(XMLMaven!Element))
202     )
203 }
204
205 rule AttainGoal{
206     from i : XMLMaven!Element(
207         i.name = 'attainGoal'
208     )
209     to o : MavenMaven!AttainGoal(
210         attainGoal <- XMLMaven!Element.allInstances() ->
211             select(d|d.name='goal' and
212                 d.getAttribute('name')=i.getAttribute('name')) ->
213                 first()
214     )
215 }
216
217
218 -- copy of tasks
219 -- task defined by the user
220 rule AntTaskDef{
221     from i : XMLMaven!Element(
222         i.name = thisModule.getXmlns('jelly:ant')+'taskdef'
223     )
224     to o : MavenMaven!AntTaskDef(
225         name <- i.getAttribute('name'),
226         classname <- i.getAttribute('classname')
227     )
228 }
229
230 rule NewTask{
231     from i : XMLMaven!Element(
232         not(XMLMaven!Element.allInstances() ->
233             select(d | d.name = thisModule.getXmlns('jelly:ant')+'taskdef'
234                 and d.getAttribute('name')=i.name) ->
235                 isEmpty())
236     )
237     to o : MavenMaven!NewTask(
238         taskName <- XMLMaven!Element.allInstances() ->
239             select(d | d.name = thisModule.getXmlns('jelly:ant')+'taskdef'
240                 and d.getAttribute('name')=i.name) ->
241                 first(),
242         attributes <- i.children ->
243             select(d | d.ocIsKindOf(XMLMaven!Attribute))
244     )

```

```
245 }
246
247 rule Attribut{
248   from i : XMLMaven!Attribute(
249     not(XMLMaven!Element.allInstances() ->
250       select(d | d.name = thisModule.getXmlns('jelly:ant')+ 'taskdef'
251         and d.getAttribute('name')=i.parent.name) ->
252         isEmpty())
253   )
254   to o : MavenMaven!Attribut(
255     name <- i.name,
256     value<- i.value
257   )
258 }
259
260 -- pre-defined tasks
261 rule Mkdir{
262   from i : XMLMaven!Element(
263     i.name = thisModule.getXmlns('jelly:ant')+ 'mkdir'
264   )
265   to o : MavenMaven!Mkdir(
266     dir <- i.getAttribute('dir')
267   )
268 }
269
270 rule Tstamp{
271   from i : XMLMaven!Element(
272     i.name = thisModule.getXmlns('jelly:ant')+ 'tstamp'
273   )
274   to o : MavenMaven!Tstamp()
275 }
276
277 rule Java{
278   from i : XMLMaven!Element(
279     i.name = thisModule.getXmlns('jelly:ant')+ 'java'
280   )
281   to o : MavenMaven!Java(
282     classname <- i.getAttribute('classname'),
283     jar <- i.getAttribute('jar'),
284     fork <- i.getAttribute('fork'),
285     classPath <- i.children ->
286     select(d | d.ocIsKindOf(XMLMaven!Element)
287       and (d.name = 'classpath' or d.name='ant:classpath'))
288   )
289 }
290
291 rule Javac{
292   from i : XMLMaven!Element(
293     i.name = thisModule.getXmlns('jelly:ant')+ 'javac'
294   )
295   to o : MavenMaven!Javac(
296     destdir <- i.getAttribute('destdir'),
297     srcdir <- i.getAttribute('srcdir'),
298     classPath <- i.children ->
299     select(d | d.ocIsKindOf(XMLMaven!Element)
300       and d.name = thisModule.getXmlns('jelly:ant')+ 'classpath' )->
301       first(),
302     inExcludes <- i.children ->
303     select(d | d.ocIsKindOf(XMLMaven!Element)
304       and (d.name = thisModule.getXmlns('jelly:ant')+ 'include' or
305         d.name = thisModule.getXmlns('jelly:ant')+ 'exclude'))
306   )

```

```
307 }
308
309 rule Javadoc{
310   from i : XMLMaven!Element(
311     i.name = thisModule.getXmlns('jelly:ant')+'javadoc'
312   )
313   to o : MavenMaven!Javadoc(
314     sourcepath <- i.getAttribute('sourcepath'),
315     destdir <- i.getAttribute('destdir'),
316     packagenames <- i.getAttribute('packagenames'),
317     defaultexcludes <- i.getAttribute('defaultexcludes'),
318     author <- i.getAttribute('author'),
319     version <- i.getAttribute('version'),
320     use <- i.getAttribute('use'),
321     windowtitle <- i.getAttribute('windowtitle')
322   )
323 }
324
325 rule Copy{
326   from i : XMLMaven!Element(
327     i.name = thisModule.getXmlns('jelly:ant')+'copy'
328   )
329   to o : MavenMaven!Copy(
330     todir <- i.getAttribute('todir'),
331     fileset <- i.children ->
332       select(d | d.ocIsKindOf(XMLMaven!Element)
333         and d.name = thisModule.getXmlns('jelly:ant')+'fileset') ->
334         first(),
335     filterset <- i.children ->
336       select(d | d.ocIsKindOf(XMLMaven!Element)
337         and d.name = thisModule.getXmlns('jelly:ant')+'filterset') ->
338         first()
339   )
340 }
341
342
343 rule Delete{
344   from i : XMLMaven!Element(
345     i.name = thisModule.getXmlns('jelly:ant')+'delete'
346   )
347   to o : MavenMaven!Delete(
348     dir <- i.getAttribute('dir')
349   )
350 }
351
352 rule Jar{
353   from i : XMLMaven!Element(
354     i.name = thisModule.getXmlns('jelly:ant')+'jar'
355   )
356   to o : MavenMaven!Jar(
357     jarfile <- i.getAttribute('jarfile'),
358     basedir <- i.getAttribute('basedir')
359   )
360 }
361
362 -- path, file and pattern
363 rule Path{
364   from i : XMLMaven!Element(
365     i.name = thisModule.getXmlns('jelly:ant')+'path'
366   )
367   to o : MavenMaven!Path(
368     id <- i.getAttribute('id'),
```

```
369     refid <- i.getAttribute('refid'),
370     fileset <- i.children ->
371     select(d | d.ocIsKindOf(XMLMaven!Element)
372           and d.name = thisModule.getXmlns('jelly:ant')+'fileset')
373   )
374 }
375
376 rule FileSet{
377   from i : XMLMaven!Element(
378     i.name = thisModule.getXmlns('jelly:ant')+'fileset'
379   )
380   to o : MavenMaven!FileSet(
381     dir <- i.getAttribute('dir'),
382     patternset <- i.children ->
383     select(d | d.ocIsKindOf(XMLMaven!Element)
384           and d.name = thisModule.getXmlns('jelly:ant')+'patternset'),
385     include <- i.children ->
386     select(d | d.ocIsKindOf(XMLMaven!Element)
387           and d.name = thisModule.getXmlns('jelly:ant')+'include'),
388     exclude <- i.children ->
389     select(d | d.ocIsKindOf(XMLMaven!Element)
390           and d.name = thisModule.getXmlns('jelly:ant')+'exclude')
391   )
392 }
393
394 rule PatternSet{
395   from i : XMLMaven!Element(
396     i.name = thisModule.getXmlns('jelly:ant')+'patternset'
397   )
398   to o : MavenMaven!PatternSet(
399     inexcludes <- i.children ->
400     select(d | d.ocIsKindOf(XMLMaven!Element)
401           and (d.name = thisModule.getXmlns('jelly:ant')+'exclude'
402              or d.name= thisModule.getXmlns('jelly:ant')+'include'))
403   )
404 }
405
406 rule ClassPath{
407   from i : XMLMaven!Element(
408     i.name = thisModule.getXmlns('jelly:ant')+'classpath'
409   )
410   to o : MavenMaven!ClassPath(
411     refid <- i.getAttribute('refid'),
412     pathElement <- i.children ->
413     select(d | d.ocIsKindOf(XMLMaven!Element)
414           and d.name = thisModule.getXmlns('jelly:ant')+'pathelement'),
415     fileset <- i.children ->
416     select(d | d.ocIsKindOf(XMLMaven!Element)
417           and d.name = thisModule.getXmlns('jelly:ant')+'fileset')
418   )
419 }
420
421 rule PathElement{
422   from i : XMLMaven!Element(
423     i.name = thisModule.getXmlns('jelly:ant')+'pathelement'
424   )
425   to o : MavenMaven!PathElement(
426     path <- i.getAttribute('path'),
427     location <- i.getAttribute('location')
428   )
429 }
430
```

```
431 rule FilterSet{
432   from i : XMLMaven!Element(
433     i.name = thisModule.getXmlns('jelly:ant')+'filterset'
434   )
435   to o : MavenMaven!FilterSet(
436     starttoken <- i.getAttribute('starttoken'),
437     endtoken <- i.getAttribute('endtoken'),
438     filter <- i.children ->
439       select(d | d.oclIsKindOf(XMLMaven!Element)
440         and d.name = thisModule.getXmlns('jelly:ant')+'filter'),
441     filtersfile <- i.children ->
442       select(d | d.oclIsKindOf(XMLMaven!Element)
443         and d.name = thisModule.getXmlns('jelly:ant')+'filtersfile')
444   )
445 }
446
447 rule Filter{
448   from i : XMLMaven!Element(
449     i.name = thisModule.getXmlns('jelly:ant')+'filter'
450   )
451   to o : MavenMaven!Filter(
452     token <- i.getAttribute('token'),
453     value <- i.getAttribute('value')
454   )
455 }
456
457 rule FiltersFile{
458   from i : XMLMaven!Element(
459     i.name = thisModule.getXmlns('jelly:ant')+'filtersfile'
460   )
461   to o : MavenMaven!FiltersFile(
462     file <- i.getAttribute('file')
463   )
464 }
465
466 rule Includes{
467   from i : XMLMaven!Element(
468     i.name = thisModule.getXmlns('jelly:ant')+'include'
469   )
470   to o : MavenMaven!Includes(
471     name <- i.getAttribute('name'),
472     ifCondition <- i.getAttribute('if'),
473     unless <- i.getAttribute('unless')
474   )
475 }
476
477 rule Excludes{
478   from i : XMLMaven!Element(
479     i.name = thisModule.getXmlns('jelly:ant')+'exclude'
480   )
481   to o : MavenMaven!Excludes(
482     name <- i.getAttribute('name'),
483     ifCondition <- i.getAttribute('if'),
484     unless <- i.getAttribute('unless')
485   )
486 }
487
488 rule IncludesFile{
489   from i : XMLMaven!Element(
490     i.name = thisModule.getXmlns('jelly:ant')+'includesfile'
491   )
492   to o : MavenMaven!IncludesFile(
```

```
493     name <- i.getAttribute('name'),
494     ifCondition <- i.getAttribute('if'),
495     unless <- i.getAttribute('unless')
496   )
497 }
498
499 rule ExcludesFile{
500   from i : XMLMaven!Element(
501     i.name = thisModule.getXmlns('jelly:ant')+'excludesfile'
502   )
503   to o : MavenMaven!ExcludesFile(
504     name <- i.getAttribute('name'),
505     ifCondition <- i.getAttribute('if'),
506     unless <- i.getAttribute('unless')
507   )
508 }
509
510
511 -----
512 -- concerning the file representing project.xml
513
514 -- helper : returns the value of a text belonging to an element
515 helper context XMLProject!Element def: getTextAux(name : String) : String =
516   self.children->
517     select(c | c.ocIsKindOf(XMLProject!Element) and c.name=name)
518     ->first().children
519     -> select(d | d.ocIsKindOf(XMLProject!Text))
520     ->first().value;
521
522 helper context XMLProject!Element def: testElement(name:String) : Boolean =
523   not (self.children ->
524     select(d | d.ocIsKindOf(XMLProject!Element) and d.name=name)->
525     first().ocIsUndefined());
526
527 helper context XMLProject!Element def:getText(name : String):String =
528   if (self.testElement(name))
529     then self.getTextAux(name)
530     else ''
531   endif;
532
533
534 -- helper : returns the value of the attribute 'name' of an element
535 -- the value must exist
536 helper context XMLProject!Element def: getAttrVal(name : String) : String =
537   self.children->
538     select(c | c.ocIsKindOf(XMLProject!Attribute) and c.name = name)
539     ->first().value;
540
541 -- helper : returns true if the attribute 'name' of an element has a value
542 helper context XMLProject!Element def: testAttribute(name : String) : Boolean =
543   not (self.children ->
544     select(d | d.ocIsKindOf(XMLProject!Attribute) and d.name = name)->
545     first().ocIsUndefined());
546
547 -- helper : returns a value of the attribute 'name' of an element
548 -- returns '' if this attribute do not exist
549 helper context XMLProject!Element def:getAttribute(name : String):String =
550   if (self.testAttribute(name))
551     then self.getAttrVal(name)
552     else ''
553   endif;
554
```

```
555 rule XMLProjectRoot2MavenProjectProject{
556     from i : XMLProject!Root
557     to o : MavenProject!Project(
558         id <- i.getAttribute('id'),
559         name <- i.getAttribute('name'),
560         description <- i.getText('description'),
561         build <- i.children ->
562             select(d | d.oclIsKindOf(XMLProject!Element) and d.name = 'build')
563             -> first()
564     )
565 }
566
567 rule Build{
568     from i : XMLProject!Element(
569         i.name = 'build'
570     )
571     to o : MavenProject!Build(
572         defaultGoal <- i.getText('defaultGoal'),
573         sourceDirectory <- i.getText('sourceDirectory')
574     )
575 }
```

IV. Maven2Ant.atl file

```
1  module Maven2Ant;
2  create OUT : Ant from INMaven : MavenMaven, INProject : MavenProject;
3
4  -- helpers for MavenMaven
5
6  -- helper which returns all Tasks concerning a goal :
7  -- the tasks obtained are those which are in preGoal,
8  -- then those in Goal and at last those in postGoal
9  helper context MavenMaven!Goal
10
11     def:getAllTasks():Sequence(MavenMaven!Task)=
12     if(self.preGoal.oclIsUndefined())
13     then if(self.postGoal.oclIsUndefined())
14     then self.getTasksAux()
15     else Sequence{self.getTasksAux(),self.postGoal.getTasksAux()}
16     endif
17     else if(self.postGoal.oclIsUndefined())
18     then Sequence{self.preGoal.getTasksAux(),self.getTasksAux()}
19     else Sequence{self.preGoal.getTasksAux(),
20     self.getTasksAux(),self.postGoal.getTasksAux()}
21     endif
22     endif;
23
24  helper context MavenMaven!AbstractGoal
25     def:getTasksAux():Sequence(MavenMaven!Task)=
26     self.contentsGoal -> select(e|e.oclIsKindOf(MavenMaven!Task));
27
28  -- helper which returns all attainGoal concerning a goal
29  -- (with preGoal and postGoal)
30  helper context MavenMaven!Goal
31
32     def:getAllAttainGoal():Sequence(MavenMaven!Goal)=
33     if(self.preGoal.oclIsUndefined())
34     then if(self.postGoal.oclIsUndefined())
35     then self.getAttainGoalAux()
36     else Sequence{self.getAttainGoalAux(),
37     self.postGoal.getAttainGoalAux()}
38     endif
39     else if(self.postGoal.oclIsUndefined())
40     then Sequence{self.preGoal.getAttainGoalAux(),
41     self.getAttainGoalAux()}
42     else Sequence{self.preGoal.getAttainGoalAux(),
43     self.getAttainGoalAux(),
44     self.postGoal.getAttainGoalAux()}
45     endif
46     endif;
47
48  helper context MavenMaven!AbstractGoal
49     def:getAttainGoalAux():Sequence(MavenMaven!Goal)=
50     self.contentsGoal ->
51     select(e|e.oclIsKindOf(MavenMaven!AttainGoal))->
52     collect(d|d.attainGoal);
53
54  -- RULE MavenProjects2AntProject
55  -- there are two elements in entry :
56  -- - MavenMaven!Project, the central element of the file representing
57  -- - MavenProject!Project, the central element of the file representing
58  -- project.xml
```



```
59  --      (defined in the 'using' part)
60  rule MavenProjects2AntProject{
61    from mm : MavenMaven!Project
62    using{
63      -- to have the second file in entry
64      mp : MavenProject!Project =
65          MavenProject!Project.allInstances()->
66          asSequence()->
67          first();
68      -- to obtain all properties (JellySet and AntProperties)
69      allJellySets : Sequence(MavenMaven!JellySet) =
70          MavenMaven!JellySet.allInstances()->
71          asSequence();
72      allProperties : Sequence(MavenMaven!AntProperty) =
73          MavenMaven!AntProperty.allInstances() ->
74          asSequence();
75      -- to obtain all taskdef (even those which are inside a goal)
76      allTaskDefs : Sequence(MavenMaven!AntTaskDef) =
77          MavenMaven!AntTaskDef.allInstances() ->
78          asSequence();
79    }
80    to a : Ant!Project(
81      name <- mp.name,
82      basedir <- mp.build.sourceDirectory,
83      default <- MavenMaven!Goal.allInstances()->
84          select (e|e.name=mp.build.defaultGoal) ->
85          first(),
86      -- if there are several properties or jellySet with the same value,
87      -- there are all represented
88      properties <- Sequence{allProperties,allJellySets},
89      path <- mm.path,
90      taskdef <- allTaskDefs,
91      targets <- mm.goals,
92      description <- mp.description
93    )
94  }
95
96  -----
97  -- concerning only MavenMaven (ie file maven.xml)
98  -- (all informations of MavenProject are extracted in the rule
99  -- MavenProjects2AntProject)
100
101  -----
102  -- properties
103
104  -- jellySet can be the equivalent of propertyName
105  -- but with ant, the value can not be changed
106  rule MavenMavenJellySet2PropertyName{
107    from mm : MavenMaven!JellySet
108    to a : Ant!PropertyName(
109      name <- mm.var,
110      value <- mm.value
111    )
112  }
113
114  rule MavenMavenPropertyValue2AntPropertyValue{
115    from m : MavenMaven!AntPropertyValue
116    to a : Ant!PropertyValue(
117      name <- m.name,
118      value <- m.value
119    )
120  }
```

```
121
122 rule MavenMavenPropertyLocation2AntPropertyLocation{
123   from m : MavenMaven!AntPropertyLocation
124   to a : Ant!PropertyLocation(
125     name <- m.name,
126     location <- m.location
127   )
128 }
129
130 rule MavenMavenAntPropertyFile2AntPropertyFile{
131   from m : MavenMaven!AntPropertyFile
132   to a : Ant!PropertyFile(
133     file <- m.file
134   )
135 }
136
137 rule MavenMavenAntPropertyEnv2AntPropertyEnv{
138   from m : MavenMaven!AntPropertyEnv
139   to a : Ant!PropertyEnv(
140     environment <- m.environment
141   )
142 }
143
144 -- rule for goals
145 rule MavenGoal2AntTarget{
146   from mm : MavenMaven!Goal
147   to a : Ant!Target(
148     name <- mm.name,
149     depends <- mm.getAllAttainGoal(),
150     tasks <- mm.getAllTasks()
151   )
152 }
153
154 -----
155 -- copy of task
156
157 -- tasks defined by the user
158 rule MavenMavenTaskDef2AntTaskDef{
159   from m : MavenMaven!AntTaskDef
160   to a : Ant!TaskDef(
161     name <- m.name,
162     classname <- m.classname
163   )
164 }
165
166 rule MavenMavenNewTask2AntNewTask{
167   from m : MavenMaven!NewTask
168   to a : Ant!NewTask(
169     taskName <- m.taskName,
170     attributes <- m.attributes
171   )
172 }
173
174 rule MavenMavenAttribut2AntAttribut{
175   from m : MavenMaven!Attribut
176   to a : Ant!Attribut(
177     name <- m.name,
178     value <- m.value
179   )
180 }
181
182 -- pre defined tasks
```



ATL TRANSFORMATION EXAMPLE

Maven to Ant

Date 05/08/2005

```
183 rule MavenMavenTstamp2AntTstamp{
184     from m : MavenMaven!Tstamp
185     to a : Ant!Tstamp()
186 }
187
188 rule MavenMavenMkdir2AntMkdir{
189     from m : MavenMaven!Mkdir
190     to a : Ant!Mkdir(
191         dir <- m.dir)
192 }
193
194 rule MavenMavenJava2AntJava{
195     from m : MavenMaven!Java
196     to a : Ant!Java(
197         classname <- m.classname,
198         jar <- m.jar,
199         fork <- m.fork,
200         classPath <- m.classPath
201     )
202 }
203
204 rule MavenMavenJavac2AntJavac{
205     from m : MavenMaven!Javac
206     to a : Ant!Javac(
207         destdir <- m.destdir,
208         srcdir <- m.srcdir,
209         classPath <- m.classPath,
210         inExcludes <- m.inExcludes
211     )
212 }
213
214 rule MavenMavenJavadoc2AntJavadoc{
215     from a : MavenMaven!Javadoc
216     to m : Ant!Javadoc(
217         sourcepath <- m.sourcepath,
218         destdir <- m.destdir,
219         packagenames <- m.packagenames,
220         defaultexcludes <- m.defaultexcludes,
221         author <- m.author,
222         version <- m.version,
223         use <- m.use,
224         windowtitle <- m.windowtitle
225     )
226 }
227
228 rule MavenMavenCopy2AntCopy{
229     from m : MavenMaven!Copy
230     to a : Ant!Copy(
231         todir <- m.todir,
232         fileset <- m.fileset,
233         filterset <- m.filterset
234     )
235 }
236
237
238 rule MavenMavenDelete2AntDelete{
239     from m : MavenMaven!Delete
240     to a : Ant!Delete(
241         dir <- m.dir)
242 }
243
244 rule MavenMavenJar2AntJar{
```

```
245     from m : MavenMaven!Jar
246     to a : Ant!Jar(
247         jarfile <- m.jarfile,
248         basedir <- m.basedir)
249 }
250
251 -----
252 -- path, pattern and filter
253 rule MavenMavenPath2AntPath{
254     from m : MavenMaven!Path
255     to a : Ant!Path(
256         id <- m.id,
257         refid <- m.refid,
258         fileset <- m.fileset,
259         path <- m.path,
260         pathElement <- m.pathElement
261     )
262 }
263
264 rule MavenMavenClassPath2AntClassPath{
265     from m : MavenMaven!ClassPath
266     to a : Ant!ClassPath(
267         refid <- m.refid,
268         pathElement <- m.pathElement,
269         fileset <- m.fileset
270     )
271 }
272
273
274
275 rule MavenMavenPathElement2AntPathElement{
276     from m : MavenMaven!PathElement
277     to a : Ant!PathElement(
278         path <- m.path,
279         location <- m.location
280     )
281 }
282
283 rule MavenMavenFileSet2AntFileSet{
284     from m : MavenMaven!FileSet
285     to a : Ant!FileSet(
286         dir <- m.dir,
287         patternset <- m.patternset,
288         include <- m.include,
289         exclude <- m.exclude
290     )
291 }
292
293 rule MavenMavenFilterSet2AntFilterSet{
294     from m : MavenMaven!FilterSet
295     to a : Ant!FilterSet(
296         starttoken <- m.starttoken,
297         endtoken <- m.endtoken,
298         filter <- m.filter,
299         filtersfile <- m.filtersfile
300     )
301 }
302
303 rule MavenMavenFilter2AntFilter{
304     from m : MavenMaven!Filter
305     to a : Ant!Filter(
306         token <- m.token,
```

```
307     value <- m.value
308   )
309 }
310
311 rule MavenMavenFiltersFile2AntFiltersFile{
312   from m : MavenMaven!FiltersFile
313   to a : Ant!FiltersFile(
314     file <- m.file
315   )
316 }
317
318 rule MavenMavenPatternset2AntPatternset{
319   from m : MavenMaven!PatternSet
320   to a : Ant!PatternSet(
321     inexcludes <- m.inexcludes
322   )
323 }
324
325
326 rule MavenMavenIncludes2AntIncludes{
327   from m : MavenMaven!Includes
328   to a : Ant!Includes(
329     name <- m.name,
330     ifCondition <- m.ifCondition,
331     unless <- m.unless
332   )
333 }
334
335 rule MavenMavenExcludes2AntExcludes{
336   from m : MavenMaven!Excludes
337   to a : Ant!Excludes(
338     name <- m.name,
339     ifCondition <- m.ifCondition,
340     unless <- m.unless
341   )
342 }
343
344 rule MavenMavenIncludesFile2AntIncludesFile{
345   from m : MavenMaven!IncludesFile
346   to a : Ant!IncludesFile(
347     name <- m.name,
348     ifCondition <- m.ifCondition,
349     unless <- m.unless
350   )
351 }
352
353 rule MavenMavenExcludesFile2AntExcludesFile{
354   from m : MavenMaven!ExcludesFile
355   to a : Ant!ExcludesFile(
356     name <- m.name,
357     ifCondition <- m.ifCondition,
358     unless <- m.unless
359   )
360 }
```

V. Ant2XML.atl file

```
1  module Ant2XML;
2  create OUT : XML from IN : Ant;
3
4  -- concatene a list of String
5  -- the elements are separated by a comma
6  helper def: concat(list : Sequence(String)) : String =
7      list -> asSet() -> iterate(element ;acc : String = '' |
8          acc +
9              if acc = ''
10             then element
11             else ',' + element
12             endif);
13
14 -- rule for a project having a description
15 rule Project2Root{
16     from i : Ant!Project(
17         if i.description.oclIsUndefined()
18         then false
19         else not(i.description='')
20         endif
21     )
22     to o : XML!Root(
23         name <- 'project',
24         children <- Sequence {itsName,itsDescription,itsBasedir,
25                             itsDefaultTarget,i.properties,
26                             i.path,i.taskdef,i.targets}
27     ),
28     itsName : XML!Attribute(
29         name <- 'name',
30         value <- i.name
31     ),
32     itsDescription : XML!Element(
33         name <- 'description',
34         children <- textText
35     ),
36     textText : XML!Text(
37         value <- i.description
38     ),
39     itsBasedir : XML!Attribute(
40         name <- 'basedir',
41         value <- i.basedir
42     ),
43     itsDefaultTarget : XML!Attribute(
44         name <- 'default',
45         value <- i.default.name
46     )
47 }
48
49 -- rule for a project without description
50 rule Project2RootWithoutDescription{
51     from i : Ant!Project(
52         if i.description.oclIsUndefined()
53         then true
54         else i.description=''
55         endif
56     )
57     to o : XML!Root(
58         name <- 'project',
```

```
59     children <- Sequence {itsName,itsBasedir,itsDefaultTarget,
60                           i.properties,i.path,i.taskdef,i.targets}
61   ),
62   itsName : XML!Attribute(
63     name <- 'name',
64     value <- i.name
65   ),
66   itsBasedir : XML!Attribute(
67     name <- 'basedir',
68     value <- i.basedir
69   ),
70   itsDefaultTarget : XML!Attribute(
71     name <- 'default',
72     value <- i.default.name
73   )
74 }
75
76 -----
77 -- properties
78 rule PropertyValue{
79   from i : Ant!PropertyValue
80   to o : XML!Element(
81     name <- 'property',
82     children <- Sequence{propertyName2,propertyValue}
83   ),
84   propertyName2 : XML!Attribute(
85     name <- 'name',
86     value <- i.name
87   ),
88   propertyValue : XML!Attribute(
89     name <- 'value',
90     value <- i.value
91   )
92 }
93
94 rule PropertyLocation{
95   from i : Ant!PropertyLocation
96   to o : XML!Element(
97     name <- 'property',
98     children <- Sequence{propertyName2,propertyLocation}
99   ),
100   propertyName2 : XML!Attribute(
101     name <- 'name',
102     value <- i.name
103   ),
104   propertyLocation : XML!Attribute(
105     name <- 'location',
106     value <- i.location
107   )
108 }
109
110 rule PropertyFile{
111   from i : Ant!PropertyFile
112   to o : XML!Element(
113     name <- 'property',
114     children <- nameFile
115   ),
116   nameFile : XML!Attribute(
117     name <- 'file',
118     value <- i.file
119   )
120 }
```

```
121
122 rule PropertyEnv{
123   from i : Ant!PropertyEnv
124   to o : XML!Element(
125     name <- 'property',
126     children <- environmentName
127   ),
128   environmentName : XML!Attribute(
129     name <- 'environment',
130     value <- i.environment
131   )
132 }
133
134 -----
135 -- target
136 rule TargetWithDescription{
137   from i : Ant!Target(
138     if i.description.oclIsUndefined()
139       then false
140       else not (i.description='')
141     endif
142   )
143   to o : XML!Element(
144     name <- 'target',
145     children <- Sequence{nameAttribute,descriptionElement,
146                          dependsAttribute,i.tasks}
147   ),
148   nameAttribute : XML!Attribute(
149     name <- 'name',
150     value <- i.name
151   ),
152   descriptionElement : XML!Element(
153     name <- 'description',
154     children <- descriptionText
155   ),
156   descriptionText : XML!Text(
157     value <- i.description
158   ),
159   dependsAttribute : XML!Attribute(
160     name <- 'depends',
161     value <- thisModule.concat(i.depends -> collect(e|e.name))
162   )
163 }
164
165 rule TargetWithoutDescription{
166   from i : Ant!Target(
167     if i.description.oclIsUndefined()
168       then true
169       else i.description=''
170     endif
171   )
172   to o : XML!Element(
173     name <- 'target',
174     children <- Sequence{nameAttribute,dependsAttribute,i.tasks}
175   ),
176   nameAttribute : XML!Attribute(
177     name <- 'name',
178     value <- i.name
179   ),
180   dependsAttribute : XML!Attribute(
181     name <- 'depends',
182     value <- thisModule.concat(i.depends -> collect(e|e.name))
```




```
183     )
184   }
185
186   -----
187   -- tasks
188
189   -- task defined by the user
190   -- taskdef (definition of the task)
191   rule TaskDef{
192     from i : Ant!TaskDef
193     to o : XML!Element(
194       name <- 'taskdef',
195       children <- Sequence{nameName,nameClassName}
196     ),
197     nameName : XML!Attribute(
198       name <- 'name',
199       value <- i.name
200     ),
201     nameClassName : XML!Attribute(
202       name <- 'classname',
203       value <- i.classname
204     )
205   }
206
207   rule NewTask{
208     from i : Ant!NewTask
209     to o : XML!Element(
210       name <- i.taskName.name,
211       children <- i.attributes
212     )
213   }
214
215   rule Attribut{
216     from i : Ant!Attribut
217     to o : XML!Attribute(
218       name <- i.name,
219       value <- i.value
220     )
221   }
222
223   -- pre-defined tasks
224   rule Tstamp{
225     from i : Ant!Tstamp
226     to o : XML!Element(
227       name <- 'tstamp'
228     )
229   }
230
231   rule Mkdir{
232     from i : Ant!Mkdir
233     to o : XML!Element(
234       name <- 'mkdir',
235       children <- dirAttribute
236     ),
237     dirAttribute : XML!Attribute(
238       name <- 'dir',
239       value <- i.dir
240     )
241   }
242
243   rule Javac{
244     from i : Ant!Javac
```


```
245     to o : XML!Element(  
246         name <- 'javac',  
247         children <- Sequence{sourceDirAttribute,destDirAttribute,  
248             i.inExcludes,i.classPath}  
249     ),  
250     sourceDirAttribute : XML!Attribute(  
251         name <- 'srcdir',  
252         value <- i.srcdir  
253     ),  
254     destDirAttribute : XML!Attribute(  
255         name <- 'destdir',  
256         value <- i.destdir  
257     )  
258 }  
259  
260 rule Copy{  
261     from i : Ant!Copy  
262     to o : XML!Element(  
263         name <- 'copy',  
264         children <- Sequence{toDirAttribute,i.fileset}  
265     ),  
266     toDirAttribute : XML!Attribute(  
267         name <- 'todir',  
268         value <- i.todir  
269     )  
270 }  
271  
272  
273 rule Exec{  
274     from i : Ant!Exec  
275     to o : XML!Element(  
276         name <- 'exec',  
277         children <- execAttribute  
278     ),  
279     execAttribute : XML!Attribute(  
280         name <- 'executable',  
281         value <- i.executable  
282     )  
283 }  
284  
285 rule Echo{  
286     from i : Ant!Echo  
287     to o : XML!Element(  
288         name <- 'echo',  
289         children <- echoAttribute  
290     ),  
291     echoAttribute : XML!Attribute(  
292         name <- 'message',  
293         value <- i.message  
294     )  
295 }  
296 -----  
297 -- path  
298  
299 -- this takes only the attribute 'id' (not 'refid')  
300 rule Path{  
301     from i : Ant!Path  
302     to o : XML!Element(  
303         name <- 'path',  
304         children <- Sequence{idAttribute,i.fileset,i.path,i.pathElement}  
305     ),  
306     idAttribute : XML!Attribute(  

```

```
307     name <- 'id',
308     value <- i.id
309   )
310 }
311
312 rule ClassPath{
313   from i : Ant!ClassPath
314   to o : XML!Element(
315     name <- 'classpath',
316     children <- refidAttribute),
317   refidAttribute : XML!Attribute(
318     name <- 'refid',
319     value <- i.refid
320   )
321 }
322
323 rule Fileset{
324   from i : Ant!FileSet
325   to o : XML!Element(
326     name <- 'fileset',
327     children <- Sequence{dirAttribute,i.patternset,i.include,i.exclude}
328   ),
329   dirAttribute : XML!Attribute(
330     name <- 'dir',
331     value <- i.dir
332   )
333 }
334
335 rule PathElement{
336   from i : Ant!PathElement
337   to o : XML!Element(
338     name <- 'pathelement'
339   )
340 }
341
342 rule PatternSet{
343   from i : Ant!PatternSet
344   to o : XML!Element(
345     name <- 'patternset',
346     children <- i.inexcludes
347   )
348 }
349
350 rule Include{
351   from i : Ant!Includes
352   to o : XML!Element(
353     name <- 'include',
354     children <- nameAttribute
355   ),
356   nameAttribute : XML!Attribute(
357     name <- 'name',
358     value <- i.name
359   )
360 }
361
362 rule Exclude{
363   from i : Ant!Excludes
364   to o : XML!Element(
365     name <- 'exclude',
366     children <- nameAttribute
367   ),
368   nameAttribute : XML!Attribute(
```

	ATL TRANSFORMATION EXAMPLE	
	Maven to Ant	Date 05/08/2005

```
369     name <- 'name',  
370     value <- i.name  
371   )  
372 }
```

	ATL TRANSFORMATION EXAMPLE	
	Maven to Ant	Date 05/08/2005

References

- [1] Maven Overview. <http://maven.apache.org/reference/project-descriptor.html>
- [2] Ant Overview. <http://ant.apache.org/manual/>
- [3] KM3: Kernel MetaMetaModel. <http://dev.eclipse.org/viewcvs/indextech.cgi/~checkout~/gmt-home/doc/atl/index.html>.