	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

1. ATL Transformation Example

1.1. Example: UMLDI → SVG

The UMLDI to SVG example describes a transformation from a UML diagram, that has its presentation information according to the UML Diagram Interchange standard [1], to its presentation using the W3C standard SVG (Scalable Vector Graphics), an XML-based format [2].

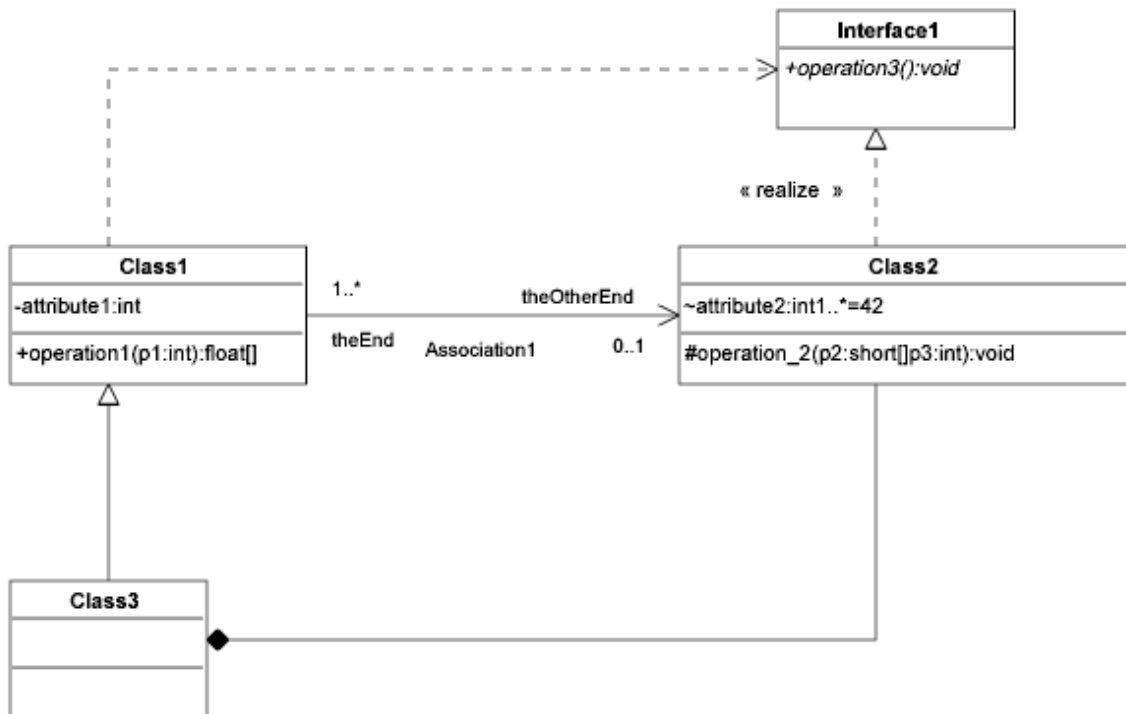
1.1.1. Transformation overview


The aim of the transformation is to generate a graphical representation of a model, and only this. In UMLDI both content and placement are stored in the xmi file. Here, the idea is to create a view of the model, visualization-based, and not data-based.

The SVG file may be exported towards another graphical format, or viewed as is.

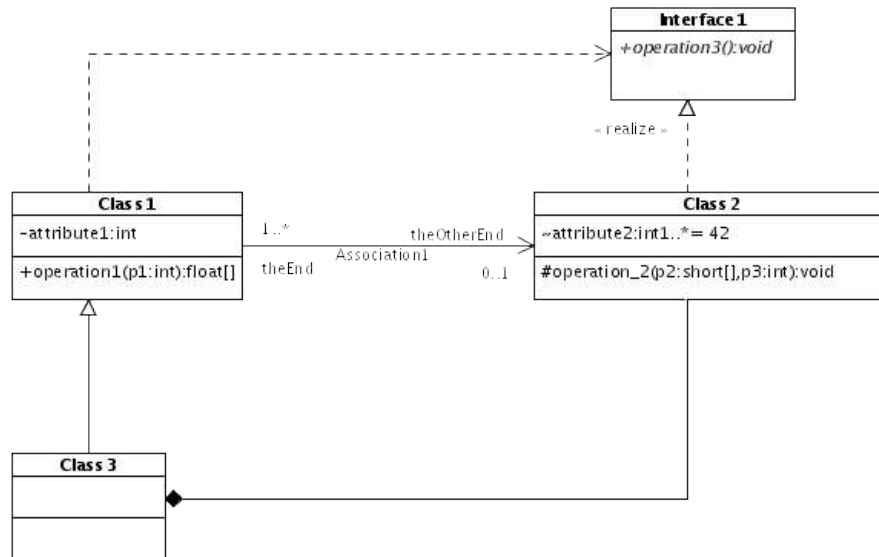
Moreover, the SVG code in the created file shall be readable by a human-being, as opposed to the output of Poseidon, for instance, that generates SVG files that are based on non abstract geometrical forms (such as rectangle, line, circle, polygon, etc). The transformation developed here shall create logical structures using as many as possible abstract geometrical forms.

The example of UML diagram (with DI) used is shown below :



	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

And here is its appearance when transformed into SVG :



1.2. Metamodels


This transformation is based on the UMLDI metamodel, that can be found in chapter 8 : Metamodel Extension, of the UMLDI specification [1].

On the other hand, it is based on a simplified SVG metamodel, centered on model rendering. Moreover, it is designed to build the SVG model the way a human designer would have, rather than a computer optimized one. Thus, for example, only G (groups) are children of Svg. That means when a class or an association is created, it is put in a group containing its name, etc. Consequently, the output is structured in a logical way.


1.3. Rules specification

These are the rules to transform a UMLDI class diagram model to a SVG model.

- For the Diagram element, a Svg element is created, having as children the elements contained by the Diagram.

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

- For each GraphNode that is linked to a class or interface, the following elements are created :
 - a Rect element, that is the external shape of the class/interface.
 - a Line that separates the name of the class/interface from the rest
 - a Text element : the name of the class/interface
 - a G element that will contain the attributes.
 - a Line that separates attributes from operations
 - a G element that will contain the operations.
- For each attribute (GraphNode), a text element is created with the following content
 - its visibility
 - its name
 - its parameters (with their types)
 - its type
 - its multiplicity
 - its initial value
- For each operation(GraphNode), a text element is created with the following content
 - its visibility
 - its name
 - its parameters (with their types)
 - its type
- For each GraphEdge, a G element is created containing :
 - a Polyline, that draws the shape of the relation
 - Text elements :
 - the name
 - the multiplicities
 - the roles
 - a Marker element, that draws the arrow head, or container head, etc

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

- For all these elements, their relative position (in UMLDI) is converted to their absolute position. As in SVG no negative coordinates are allowed, that is checked and if necessary the whole diagram is given an offset.


1.4. ATL code

This code consists in 27 helpers and 15 rules. Among the helpers, offset computes the minimum X and Y coordinates of the UMLDI model and adds a margin. It is used by the `getAbsoluteX` and `getAbsoluteY` helpers to return the absolute coordinates of elements, corrected with the offset.

In the other helpers, there are those that return things, and the boolean ones. The latter are used to create elements with the appropriate properties, while the first ones are used in the body of the rules, to return elements in a better way.

The rules are laid out this way :

- the `Diagram2SVG` rule.
- the classes and interfaces creation rules.
- the attributes and operations rules
- the relations rules (`GraphEdges`)

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```


module UMLDI2SVG;
create OUT: SVG from IN: UMLDI;

-- HELPERS
  -- Boolean Helpers

      -- HELPER isOfKind
      -- Returns true if the node is of kind (OCL way)
of the input
      -- For example if the element is a UML Class
      -- CONTEXT:    UMLDI!GraphElement
      -- IN:         kind: String
      -- RETURN:     Boolean
helper context UMLDI!GraphElement def: isOfKind(kind:
String): Boolean =
  if self.semanticModel.oclIsKindOf
(UMLDI!Uml1SemanticModelBridge) then
    self.semanticModel.element.oclIsKindOf(kind)
  else
    false
  endif;

      -- HELPER isOfType
      -- Returns true if the element is of the input
type
      -- CONTEXT:    UMLDI!GraphNode
      -- IN:         N/A
      -- RETURN:     Boolean
helper context UMLDI!GraphNode def: isOfType(testType:
String): Boolean =
  if self.semanticModel.oclIsKindOf
(UMLDI!SimpleSemanticModelElement) then
    self.semanticModel.typeInfo = testType
  else
    false
  endif;

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005


```

-- HELPER hasAttributes
-- Returns true if the current class node has
some attributes
    -- CONTEXT:    UMLDI!GraphNode
    -- IN:         N/A
    -- RETURN:     Boolean
    -- CALLS:      isOfType
helper context UMLDI!GraphNode def: hasAttributes():
Boolean =
    if self.contained->exists( e | e.isOfType
('AttributeCompartment')) then
        self.contained->select( e | e.isOfType
('AttributeCompartment'))->
            first().contained->exists( e | e.isOfType
('DelimitedSection'))
    else
        false
    endif;

-- HELPER hasMethods
-- Returns true if the current class node has
some methods
    -- CONTEXT:    UMLDI!GraphNode
    -- IN:         N/A
    -- RETURN:     Boolean
    -- CALLS:      isOfType
helper context UMLDI!GraphNode def: hasMethods(): Boolean =
    if self.contained->exists( e | e.isOfType
('OperationCompartment')) then
        self.contained->select( e | e.isOfType
('OperationCompartment'))->
            first().contained->exists( e | e.isOfType
('DelimitedSection'))
    else
        false
    endif;

-- HELPER hasAttSeparator

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005

```

-- Returns true if the current class node has a
seaparator between the
-- attributes and methods compartments
-- CONTEXT:    UMLDI!GraphNode
-- IN:         N/A
-- RETURN:     Boolean
-- CALLS:      isOfType

helper context UMLDI!GraphNode def: hasAttSeparator():
Boolean =
  self.contained->select( e | e.isOfType
('CompartmentSeparator'))->size() = 2;

-- HELPER isNamed
-- Checks whether the association is named.
-- CONTEXT:    UMLDI!GraphEdge
-- IN:         N/A
-- RETURN:     Boolean


helper context UMLDI!GraphEdge def: isNamed(): Boolean =
  not self.semanticModel.element.name.oclIsUndefined();

helper context UMLDI!GraphEdge def: isStereotypeNamed():
Boolean =
  not self.contained->select( e |
  e .isOfType('StereotypeCompartment'))->
  first().contained->select( e |
  e.isOfKind(UMLDI!Stereotype))->
  first().
semanticModel.element.name.oclIsUndefined();

-- HELPER hasLeftRole
-- Checks whether the association has a role on
-- its first connection.
-- CONTEXT:    UMLDI!GraphEdge
-- IN:         N/A
-- RETURN:     Boolean

helper context UMLDI!GraphEdge def: hasLeftRole(): Boolean
=
  not self.semanticModel.element.connection->
  first().name.oclIsUndefined();

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```

-- HELPER hasRightRole
-- Checks whether the association has a role on
-- its last connection.
-- CONTEXT:    UMLDI!GraphEdge
-- IN:         N/A
-- RETURN:     Boolean

helper context UMLDI!GraphEdge def: hasRightRole(): Boolean
=
    not self.semanticModel.element.connection->
      last().name.oclIsUndefined();


-- HELPER hasLeftMultiplicity
-- Checks whether the association has a
multiplicity fifferent
-- from 1-1 on it first connection.
-- CONTEXT:    UMLDI!GraphEdge
-- IN:         N/A
-- RETURN:     Boolean

helper context UMLDI!GraphEdge def: hasLeftMultiplicity():
Boolean =
    self.semanticModel.element.connection->first().
multiplicity.range->
    asSequence()->first().lower <> 1 or
    self.semanticModel.element.connection->first().
multiplicity.range->
    asSequence()->last().upper <> 1;

-- HELPER hasRightMultiplicity
-- Checks whether the association has a
multiplicity fifferent
-- from 1-1 on it last connection.
-- CONTEXT:    UMLDI!GraphEdge
-- IN:         N/A
-- RETURN:     Boolean

helper context UMLDI!GraphEdge def: hasRightMultiplicity():
Boolean =
    self.semanticModel.element.connection->last().
multiplicity.range->

```


	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005


```

    asSequence()->first().lower <> 1 or
    self.semanticModel.element.connection->last().
multiplicity.range->
    asSequence()->last().upper <> 1;

-- HELPER hasMarkerStart
-- Checks whether the association has a marker,
-- ie when it is either a one way association or
-- a composition or aggregation one (or maybe
both), and
-- that it should be on the first side of the
association.
-- CONTEXT:    UMLDI!GraphEdge
-- IN:         N/A
-- RETURN:     Boolean
helper context UMLDI!GraphEdge def: hasMarkerStart():
Boolean =
    (self.semanticModel.element.connection->first().
isNavigable and
    not(self.semanticModel.element.connection->last().
isNavigable))
    or self.semanticModel.element <> 'none';

-- HELPER hasMarkerEnd
-- Checks whether the association has a marker,
-- ie when it is either a one way association or
-- a composition or aggregation one (or maybe
both), and
-- that it should be on the last side of the
association.
-- CONTEXT:    UMLDI!GraphEdge
-- IN:         N/A
-- RETURN:     Boolean
helper context UMLDI!GraphEdge def: hasMarkerEnd(): Boolean
=
    (not(self.semanticModel.element.connection->first().
isNavigable) and
    self.semanticModel.element.connection->last().
isNavigable)

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```

or self.semanticModel.element <> 'none';


-- End Boolean Helpers

-- Coordinates Helpers

    -- HELPER getAbsoluteNoOffsetX
    -- Returns the absolute horizontal coordinate
computed by
    -- getting the position of the current node and
by
    -- recursively adding it to its parent's (XML
way)
        -- CONTEXT:    UMLDI!GraphElement
        -- IN:          N/A
        -- RETURN:     Real
helper context UMLDI!GraphElement def: getAbsoluteNoOffsetX
(): Real =
    if not self.oclIsKindOf(UMLDI!Diagram) then
        self.position.x +
self.container.getAbsoluteNoOffsetX()
    else
        self.position.x
    endif;

    -- HELPER getAbsoluteNoOffsetY
    -- Returns the absolute vertical coordinate
computed by
    -- getting the position of the current node and
by
    -- recursively adding it to its parent's (XML
way)
        -- CONTEXT:    UMLDI!GraphElement
        -- IN:          N/A
        -- RETURN:     Real
helper context UMLDI!GraphElement def: getAbsoluteNoOffsetY
(): Real =
    if not self.oclIsKindOf(UMLDI!Diagram) then

```


	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```

        self.position.y +
self.container.getAbsoluteNoOffsetY()
    else
        self.position.y
    endif;

    -- HELPER offset
    -- Returns a tuple containing the smallest
absolute horizontal
    -- coordinate and the smallest vertical one. This
helper is
    -- computed only once. 0 is added so that the
minimum is
    -- always negative or null.
    -- Another offset called viewOffset is added so
as to have some
    -- margin
    -- CONTEXT:    thisModule
    -- IN:         N/A
    -- RETURN:     Tuple (minX: Real, minY:
Real)
    -- CALLS:     getAbsoluteNoOffsetX
    --           getAbsoluteNoOffsetY
helper def: offset: TupleType (minX: Real, minY: Real) =
    let leaves: Sequence(UMLDI!GraphElement) =
        UMLDI!GraphElement.allInstances()->select( e |
e.contained->
            isEmpty()) in
    let setMinX: Set (Real) = leaves->iterate( e; acc: Set
(Real) =
        Set {} |
        acc->including(e.getAbsoluteNoOffsetX())) in
    let setMinY: Set (Real) = leaves->iterate( e; acc: Set
(Real) =
        Set {} |
        acc->including(e.getAbsoluteNoOffsetY())) in
    let viewOffset: Real = 20.0 in
    Tuple { x = setMinX->iterate(e; acc: Real = 0 |

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```

if e < acc then e else acc endif) - viewOffset,
y = setMinY->iterate(e; acc: Real = 0 |
if e < acc then e else acc endif) - viewOffset };

-- HELPER getAbsoluteX
-- Returns the absolute X coordinate corrected
with the X offset
-- CONTEXT:    UMLDI!GraphElement
-- IN:         N/A
-- RETURN:     Real
-- CALLS:     getAbsoluteNoOffsetX
--            offset

helper context UMLDI!GraphElement def: getAbsoluteX(): Real
=
    self.getAbsoluteNoOffsetX() - thisModule.offset.x;

-- HELPER getAbsoluteY
-- Returns the absolute Y coordinate corrected
with the Y offset
-- CONTEXT:    UMLDI!GraphElement
-- IN:         N/A
-- RETURN:     Real
-- CALLS:     getAbsoluteNoOffsetY
--            offset


helper context UMLDI!GraphElement def: getAbsoluteY(): Real
=
    self.getAbsoluteNoOffsetY() - thisModule.offset.y;

-- End coordinates helpers

-- HELPER diagram
-- Returns the diagram element of the UMLDI model
-- CONTEXT:    thisModule
-- IN:         N/A
-- RETURN:     UMLDI!Diagram
helper def: diagram: UMLDI!Diagram =
    UMLDI!Diagram.allInstances()->asSequence()->first();

-- UMLDI information helpers

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005


```

-- HELPER mult
-- Checks whether the attribute has a
multiplicity. If so,
-- proceeds to getMultiplicity.
-- CONTEXT:    UMLDI!GraphNode
-- IN:         N/A
-- RETURN:     String (multiplicity)
-- CALLS:      getMultiplicity
helper context UMLDI!GraphNode def: mult(): String =
  if
self.semanticModel.element.multiplicity.oclIsUndefined()
then
  ''
else

self.semanticModel.element.multiplicity.getMultiplicity()
endif;

-- HELPER getMultiplicity
-- Returns either an empty string (multiplicity
1-1) or
-- the corresponding multiplicity.
-- CONTEXT:    UMLDI!Multiplicity
-- IN:         N/A
-- RETURN:     String
helper context UMLDI!Multiplicity def: getMultiplicity():
String =
  let lower: String = self.range->asSequence()->first().
lower in
  let upper: String = self.range->asSequence()->last().
upper in
  if lower = 0 and upper = 0-1 then
    '*'
  else
    if lower = 0 then
      lower.toString() + '..' + upper.toString()
    else

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005


```

    if upper = 0-1 then
      lower.toString() + '..' + '*'
    else
      lower.toString() + '..' + upper.toString
  ()
    endif
  endif
endif;

-- HELPER getInitialValue
-- Returns if it exists the initial value for the
caller
-- attribute.
-- CONTEXT:    UMLDI!GraphNode
-- IN:         N/A
-- RETURN:     String
helper context UMLDI!GraphNode def: getInitialValue():
String =
  let elt: UMLDI!Attribute = self.semanticModel.element
in
  if not elt.initialValue.oclIsUndefined() then
    if elt.initialValue <> '' then
      '= ' + elt.initialValue.body
    else
      ''
    endif
  else
    ''
  endif;

-- HELPER getVisibility
-- Returns the visibility of the current
attribute/operation.
-- CONTEXT:    UMLDI!GraphNode
-- IN:         N/A
-- RETURN:     String
helper context UMLDI!GraphNode def: getVisibility(): String
=

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005


```

    let visibility: String =
self.semanticModel.element.visibility in
    if visibility = #vk_public then
        '+'
    else
        if visibility = #vk_package then
            '~'
        else if visibility = #vk_private then
            '-'
        else
            '#'
        endif
    endif
endif;

-- HELPER getParameters
-- Returns the parameters of the current
attribute/operation,
-- with their type.
-- CONTEXT:    UMLDI!GraphNode
-- IN:         N/A
-- RETURN:     String

helper context UMLDI!GraphNode def: getParameters(): String
=
    let element: UMLDI!Operation =
self.semanticModel.element in
    let end: Integer = element.parameter->size() in
    if end <> 1 then
        element.parameter->iterate( e; acc: String = '' |
            if e.kind <> #pdk_return then
                if e.name = element.parameter->last().
name then
                    acc + e.name + ':' + e.type.name
                else
                    acc + e.name + ':' + e.type.name +
', '
            endif

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```


        else
            ''
        endif)
    else
        ''
    endif;

    -- HELPER getReturnParameter
    -- Returns the return type of the current
operation.
        -- CONTEXT:    UMLDI!GraphNode
        -- IN:          N/A
        -- RETURN:     String
helper context UMLDI!GraphNode def: getReturnParameter():
String =
    self.semanticModel.element.parameter->
        select( p | p.kind = #pdk_return)->first().
type.name;

    -- HELPER getName
    -- Returns the name of the current association.
        -- CONTEXT:    UMLDI!GraphEdge
        -- IN:          N/A
        -- RETURN:     String
helper context UMLDI!GraphEdge def: getName(): String =
    if self.isNamed() then
        self.semanticModel.element.name
    else
        ''
    endif;

    -- HELPER getMarker
    -- Returns the marker type for this part of the
association
    -- or 'none' if there is none.
        -- CONTEXT:    UMLDI!GraphNode
        -- IN:          position: String (start|
end)
        -- RETURN:     String


```


	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005

```

helper context UMLDI!GraphEdge def: getMarker(position:
String): String =
  let element: UMLDI!Association =
    self.semanticModel.element in
  let connection1: UMLDI!AssociationEnd =
    element.connection->first() in
  let connection2: UMLDI!AssociationEnd =
    element.connection->last() in
  if position = 'start' then
    if self.hasMarkerStart() then
      if connection1.isNavigable and
        not connection2.isNavigable then
        'url(#Association)'
      else
        if connection1.aggregation =
#ak_composite then
          'url(#Composition)'
        else
          if connection1.aggregation =
#ak_none then
            'none'
          else
            'url(#Aggregation)'
          endif
        endif
      endif
    else
      'none'
    endif
  else
    if self.hasMarkerEnd() then
      if not(connection1.isNavigable) and
        connection2.isNavigable then
        'url(#Association)'
      else

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005


```

        if connection2.aggregation =
#ak_composite then
            'url(#Composition)'
        else
            if connection2.aggregation =
#ak_none then
                'none'
            else
                'url(#Aggregation)'
            endif
        endif
    endif
else
    'none'
endif
endif;

-- End Helpers

-- Rules
-- RULE Diagram2SVG
-- Creates the SVG element with its definition element
-- which contains all the different marker symbols
that
-- can be found in a class diagram. That part is
static,
-- whereas size and position are computed.
rule Diagram2SVG {
    from
        d: UMLDI!Diagram
    to
        out: SVG!Svg (
            namespace <- 'http://www.w3.org/2000/svg',
            version <- '1.0',
            position <- abs,
            size <- dim,
            children <- Sequence {definitions}
        ),


```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005

```

abs: SVG!AbsoluteCoord (
  x <- d.getAbsoluteX(),
  y <- d.getAbsoluteY()
),
dim: SVG!Dimension (
  width <- d.size.width,
  height <- d.size.height
),
definitions: SVG!Defs (
  groupContent <-
    Sequence {Association,
Generalization, Dependency,
      Aggregation, Composition}
),
  Association: SVG!Marker (
    identifier <- 'Association',
    refX <- 10.0,
    refY <- 5.0,
    markerWidth <- 11.0,
    markerHeight <- 11.0,
    orient <- 'auto',
    fill <- 'none',
    viewBox <- '0 0 10 10',
    drawing <- associationPathGroup
  ),
  associationPathGroup: SVG!G (
    groupContent <- associationPath
  ),
    associationPath: SVG!Path (
      stroke <- 'black',
      d <- 'M 0 0 L 10 5 L 0 10'
    ),
  Generalization: SVG!Marker (
    identifier <- 'Generalization',
    refX <- 10.0,
    refY <- 5.0,
    markerWidth <- 11.0,
    markerHeight <- 11.0,
    orient <- 'auto',


```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005

```

fill <- 'white',
viewBox <- '0 0 10 10',
drawing <- generalizationPathGroup
),
generalizationPathGroup: SVG!G (
  groupContent <-
    Sequence
{generalizationPath1, generalizationPath2}
),
  generalizationPath1: SVG!Path (
    stroke <- 'black',
    d <- 'M 0 0 L 10 5 L 0 10'
  ),
  generalizationPath2: SVG!Path (
    stroke <- 'black',
    d <- 'M 0.05 0.2 L 0.2
9.95'
    ),
Dependency: SVG!Marker (
  identifier <- 'Dependency',
  refX <- 10.0,
  refY <- 5.0,
  markerWidth <- 11.0,
  markerHeight <- 11.0,
  orient <- 'auto',
  fill <- 'none',
  viewBox <- '0 0 10 10',
  drawing <- dependencyPathGroup
),
dependencyPathGroup: SVG!G (
  groupContent <- dependencyPath
),
  dependencyPath: SVG!Path (
    stroke <- 'black',
    d <- 'M 0 0 L 10 5 L 0 10'
  ),
Aggregation: SVG!Marker (
  identifier <- 'Aggregation',
  refX <- 10.0,

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```


        refY <- 5.0,
        markerWidth <- 11.0,
        markerHeight <- 11.0,
        orient <- '180',
        fill <- 'white',
        viewBox <- '0 0 10 10',
        drawing <- aggregationPathGroup
      ),
      aggregationPathGroup: SVG!G (
        groupContent <- aggregationPath

      ),
      aggregationPath: SVG!Path (
        stroke <- 'black',
        d <- 'M 0 5 L 5 10 L 10 5
L 5 0 L 0 5'
      ),
      Composition: SVG!Marker (
        identifier <- 'Composition',
        refX <- 10.0,
        refY <- 5.0,
        markerWidth <- 11.0,
        markerHeight <- 11.0,
        orient <- '180',
        fill <- 'black',
        viewBox <- '0 0 10 10',
        drawing <- compositionPathGroup
      ),
      compositionPathGroup: SVG!G (
        groupContent <- compositionPath

      ),
      compositionPath: SVG!Path (
        stroke <- 'black',
        d <- 'M 0 5 L 5 10 L 10 5
L 5 0 L 0 5'
      )
    }

-- Classes matching

```


	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```

-- All these rules return a Group containing:
--     + the rectangle drawing the class
--     + the name of the class
--     + the line under the name
-- Attributes and methods, if they exist, are treated
in other rules,
-- and are put in a group each. This in fine gives a
structured class.

-- RULE FullClassBox
-- Matches classes that do contain attributes and
methods.
rule FullClassBox {
  from
    n: UMLDI!GraphNode (
      if n.isOfKind(UMLDI!Class) or n.isOfKind
(UMLDI!Interface) then
        n.hasAttributes() and n.hasMethods() and
n.hasAttSeparator()
      else
        false
      endif
    )
  to
    -- The group element that structures the class
    out: SVG!G (
      name <- 'Class_' +
n.semanticModel.element.name,
      groupContent <-
        Sequence {rect, nameSep, name, attlist,
attSep,
          methodlist},
      root <- thisModule.diagram
    ),
    -- The rectangle that draws the outline of the
class
    -- with its size and its position
    rect: SVG!Rect (


```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```

        fill <- 'white',
        stroke <- 'black',
        position <- rectpos,
        size <- rectdim
      ),
      rectpos: SVG!AbsoluteCoord (
        x <- n.getAbsoluteX(),
        y <- n.getAbsoluteY()
      ),
      rectdim: SVG!Dimension (
        width <- n.size.width,
        height <- n.size.height
      ),
      -- The separator that is drawn below the name of
the class
      -- with its origin and its end
      nameSep: SVG!Line (
        between <- Sequence {origin1, end1},
        stroke <- 'black'
      ),
      origin1: SVG!Point (
        position <- origin1Pos
      ),
      origin1Pos: SVG!AbsoluteCoord (
        x <- n.getAbsoluteX(),
        y <- n.contained->select( e |
          e.semanticModel.typeInfo =
'CompartmentSeparator')->
          first().size.height + n.getAbsoluteY() +
          n.contained->select( e |
            e.isOfType('NameCompartment'))->
            first().contained->select( e |
              e.isOfType('Name'))->first().
size.height
        ),
      end1: SVG!Point (
        position <- end1Pos
      ),
      end1Pos: SVG!AbsoluteCoord (

```


	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005

```

x <- n.getAbsoluteX() + n.size.width,
y <- n.contained->select( e |
  e.semanticModel.typeInfo =
'CompartmentSeparator')->
  first().size.height +
  n.getAbsoluteY() +
  n.contained->select( e |
    e.isOfType('NameCompartment'))->
    first().contained->select( e |
      e.isOfType('Name'))->first().
size.height
  ),

  -- The name of the class, with its weight, style,
and position
  name: SVG!Text (
    content <- n.semanticModel.element.name,
    position <- textPos,
    attribute <- Sequence {fontWeight,
fontStyle},
    fill <- 'black',
    fontSize <- '11px'
  ),
  textPos: SVG!AbsoluteCoord (
    x <- n.contained->select( e |
      e.semanticModel.typeInfo =
'NameCompartment')->
      first().contained->first().getAbsoluteX
    ),
    y <- n.contained->select( e |
      e.semanticModel.typeInfo =
'NameCompartment')->
      first().contained->first().getAbsoluteY
    ) +
    n.contained->select( e |
      e.semanticModel.typeInfo =
'NameCompartment')->
      first().size.height/2
  ),
  fontWeight: SVG!FontWeight (

```


	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005


```

        bold <- true
      ),
      fontStyle: SVG!FontStyle (
        italic <- n.semanticModel.element.isAbstract
      ),

      -- The group that will contain and thus structure
the attributes
      -- of the class
      attlist: SVG!G (
        name <- 'Attributes of Class_' +
n.semanticModel.element.name,
        groupContent <- n.contained->select( e |
          e.isOfType('AttributeCompartment'))->
first().contained->
          select( e | e.isOfType
('DelimitedSection'))->first().contained
        ),

      -- The separator between attributes and methods
      -- with its origin and its end
      attSep: SVG!Line (
        between <- Sequence {origin2, end2},
        stroke <- 'black'
      ),
      origin2: SVG!Point (
        position <- origin2Pos
      ),
      origin2Pos: SVG!AbsoluteCoord (
        x <- n.getAbsoluteX(),
        y <- n.contained->select( e |
          e.semanticModel.typeInfo =
'NameCompartment' )->
          last().size.height + n.getAbsoluteY() +
          n.contained->select( e |
            e.isOfType
('AttributeCompartment' )->
            first().size.height
          ),

```


	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005

```

end2: SVG!Point (
    position <- end2Pos
),
end2Pos: SVG!AbsoluteCoord (
    x <- n.getAbsoluteX() + n.size.width,
    y <- n.contained->select( e |
        e.semanticModel.typeInfo =
'NameCompartment' )->
        last().size.height + n.getAbsoluteY() +
        n.contained->select( e |
            e.isOfType
('AttributeCompartment' )->
            first().size.height
        ),
    -- The group that will contain and thus structure
the methods
    -- of the class
methodlist: SVG!G (
    name <- 'Operations of Class_' +
n.semanticModel.element.name,
    groupContent <- n.contained->select( e |
        e.isOfType('OperationCompartment' )-
>first().contained->
        select( e | e.isOfType
('DelimitedSection' )->first().contained
        )
    )
}

-- RULE EmptyClassBoxNoSeparator
-- Matches classes that do not have attributes,
methods nor separator
-- (the line that separates attributes from
methods)
rule EmptyClassBoxNoSeparator {
    from
        n: UMLDI!GraphNode (
            if n.isOfKind(UMLDI!Class) or n.isOfKind
(UMLDI!Interface) then

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005


```

        not(n.hasAttributes()) and not
(n.hasMethods())
        and not(n.hasAttSeparator())
    else
        false
    endif
)
to
-- The group element that structures the class
out: SVG!G (
    name <- 'Class_' +
n.semanticModel.element.name,
    groupContent <- Sequence {rect, nameSep,
name},
    root <- thisModule.diagram
),

-- The rectangle that draws the outline of the
class
-- with its size and its position
rect: SVG!Rect (
    fill <- 'white',
    stroke <- 'black',
    position <- rectpos,
    size <- rectdim
),
rectpos: SVG!AbsoluteCoord (
    x <- n.getAbsoluteX(),
    y <- n.getAbsoluteY()
),
rectdim: SVG!Dimension (
    width <- n.size.width,
    height <- n.size.height
),

-- The separator that is drawn below the name of
the class
-- with its origin and its end
nameSep: SVG!Line (

```


	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005

```

        between <- Sequence {origin1, end1},
        stroke <- 'black'
    ),
    origin1: SVG!Point (
        position <- origin1Pos
    ),
    origin1Pos: SVG!AbsoluteCoord (
        x <- n.getAbsoluteX(),
        y <- n.contained->select( e |
            e.semanticModel.typeInfo =
'CompartmentSeparator')->
            first().size.height +
            n.getAbsoluteY() +
            n.contained->select( e |
                e.isOfType('NameCompartment'))->
                first().contained->select( e |
                    e.isOfType('Name'))->first().
size.height
    ),
    end1: SVG!Point (
        position <- end1Pos
    ),
    end1Pos: SVG!AbsoluteCoord (
        x <- n.getAbsoluteX() + n.size.width,
        y <- n.contained->select( e |
            e.semanticModel.typeInfo =
'CompartmentSeparator')->
            first().size.height +
            n.getAbsoluteY() +
            n.contained->select( e |
                e.isOfType('NameCompartment'))->
                first().contained->select( e |
                    e.isOfType('Name'))->first().
size.height
    ),

    -- The name of the class, with its weight, style,
and position
    name: SVG!Text (
        content <- n.semanticModel.element.name,

```


	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005

```

        position <- textPos,
        attribute <- Sequence {fontWeight},
        fill <- 'black',
        fontSize <- '11px'
      ),
      textPos: SVG!AbsoluteCoord (
        x <- n.contained->select( e |
          e.semanticModel.typeInfo =
'NameCompartment' )->
          first().contained->first().getAbsoluteX
      ( ),
        y <- n.contained->select( e |
          e.semanticModel.typeInfo =
'NameCompartment' )->
          first().contained->first().getAbsoluteY
      ( ) +
        n.contained->select( e |
          e.semanticModel.typeInfo =
'NameCompartment' )->
          first().size.height/2
      ),
      fontWeight: SVG!FontWeight (
        bold <- true
      )
    }

-- RULE EmptyClassBoxWithSeparator
-- Matches classes that do not have attributes
nor methods but do
-- have a separator.
rule EmptyClassBoxWithSeparator {
  from
    n: UMLDI!GraphNode (
      if n.isOfKind(UMLDI!Class) or n.isOfKind
(UMLDI!Interface) then
        not(n.hasAttributes()) and not
(n.hasMethods())
        and n.hasAttSeparator()
      else

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005


```

        false
      endif
    )
  to
    -- The group element that structures the class
    out: SVG!G (
      name <- 'Class_' +
n.semanticModel.element.name,
      groupContent <- Sequence {rect, nameSep,
name, attSep},
      root <- thisModule.diagram
    ),

    -- The rectangle that draws the outline of the
class
    -- with its size and its position
    rect: SVG!Rect (
      fill <- 'white',
      stroke <- 'black',
      position <- rectpos,
      size <- rectdim
    ),
    rectpos: SVG!AbsoluteCoord (
      x <- n.getAbsoluteX(),
      y <- n.getAbsoluteY()
    ),
    rectdim: SVG!Dimension (
      width <- n.size.width,
      height <- n.size.height
    ),

    -- The separator that is drawn below the name of
the class
    -- with its origin and its end
    nameSep: SVG!Line (
      between <- Sequence {origin1, end1},
      stroke <- 'black'
    ),
    origin1: SVG!Point (

```


	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005

```

        position <- origin1Pos
      ),
      origin1Pos: SVG!AbsoluteCoord (
        x <- n.getAbsoluteX(),
        y <- n.contained->select( e |
          e.semanticModel.typeInfo =
'CompartmentSeparator')->
          first().size.height +
          n.getAbsoluteY() +
          n.contained->select( e |
            e.isOfType('NameCompartment'))->
            first().contained->select( e |
              e.isOfType('Name'))->first().
size.height
        ),
        end1: SVG!Point (
          position <- end1Pos
        ),
        end1Pos: SVG!AbsoluteCoord (
          x <- n.getAbsoluteX() + n.size.width,
          y <- n.contained->select( e |
            e.semanticModel.typeInfo =
'CompartmentSeparator')->
            first().size.height +
            n.getAbsoluteY() +
            n.contained->select( e |
              e.isOfType('NameCompartment'))->
              first().contained->select( e |
                e.isOfType('Name'))->first().
size.height
        ),

        -- The name of the class, with its weight, style,
and position
        name: SVG!Text (
          content <- n.semanticModel.element.name,
          position <- textPos,
          attribute <- Sequence {fontWeight},
          fill <- 'black',
          fontSize <- '11px'

```


	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005

```

    ),
    textPos: SVG!AbsoluteCoord (
      x <- n.contained->select( e |
        e.semanticModel.typeInfo =
'NameCompartment' )->
        first().contained->first().getAbsoluteX
    ),
      y <- n.contained->select( e |
        e.semanticModel.typeInfo =
'NameCompartment' )->
        first().contained->first().getAbsoluteY
    ) +
      n.contained->select( e |
        e.semanticModel.typeInfo =
'NameCompartment' )->
        first().size.height/2
    ),
    fontWeight: SVG!FontWeight (
      bold <- true
    ),

-- The separator between attributes and methods
-- with its origin and its end
attSep: SVG!Line (
  between <- Sequence {origin2, end2},
  stroke <- 'black'
),
origin2: SVG!Point (
  position <- origin2Pos
),
origin2Pos: SVG!AbsoluteCoord (
  x <- n.getAbsoluteX(),
  y <- n.contained->select( e |
    e.semanticModel.typeInfo =
'NameCompartment' )->
    last().size.height +
    n.getAbsoluteY() +
    n.contained->select( e |
      e.isOfType
('AttributeCompartment' )->

```



	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```

        first().size.height
    ),
    end2: SVG!Point (
        position <- end2Pos
    ),
    end2Pos: SVG!AbsoluteCoord (
        x <- n.getAbsoluteX() + n.size.width,
        y <- n.contained->select( e |
            e.semanticModel.typeInfo =
'NameCompartment' )->
            last().size.height +
            n.getAbsoluteY() +
            n.contained->select( e |
                e.isOfType
('AttributeCompartment' ) )->
            first().size.height
        )
    )
}

-- RULE AttributeOnlyClassBoxWithSeparator
-- Matches classes that do have attributes, a
separator
-- but no methods.
rule AttributeOnlyClassBoxWithSeparator {
    from
        n: UMLDI!GraphNode (
            if n.isOfKind(UMLDI!Class) or n.isOfKind
(UMLDI!Interface) then
                n.hasAttributes() and not(n.hasMethods
()) and
                n.hasAttSeparator()
            else
                false
            endif
        )
    to
        -- The group element that structures the class
        out: SVG!G (

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005


```

        name <- 'Class_' +
n.semanticModel.element.name,
        groupContent <- Sequence {rect, nameSep,
name, attlist, attSep},
        root <- thisModule.diagram
    ),

-- The rectangle that draws the outline of the
class
-- with its size and its position
rect: SVG!Rect (
    fill <- 'white',
    stroke <- 'black',
    position <- rectpos,
    size <- rectdim
),
rectpos: SVG!AbsoluteCoord (
    x <- n.getAbsoluteX(),
    y <- n.getAbsoluteY()
),
rectdim: SVG!Dimension (
    width <- n.size.width,
    height <- n.size.height
),

-- The separator that is drawn below the name of
the class
-- with its origin and its end
nameSep: SVG!Line (
    between <- Sequence {origin1, end1},
    stroke <- 'black'
),
origin1: SVG!Point (
    position <- origin1Pos
),
origin1Pos: SVG!AbsoluteCoord (
    x <- n.getAbsoluteX(),
    y <- n.contained->select( e |
        e.semanticModel.typeInfo =
'CompartmentSeparator' )->


```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```

        first().size.height +
        n.getAbsoluteY() +
        n.contained->select( e |
            e.isOfType('NameCompartment'))->
            first().contained->select( e |
                e.isOfType('Name'))->first().
size.height
    ),
    end1: SVG!Point (
        position <- end1Pos
    ),
    end1Pos: SVG!AbsoluteCoord (
        x <- n.getAbsoluteX() + n.size.width,
        y <- n.contained->select( e |
            e.semanticModel.typeInfo =
'CompartmentSeparator')->
            first().size.height +
            n.getAbsoluteY() +
            n.contained->select( e |
                e.isOfType('NameCompartment'))->
                first().contained->select( e |
                    e.isOfType('Name'))->first().
size.height
    ),
    -- The name of the class, with its weight, style,
and position
    name: SVG!Text (
        content <- n.semanticModel.element.name,
        position <- textPos,
        attribute <- Sequence {fontWeight},
        fill <- 'black',
        fontSize <- '11px'
    ),
    textPos: SVG!AbsoluteCoord (
        x <- n.contained->select( e |
            e.semanticModel.typeInfo =
'NameCompartment')->
            first().contained->first().getAbsoluteX
    ),


```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005

```

        y <- n.contained->select( e |
            e.semanticModel.typeInfo =
'NameCompartment' )->
            first().contained->first().getAbsoluteY
() +
            n.contained->select( e |
            e.semanticModel.typeInfo =
'NameCompartment' )->
            first().size.height/2
    ),
    fontWeight: SVG!FontWeight (
        bold <- true
    ),
    -- The group that will contain and thus structure
the attributes
    -- of the class
    attlist: SVG!G (
        name <- 'Attributes of Class_' +
n.semanticModel.element.name,
        groupContent <- n.contained->select( e |
            e.isOfType('AttributeCompartment'))->
            first().contained->select( e |
                e.isOfType('DelimitedSection'))-
>first().contained
    ),
    -- The separator between attributes and methods
    -- with its origin and its end
    attSep: SVG!Line (
        between <- Sequence {origin2, end2},
        stroke <- 'black'
    ),
    origin2: SVG!Point (
        position <- origin2Pos
    ),
    origin2Pos: SVG!AbsoluteCoord (
        x <- n.getAbsoluteX(),
        y <- n.contained->select( e |

```


	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005

```

        e.semanticModel.typeInfo =
'NameCompartment')->
        last().size.height +
        n.getAbsoluteY() +
        n.contained->select( e |
            e.isOfType
('AttributeCompartment'))->
            first().size.height
    ),
    end2: SVG!Point (
        position <- end2Pos
    ),
    end2Pos: SVG!AbsoluteCoord (
        x <- n.getAbsoluteX() + n.size.width,
        y <- n.contained->select( e |
            e.semanticModel.typeInfo =
'NameCompartment')->
            last().size.height +
            n.getAbsoluteY() +
            n.contained->select( e |
                e.isOfType
('AttributeCompartment'))->
                first().size.height
        )
    )
}

-- RULE AttributeOnlyClassBoxNoSeparator
-- Matches classes that do have attributes, no
separator
-- and no methods.
rule AttributeOnlyClassBoxNoSeparator {
    from
        n: UMLDI!GraphNode (
            if n.isOfKind(UMLDI!Class) or n.isOfKind
(UMLDI!Interface) then
                n.hasAttributes() and not(n.hasMethods
()) and
                    not(n.hasAttSeparator())
            else

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005


```

        false
      endif
    )
  to
    -- The group element that structures the class
    out: SVG!G (
      name <- 'Class_' +
n.semanticModel.element.name,
      groupContent <- Sequence {rect, nameSep,
name, attlist},
      root <- thisModule.diagram
    ),

    -- The rectangle that draws the outline of the
class
    -- with its size and its position
    rect: SVG!Rect (
      fill <- 'white',
      stroke <- 'black',
      position <- rectpos,
      size <- rectdim
    ),
    rectpos: SVG!AbsoluteCoord (
      x <- n.getAbsoluteX(),
      y <- n.getAbsoluteY()
    ),
    rectdim: SVG!Dimension (
      width <- n.size.width,
      height <- n.size.height
    ),

    -- The separator that is drawn below the name of
the class
    -- with its origin and its end
    nameSep: SVG!Line (
      between <- Sequence {origin1, end1},
      stroke <- 'black'
    ),
    origin1: SVG!Point (

```


	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005

```

        position <- origin1Pos
      ),
      origin1Pos: SVG!AbsoluteCoord (
        x <- n.getAbsoluteX(),
        y <- n.contained->select( e |
          e.semanticModel.typeInfo =
'CompartmentSeparator')->
          first().size.height +
          n.getAbsoluteY() +
          n.contained->select( e |
            e.isOfType('NameCompartment'))->
            first().contained->select( e |
              e.isOfType('Name'))->first().
size.height
        ),
        end1: SVG!Point (
          position <- end1Pos
        ),
        end1Pos: SVG!AbsoluteCoord (
          x <- n.getAbsoluteX() + n.size.width,
          y <- n.contained->select( e |
            e.semanticModel.typeInfo =
'CompartmentSeparator')->
            first().size.height +
            n.getAbsoluteY() +
            n.contained->select( e |
              e.isOfType('NameCompartment'))->
              first().contained->select( e |
                e.isOfType('Name'))->first().
size.height
        ),

        -- The name of the class, with its weight, style,
and position
        name: SVG!Text (
          content <- n.semanticModel.element.name,
          position <- textPos,
          attribute <- Sequence {fontWeight},
          fill <- 'black',
          fontSize <- '11px'

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005


```

    ),
    textPos: SVG!AbsoluteCoord (
      x <- n.contained->select( e |
        e.semanticModel.typeInfo =
'NameCompartment' )->
        first().contained->first().getAbsoluteX
    ),
      y <- n.contained->select( e |
        e.semanticModel.typeInfo =
'NameCompartment' )->
        first().contained->first().getAbsoluteY
    ) +
      n.contained->select( e |
        e.semanticModel.typeInfo =
'NameCompartment' )->
        first().size.height/2
    ),
    fontWeight: SVG!FontWeight (
      bold <- true
    ),

    -- The group that will contain and thus structure
the attributes
    -- of the class
    attlist: SVG!G (
      name <- 'Attributes of Class_' +
n.semanticModel.element.name,
      groupContent <- n.contained->select( e |
        e.isOfType('AttributeCompartment')->
        first().contained->select( e |
          e.isOfType('DelimitedSection')-
>first().contained
        )
      )
    }

    -- RULE MethodOnlyClassBoxWithSeparator
    -- Matches classes that do have methods, a
separator
    -- and no attributes.

```



	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005

```

rule MethodOnlyClassBoxWithSeparator {
  from
    n: UMLDI!GraphNode (
      if n.isOfKind(UMLDI!Class) or n.isOfKind
(UMLDI!Interface) then
        not(n.hasAttributes()) and n.hasMethods
()
        and n.hasAttSeparator()
      else
        false
      endif
    )
  to
    -- The group element that structures the class
    out: SVG!G (
      name <- 'Class_' +
n.semanticModel.element.name,
      groupContent <- Sequence {rect, nameSep,
name, attSep, methodlist},
      root <- thisModule.diagram
    ),

    -- The rectangle that draws the outline of the
class
    -- with its size and its position
    rect: SVG!Rect (
      fill <- 'white',
      stroke <- 'black',
      position <- rectpos,
      size <- rectdim
    ),
    rectpos: SVG!AbsoluteCoord (
      x <- n.getAbsoluteX(),
      y <- n.getAbsoluteY()
    ),
    rectdim: SVG!Dimension (
      width <- n.size.width,
      height <- n.size.height
    ),
  )
}


```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005

```

-- The separator that is drawn below the name of
the class
-- with its origin and its end
nameSep: SVG!Line (
  between <- Sequence {origin1, end1},
  stroke <- 'black'
),
origin1: SVG!Point (
  position <- origin1Pos
),
origin1Pos: SVG!AbsoluteCoord (
  x <- n.getAbsoluteX(),
  y <- n.contained->select( e |
    e.semanticModel.typeInfo =
'CompartmentSeparator')->
    first().size.height +
    n.getAbsoluteY() +
    n.contained->select( e |
      e.isOfType('NameCompartment'))->
      first().contained->select( e |
        e.isOfType('Name'))->first().
size.height
  ),
end1: SVG!Point (
  position <- end1Pos
),
end1Pos: SVG!AbsoluteCoord (
  x <- n.getAbsoluteX() + n.size.width,
  y <- n.contained->select( e |
    e.semanticModel.typeInfo =
'CompartmentSeparator')->
    first().size.height +
    n.getAbsoluteY() +
    n.contained->select( e |
      e.isOfType('NameCompartment'))->
      first().contained->select( e |
        e.isOfType('Name'))->first().
size.height
  ),

```


	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```

-- The name of the class, with its weight, style,
and position
name: SVG!Text (
  content <- n.semanticModel.element.name,
  position <- textPos,
  attribute <- Sequence {fontWeight},
  fill <- 'black',
  fontSize <- '11px'
),
textPos: SVG!AbsoluteCoord (
  x <- n.contained->select( e |
    e.semanticModel.typeInfo =
'NameCompartment' )->
    first().contained->first().getAbsoluteX
(),
  y <- n.contained->select( e |
    e.semanticModel.typeInfo =
'NameCompartment' )->
    first().contained->first().getAbsoluteY
() +
    n.contained->select( e |
    e.semanticModel.typeInfo =
'NameCompartment' )->
    first().size.height/2
),
fontWeight: SVG!FontWeight (
  bold <- true
),

-- The separator between attributes and methods
-- with its origin and its end
attSep: SVG!Line (
  between <- Sequence {origin2, end2},
  stroke <- 'black'
),
origin2: SVG!Point (
  position <- origin2Pos
),
origin2Pos: SVG!AbsoluteCoord (

```


	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005

```

x <- n.getAbsoluteX(),
y <- n.contained->select( e |
    e.semanticModel.typeInfo =
'NameCompartment' )->
    last().size.height +
    n.getAbsoluteY() +
    n.contained->select( e |
        e.isOfType
('AttributeCompartment' ) )->
        first().size.height
    ),
end2: SVG!Point (
    position <- end2Pos
),
end2Pos: SVG!AbsoluteCoord (
    x <- n.getAbsoluteX() + n.size.width,
    y <- n.contained->select( e |
        e.semanticModel.typeInfo =
'NameCompartment' )->
        last().size.height + n.getAbsoluteY()
    ),
    -- The group that will contain and thus structure
the methods
    -- of the class
methodlist: SVG!G (
    name <- 'Operations of Class_' +
n.semanticModel.element.name,
    groupContent <- n.contained->select( e |
        e.isOfType('OperationCompartment' ) )->
        first().contained->select( e |
            e.isOfType('DelimitedSection' ) )-
>first().contained
    )
}

    -- RULE MethodOnlyClassBoxNoSeparator
    -- Matches classes that do have methods, no
separator
    -- and no attributes.

```


	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```

rule MethodOnlyClassBoxNoSeparator {
  from
    n: UMLDI!GraphNode (
      if n.isOfKind(UMLDI!Class) or n.isOfKind
(UMLDI!Interface) then
        not(n.hasAttributes()) and n.hasMethods
() and
        not(n.hasAttSeparator())
      else
        false
      endif
    )
  to
    -- The group element that structures the class
    out: SVG!G (
      name <- 'Class_' +
n.semanticModel.element.name,
      groupContent <- Sequence {rect, nameSep,
name, methodlist},
      root <- thisModule.diagram
    ),

    -- The rectangle that draws the outline of the
class
    -- with its size and its position
    rect: SVG!Rect (
      fill <- 'white',
      stroke <- 'black',
      position <- rectpos,
      size <- rectdim
    ),
    rectpos: SVG!AbsoluteCoord (
      x <- n.getAbsoluteX(),
      y <- n.getAbsoluteY()
    ),
    rectdim: SVG!Dimension (
      width <- n.size.width,
      height <- n.size.height
    ),


```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005

```

-- The separator that is drawn below the name of
the class
-- with its origin and its end
nameSep: SVG!Line (
  between <- Sequence {origin1, end1},
  stroke <- 'black'
),
origin1: SVG!Point (
  position <- origin1Pos
),
origin1Pos: SVG!AbsoluteCoord (
  x <- n.getAbsoluteX(),
  y <- n.contained->select( e |
    e.semanticModel.typeInfo =
'CompartmentSeparator')->
    first().size.height +
    n.getAbsoluteY() +
    n.contained->select( e |
      e.isOfType('NameCompartment'))->
      first().contained->select( e |
        e.isOfType('Name'))->first().
size.height
  ),
end1: SVG!Point (
  position <- end1Pos
),
end1Pos: SVG!AbsoluteCoord (
  x <- n.getAbsoluteX() + n.size.width,
  y <- n.contained->select( e |
    e.semanticModel.typeInfo =
'CompartmentSeparator')->
    first().size.height +
    n.getAbsoluteY() +
    n.contained->select( e |
      e.isOfType('NameCompartment'))->
      first().contained->select( e |
        e.isOfType('Name'))->first().
size.height
  ),

```


	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005

```
-- The name of the class, with its weight, style,
and position
```

```
name: SVG!Text (
  content <- n.semanticModel.element.name,
  position <- textPos,
  attribute <- Sequence {fontWeight},
  fill <- 'black',
  fontSize <- '11px'
),
textPos: SVG!AbsoluteCoord (
  x <- n.contained->select( e |
    e.semanticModel.typeInfo =
'NameCompartment' )->
    first().contained->first().getAbsoluteX
(),
  y <- n.contained->select( e |
    e.semanticModel.typeInfo =
'NameCompartment' )->
    first().contained->first().getAbsoluteY
() +
    n.contained->select( e |
    e.semanticModel.typeInfo =
'NameCompartment' )->
    first().size.height/2
),
fontWeight: SVG!FontWeight (
  bold <- true
),
```

```
-- The group that will contain and thus structure
the methods
```

```
-- of the class
methodlist: SVG!G (
  name <- 'Operations of Class_' +
n.semanticModel.element.name,
  groupContent <- n.contained->select( e |
    e.isOfType('OperationCompartment') )->
    first().contained->select( e |
```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```


                                e.isOfType('DelimitedSection'))-
>first().contained
    )
}

-- End classes matching

-- Attributes and Methods

    -- RULE Attributes
    -- Matches attributes and returns the
corresponding text with:
    --     + the visibility
    --     + the name of the attribute
    --     + its type
    --     + its multiplicity
    --     + its initial value
rule Attributes {
  from
    a: UMLDI!GraphNode (
      a.isOfKind(UMLDI!Attribute)
    )
  to
    out: SVG!Text (
      content <- a.getVisibility() +
        a.semanticModel.element.name + ':' +
        a.semanticModel.element.type.name +
        a.mult() +
        a.getInitialValue(),
      fill <- 'black',
      position <- atPos,
      fontSize <- '11px'
    ),
    atPos: SVG!AbsoluteCoord (
      x <- a.getAbsoluteX(),
      y <- a.getAbsoluteY() + a.contained->select
(e |
      e.isOfType('Name'))->first().
size.height/2

```



	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```

    )
  }

  -- RULE Methods
  -- Matches attributes and returns the
  corresponding text with:
  --     + the visibility
  --     + the name of the method
  --     + its parameters with their types
  --     + its return type
rule Methods {
  from
    m: UMLDI!GraphNode (
      m.isOfKind(UMLDI!Operation)
    )
  to
    out: SVG!Text (
      content <- m.getVisibility() +
        m.semanticModel.element.name +
        '(' + m.getParameters() + ')' + ':' +
        m.getReturnParameter(),
      fill <- 'black',
      position <- OpPos,
      fontSize <- '11px',
      attribute <- Sequence {fontStyle}
    ),
    OpPos: SVG!AbsoluteCoord (
      x <- m.getAbsoluteX(),
      y <- m.getAbsoluteY() + m.contained->select
      ( e |
        e.isOfType('Name'))->first().
      size.height/2
    ),
    fontStyle: SVG!FontStyle (
      italic <- m.semanticModel.element.isAbstract
    )
  }

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```


-- End Attributes and Methods

-- Association, dependencies, etc

    -- RULE GraphEdge2Association
    -- Creates an association
rule GraphEdge2Association {
  from
    ge: UMLDI!GraphEdge (
      ge.isOfKind(UMLDI!Association)
    )
  to
    -- The group that will contain:
    --     + the line
    --     + the name
    --     + the multiplicities
    --     + the roles
    out: SVG!G (
      name <- ge.semanticModel.element.name,
      groupContent <- Sequence {polyline, name,
leftMultiplicity,
      rightMultiplicity, leftRole, rightRole},
      root <- thisModule.diagram
    ),

    -- The line, with its position, markers, and
waypoints
    polyline: SVG!Polyline (
      waypoints <- Sequence {wps}->flatten(),
      stroke <- 'black',
      fill <- 'white',
      markerEnd <- ge.getMarker('end'),
      markerStart <- ge.getMarker('start')
    ),
    wps: distinct SVG!Point foreach (Point in
ge.waypoints) (
      position <- PPos
    ),

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005


```

PPos: distinct SVG!AbsoluteCoord foreach(Point in
ge.waypoints) (
  x <- Point.base.x - thisModule.offset.x,
  y <- Point.base.y - thisModule.offset.y
),

-- The name of the association, with its position
name: SVG!Text (
  content <- ge.getName(),
  fill <- 'black',
  position <- namePos,
  fontSize <- '10px'
),
namePos: SVG!AbsoluteCoord (
  x <- if ge.isNamed() then
    ge.contained->select( e |
    e.isOfType('DirectedName'))->first
().getAbsoluteX()
    else 0.0 endif,
  y <- if ge.isNamed() then
    ge.contained->select( e |
    e.isOfType('DirectedName'))->first
().getAbsoluteY()
    else 0.0 endif
),

-- The first multiplicity
leftMultiplicity: SVG!Text (
  content <- if ge.hasLeftMultiplicity() then
ge.semanticModel.element.connection->
  first().
multiplicity.getMultiplicity()
    else '' endif,
  fill <- 'black',
  position <- leftMultPos,
  fontSize <- '10px'
),
leftMultPos: SVG!AbsoluteCoord (


```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```

x <- if ge.hasLeftMultiplicity() then
  ge.contained->select( e |
    e.isOfKind
(UMLDI!AssociationEnd))->
    first().contained->select( e |
      e.isOfType
('Multiplicity'))->first().getAbsoluteX()
    else 0.0 endif,
  y <- if ge.hasLeftMultiplicity() then
    ge.contained->select( e |
      e.isOfKind
(UMLDI!AssociationEnd))->
        first().contained->select( e |
          e.isOfType
('Multiplicity'))->
            first().getAbsoluteY() +
              ge.contained->select( e |
                e.isOfKind
(UMLDI!AssociationEnd))->
                  first().contained-
>select( e |
          e.isOfType
('Multiplicity'))->
            first().
size.height
    else 0.0 endif
  ),
-- The second multiplicity
rightMultiplicity: SVG!Text (
  content <- if ge.hasRightMultiplicity() then
    ge.semanticModel.element.connection->
      last().multiplicity.getMultiplicity
()
    else '' endif,
  fill <- 'black',
  position <- rightMultPos,
  fontSize <- '10px'
),

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005


```

rightMultPos: SVG!AbsoluteCoord (
  x <- if ge.hasRightMultiplicity() then
    ge.contained->select( e |
      e.isOfKind(UMLDI!AssociationEnd))->
      last().contained->select( e |
        e.isOfType('Multiplicity'))-
>last().getAbsoluteX()
    else 0.0 endif,
  y <- if ge.hasRightMultiplicity() then
    ge.contained->select( e |
      e.isOfKind(UMLDI!AssociationEnd))->
      last().contained->select( e |
        e.isOfType('Multiplicity'))-
>last().getAbsoluteY() +
    ge.contained->select( e |
      e.isOfKind
(UMLDI!AssociationEnd))->
      last().contained->select
( e |
      e.isOfType
('Multiplicity'))->last().size.height
    else 0.0 endif
),

-- The first role
leftRole: SVG!Text (
  content <- if ge.hasLeftRole() then

ge.semanticModel.element.connection->first().name
    else '' endif,
  fill <- 'black',
  position <- leftRolePos,
  fontSize <- '10px'
),
leftRolePos: SVG!AbsoluteCoord (
  x <- if ge.hasLeftRole() then
    ge.contained->select( e |
      e.isOfKind(UMLDI!AssociationEnd))->
      first().contained->select( e |


```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```

                                e.isOfType('Name'))->first().
getAbsoluteX()
                                else 0.0 endif,
y <- if ge.hasLeftRole() then
    ge.contained->select( e |
        e.isOfKind(UMLDI!AssociationEnd))->
        first().contained->select( e |
            e.isOfType('Name'))->first().
getAbsoluteY() +
                                ge.contained->select( e |
                                    e.isOfKind
(UMLDI!AssociationEnd))->
                                first().contained->select(
e |
                                    e.isOfType('Name'))->
>first().size.height
                                else 0.0 endif
),
-- The second role
rightRole: SVG!Text (
    content <- if ge.hasRightRole() then
ge.semanticModel.element.connection->last().name
                                else '' endif,
    fill <- 'black',
    position <- rightRolePos,
    fontSize <- '10px'
),
rightRolePos: SVG!AbsoluteCoord (
    x <- if ge.hasRightRole() then
        ge.contained->select( e |
            e.isOfKind(UMLDI!AssociationEnd))->
            last().contained->select( e |
                e.isOfType('Name'))->first().
getAbsoluteX()
                                else 0.0 endif,
y <- if ge.hasRightRole() then
    ge.contained->select( e |

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```

e.isOfKind(UMLDI!AssociationEnd))->
last().contained->select( e |
    e.isOfType('Name'))->first().
getAbsoluteY() +
    ge.contained->select( e |
        e.isOfKind
(UMLDI!AssociationEnd))->
        last().contained->select
( e |
    e.isOfType('Name'))-
>first().size.height
    else 0.0 endif
)
}

```

```

-- RULE GraphEdge2Dependency
-- Creates a group containing:
--     + the line that draws the dependency,
with its marker
--     + the name of the dependency

```

```


rule GraphEdge2Dependency {
  from
    ge: UMLDI!GraphEdge (
      ge.isOfKind(UMLDI!Dependency)
    )
  to
    -- The group element that will contain:
    --     + the line
    --     + the name
    out: SVG!G (
      name <- ge.semanticModel.element.name,
      root <- thisModule.diagram,
      groupContent <- Sequence {polyline, name}
    ),

```

```

-- The line, with its marker, style, and
waypoints
polyline: SVG!Polyline(
  waypoints <- Sequence {wps}->flatten(),

```


	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005

```

        stroke <- 'black',
        fill <- 'white',
        strokeDashArray <- '5,5',
        markerEnd <- 'url(#Dependency)',
        markerStart <- 'none'
      ),
      wps: distinct SVG!Point foreach (Point in
ge.waypoints) (
        position <- PPos
      ),
      PPos: distinct SVG!AbsoluteCoord foreach(Point in
ge.waypoints) (
        x <- Point.base.x - thisModule.offset.x,
        y <- Point.base.y - thisModule.offset.y
      ),

-- The name of the Dependency
name: SVG!Text (
  content <- ge.getName(),
  fill <- 'black',
  position <- namePos,
  fontSize <- '10px'
),
namePos: SVG!AbsoluteCoord (
  x <- if ge.isNamed() then
    ge.contained->select( e |
      e.isOfType('DirectedName'))->first
    ).contained->select( e |
      e.isOfType('Name'))->first().
    getAbsoluteX()
    else 0.0 endif,
  y <- if ge.isNamed() then
    ge.contained->select( e |
      e.isOfType('DirectedName'))->first
    ).contained->select( e |
      e.isOfType('Name'))->first().
    getAbsoluteY()
    else 0.0 endif
  )
)

```



	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```

}

-- RULE GraphEdge2Generalization
-- Creates a group containing:
--     + the line that draws the
generalization,
--     + the name of the generalization
--     with its marker
rule GraphEdge2Generalization {
  from
    ge: UMLDI!GraphEdge (
      ge.isOfKind(UMLDI!Generalization)
    )
  to
    -- The group element that will contain:
    --     + the line
    --     + the name
    out: SVG!G (
      name <- ge.semanticModel.element.name,
      root <- thisModule.diagram,
      groupContent <- Sequence {polyline, name}
    ),
    -- The line, with its marker and its position
    polyline: SVG!Polyline(
      waypoints <- Sequence {wps}->flatten(),
      stroke <- 'black',
      fill <- 'white',
      markerEnd <- 'url(#Generalization)',
      markerStart <- 'none'
    ),
    wps: distinct SVG!Point foreach (Point in
ge.waypoints) (
      position <- PPos
    ),
    PPos: distinct SVG!AbsoluteCoord foreach(Point in
ge.waypoints) (
      x <- Point.base.x - thisModule.offset.x,
      y <- Point.base.y - thisModule.offset.y

```


	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```

    ),
    -- The name, with its position
    name: SVG!Text (
      content <- ge.getName(),
      fill <- 'black',
      position <- namePos,
      fontSize <- '10px'
    ),
    namePos: SVG!AbsoluteCoord (
      x <- if ge.isNamed() then
        ge.contained->select( e |
          e.isOfType('DirectedName'))->first
        ().contained->select( e |
          e.isOfType('Name'))->first().
        getAbsoluteX()
        else 0.0 endif,
      y <- if ge.isNamed() then
        ge.contained->select( e |
          e.isOfType('DirectedName'))->first
        ().contained->select( e |
          e.isOfType('Name'))->first().
        getAbsoluteY()
        else 0.0 endif
    )
  }

  -- RULE GraphEdge2Abstraction
  -- Creates a group containing:
  --       + the line that draws the abstraction,
  --       with its marker
  --       + the name of the abstraction
rule GraphEdge2Abstraction {
  from
    ge: UMLDI!GraphEdge (
      ge.isOfKind(UMLDI!Abstraction)
    )
  to
    -- The group element that will contain:

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005


```

--          + the line
--          + the name
out: SVG!G (
    name <- ge.semanticModel.element.name,
    root <- thisModule.diagram,
    groupContent <- Sequence {polyline, name}
),

-- The line, with its marker, its style and its
position
polyline: SVG!Polyline(
    waypoints <- Sequence {wps}->flatten(),
    stroke <- 'black',
    fill <- 'white',
    strokeDashArray <- '5,5',
    markerEnd <- 'url(#Generalization)',
    markerStart <- 'none'
),
    wps: distinct SVG!Point foreach (Point in
ge.waypoints) (
    position <- PPos
),
    PPos: distinct SVG!AbsoluteCoord foreach(Point in
ge.waypoints) (
    x <- Point.base.x - thisModule.offset.x,
    y <- Point.base.y - thisModule.offset.y
),

-- The name, with its position
name: SVG!Text (
    content <- if ge.isStereotypeNamed() then
        '&#171; ' +
        ge.contained->select( e |
        e .isOfType
('StereotypeCompartment'))->
            first().contained->select( e |
            e.isOfKind
(UMLDI!Stereotype))->

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005


```

                                first().
semanticModel.element.name +
                                ' &#187;'
                                else '' endif,
                                fill <- 'black',
                                position <- namePos,
                                fontSize <- '10px'
                                ),
                                namePos: SVG!AbsoluteCoord (
                                x <- if ge.isStereotypeNamed() then
                                ge.contained->select( e |
                                e .isOfType
('StereotypeCompartment'))->
                                first().contained->select( e |
                                e.isOfKind(UMLDI!Stereotype))-
>first().getAbsoluteX()
                                else 0.0 endif,
                                y <- if ge.isStereotypeNamed() then
                                ge.contained->select( e |
                                e .isOfType
('StereotypeCompartment'))->
                                first().contained->select( e |
                                e.isOfKind(UMLDI!Stereotype))-
>first().getAbsoluteY()
                                else 0.0 endif
                                )
}

-- End Association, dependencies, etc

-- End rules

```


	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005

1. Output svg file (the model is serialized into an actual SVG file)

```

<svg xmlns="http://www.w3.org/2000/svg" x="20.0" y="67.0">
  <defs>
    <marker refX="10.0" refY="5.0" id="Association"
markerWidth="11.0" markerHeight="11.0" orient="auto"
fill="none" viewBox="0 0 10 10">
      <g>
        <path d="M 0 0 L 10 5 L 0 10" stroke="black"/>
      </g>
    </marker>
    <marker refX="10.0" refY="5.0" id="Generalization"
markerWidth="11.0" markerHeight="11.0" orient="auto"
fill="white" viewBox="0 0 10 10">
      <g>
        <path d="M 0 0 L 10 5 L 0 10" stroke="black"/>
        <path d="M 0.05 0.2 L 0.2 9.95" stroke="black"/>
      </g>
    </marker>
    <marker refX="10.0" refY="5.0" id="Dependency"
markerWidth="11.0" markerHeight="11.0" orient="auto"
fill="none" viewBox="0 0 10 10">
      <g>
        <path d="M 0 0 L 10 5 L 0 10" stroke="black"/>
      </g>
    </marker>
    <marker refX="10.0" refY="5.0" id="Aggregation"
markerWidth="11.0" markerHeight="11.0" orient="180"
fill="white" viewBox="0 0 10 10">
      <g>
        <path d="M 0 5 L 5 10 L 10 5 L 5 0 L 0 5"
stroke="black"/>
      </g>
    </marker>
    <marker refX="10.0" refY="5.0" id="Composition"
markerWidth="11.0" markerHeight="11.0" orient="180"
fill="black" viewBox="0 0 10 10">
      <g>


```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Palès
	UMLDI to SVG	Date 07/04/2005

```

      <path d="M 0 5 L 5 10 L 10 5 L 5 0 L 0 5"
stroke="black"/>
    </g>
  </marker>
</defs>
<g>
  <rect x="490.0" y="137.0" width="232.3257"
height="70.0" fill="white" stroke="black"/>
  <line x1="490.0" y1="152.0" x2="722.3257" y2="152.0"
stroke="black"/>
  <text x="587.3748" y="149.0" fill="black" font-
size="11px" font-weight="bold">Class2</text>
  <g>
    <text x="495.0" y="168.0" fill="black" font-
size="11px">~attribute2:int1..*= 42</text>
  </g>
  <line x1="490.0" y1="179.0" x2="722.3257" y2="179.0"
stroke="black"/>
  <g>
    <text x="495.0" y="193.0" fill="black" font-
size="11px">#operation_2(p2:short[],p3:int):void</text>
  </g>
</g>
<g>
  <rect x="150.0" y="137.0" width="150.4004"
height="70.0" fill="white" stroke="black"/>
  <line x1="150.0" y1="152.0" x2="300.4004" y2="152.0"
stroke="black"/>
  <text x="206.4121" y="149.0" fill="black" font-
size="11px" font-weight="bold">Class1</text>
  <g>
    <text x="155.0" y="168.0" fill="black" font-
size="11px">-attributel:int</text>
  </g>
  <line x1="150.0" y1="179.0" x2="300.4004" y2="179.0"
stroke="black"/>
  <g>
    <text x="155.0" y="193.0" fill="black" font-
size="11px">+operation1(p1:int):float[]</text>
  </g>


```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```

</g>
<g>
  <rect x="150.0" y="307.0" width="100.0" height="70.0"
fill="white" stroke="black"/>
  <line x1="150.0" y1="322.0" x2="250.0" y2="322.0"
stroke="black"/>
  <text x="181.2119" y="319.0" fill="black" font-
size="11px" font-weight="bold">Class3</text>
  <g/>
  <line x1="150.0" y1="349.0" x2="250.0" y2="349.0"
stroke="black"/>
  <g/>
</g>
<g>
  <rect x="540.0" y="17.0" width="120.4619" height="60.0"
fill="white" stroke="black"/>
  <line x1="540.0" y1="32.0" x2="660.4619" y2="32.0"
stroke="black"/>
  <text x="571.552" y="29.0" fill="black" font-
size="11px" font-weight="bold">Interface1</text>
  <g>
    <text x="545.0" y="47.0" fill="black" font-
size="11px" font-style="italic">+operation3():void</text>
  </g>
</g>
<g>
  <polyline points="250.0,337.0 590.0,337.0 590.0,207.0 "
stroke="black" fill="white" marker-end="none" marker-
start="url(#Composition)"/>
  <text x="0.0" y="0.0" fill="black" font-
size="10px"></text>
  <text x="0.0" y="0.0" fill="black" font-
size="10px"></text>
  <text x="0.0" y="0.0" fill="black" font-
size="10px"></text>
  <text x="0.0" y="0.0" fill="black" font-
size="10px"></text>
  <text x="0.0" y="0.0" fill="black" font-
size="10px"></text>
</g>


```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005


```

<g>
  <polyline points="300.4004,172.0 490.0,172.0 "
stroke="black" fill="white" marker-end="url(#Association)"
marker-start="none"/>
  <text x="361.0589" y="182.0" fill="black" font-
size="10px">Association1</text>
  <text x="313.3908" y="164.5" fill="black" font-
size="10px">1..*</text>
  <text x="456.1375" y="193.5" fill="black" font-
size="10px">0..1</text>
  <text x="313.3908" y="190.7495" fill="black" font-
size="10px">theEnd</text>
  <text x="410.1126" y="169.125" fill="black" font-
size="10px">theOtherEnd</text>
</g>
<g>
  <polyline points="200.0,137.0 200.0,47.0 540.0,47.0 "
stroke="black" fill="white" stroke-dasharray="5,5" marker-
end="url(#Dependency)"/>
  <text x="0.0" y="0.0" fill="black" font-
size="10px"></text>
</g>
<g>
  <polyline points="590.0,137.0 590.0,77.0 "
stroke="black" fill="white" stroke-dasharray="5,5" marker-
end="url(#Dependency)"/>
  <text x="0.0" y="0.0" fill="black" font-
size="10px"></text>
</g>
<g>
  <polyline points="200.0,307.0 200.0,207.0 "
stroke="black" fill="white" marker-end="url
(#Generalization)" marker-start="none"/>
  <text x="0.0" y="0.0" fill="black" font-
size="10px"></text>
</g>
<g>
  <polyline points="590.0,137.0 590.0,77.0 "
stroke="black" fill="white" stroke-dasharray="5,5" marker-
end="url(#Generalization)"/>

```


	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```
<text x="529.0142" y="100.0" fill="black" font-size="10px">⌈; realize ⌋;</text>  
</g>  
</svg>
```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

II. SVG metamodel in KM3 format

```

package SVG {


    abstract class Element {
        reference owner[*] : SvgFile oppositeOf elements;
        reference target[*] : Use oppositeOf use;
        reference "attribute"[*] : Attribute oppositeOf
attOwner;
        reference position[0-1] container : Coordinates;
        reference size[0-1] container : Dimension;
        reference root[0-1] : Svg oppositeOf children;
        attribute fill[0-1] : String;
        attribute viewBox[0-1] : String;
        reference group[0-1] : GroupingElement oppositeOf
groupContent;
        attribute identifier[0-1] : String;
        reference drawsMarker[0-1] : Marker oppositeOf
drawing;
    }

    -- Structural Elements
    abstract class StructuralElement extends Element {
    }

    class Image extends StructuralElement {
        reference refereee[*] : ReferencedFile oppositeOf
referer;
    }

    class Svg extends StructuralElement {
        reference owner[*] : SvgFile oppositeOf tag;
        reference children[*] ordered container : Element
oppositeOf root;
        attribute namespace[0-1] : String;
        attribute version[0-1] : String;
        attribute baseProfile[0-1] : String;
    }

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```

abstract class GroupingElement extends
StructuralElement {
    reference groupContent[*] ordered container :
Element oppositeOf group;
}

class G extends GroupingElement {
    attribute name[0-1] : String;
}

class Defs extends GroupingElement {
}

class Symbol extends GroupingElement {
}

class Use extends StructuralElement {
    reference use[*] : Element oppositeOf target;
}

abstract class GraphicalElement extends Element {
    attribute stroke[0-1] : String;
}


abstract class Shape extends GraphicalElement {
}

abstract class TextElement extends GraphicalElement {
    attribute rotate[0-1] : Double;
    attribute textLength[0-1] : String;
    attribute fontSize[0-1] : String;
}

class Rect extends Shape {
    attribute rx[0-1] : Double;
    attribute ry[0-1] : Double;
}

class Circle extends Shape {

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```

}

class Ellipse extends Shape {
}

class Line extends Shape {
    reference between[2-2] : Point;
    attribute markerEnd[0-1] : String;
    attribute markerStart[0-1] : String;
}

class Polyline extends Shape {
    reference waypoints[*] ordered container : Point;
    attribute strokeDashArray[0-1] : String;
    attribute markerEnd[0-1] : String;
    attribute markerStart[0-1] : String;
}


class Polygon extends Shape {
    reference waypoints[*] ordered : Point;
    attribute markerEnd[0-1] : String;
    attribute markerStart[0-1] : String;
}

class Path extends Shape {
    attribute pathLength[0-1] : Double;
    attribute d : String;
    attribute markerEnd[0-1] : String;
    attribute markerStart[0-1] : String;
}

class Point extends Shape {
}

class Marker extends Shape {
    attribute markerUnits[0-1] : String;
    attribute refX[0-1] : Double;
    attribute refY[0-1] : Double;
    attribute markerWidth[0-1] : Double;
    attribute markerHeight[0-1] : Double;
}

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```

        attribute orient[0-1] : String;
        reference drawing[*] container : Element
oppositeOf drawsMarker;
    }

class Text extends TextElement {
    attribute lengthAdjust[0-1] : String;
    attribute content : String;
}

class Tspan extends TextElement {
    attribute content[0-1] : String;
}

class Tref extends TextElement {
    reference xlinkHref : TextElement;
}

abstract class Attribute {
    reference attOwner[*] : Element oppositeOf
"attribute";
}


abstract class Transform extends Attribute {
}

class Scale extends Transform {
    attribute sx : Double;
    attribute sy : Double;
}

class Translate extends Transform {
    attribute tx : Double;
    attribute ty : Double;
}

class Rotate extends Transform {
    attribute angle : Double;
    attribute cx : Double;
    attribute cy : Double;
}

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

```

}

class VBox extends Attribute {
    attribute minX : Double;
    attribute minY : Double;
}

class Visibility extends Attribute {
    attribute visible : Boolean;
}

class FontWeight extends Attribute {
    attribute bold : Boolean;
}

class FontStyle extends Attribute {
    attribute italic : Boolean;
}


-- Coordinates and Dimension
    -- For width, height. length is the longer radius of
an ellipse.
    class Dimension {
        attribute width : Double;
        attribute height : Double;
    }

    -- Coordinates are either relative or absolute
abstract class Coordinates {
    attribute x : Double;
    attribute y : Double;
}

class RelativeCoord extends Coordinates {
}

class AbsoluteCoord extends Coordinates {
}
-- End Coordinates and Dimension

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005


```

-- Files
  -- A file that is referenced by some tag in the
document
  abstract class ReferencedFile {
    reference referer[*] : Image oppositeOf referee;
    attribute name : String;
  }

  -- A svg file that is referenced via a use tag calling
its svg tag
  class SvgFile extends ReferencedFile {
    reference tag : Svg oppositeOf owner;
    reference elements[*] : Element oppositeOf owner;
  }
-- End Files
}

package PrimitiveTypes {
  datatype Boolean;
  datatype Integer;
  datatype String;
  datatype Double;
}

```

	ATL TRANSFORMATION EXAMPLE	Contributor Jean Paliès
	UMLDI to SVG	Date 07/04/2005

References

- [1] UML Diagram Interchange 2.0 Adopted specification, OMG, <http://www.omg.org/cgi-bin/doc?ptc/2003-09-01>
- [2] Scalable Vector Graphics 1.1, World Wide Web Consortium, <http://www.w3.org/TR/SVG11/>