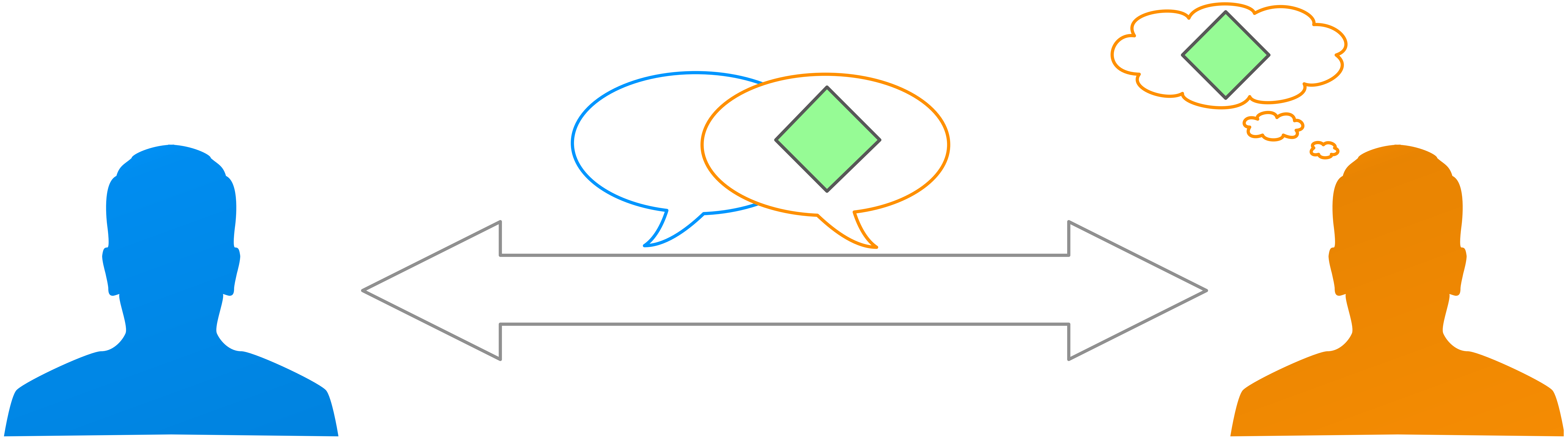
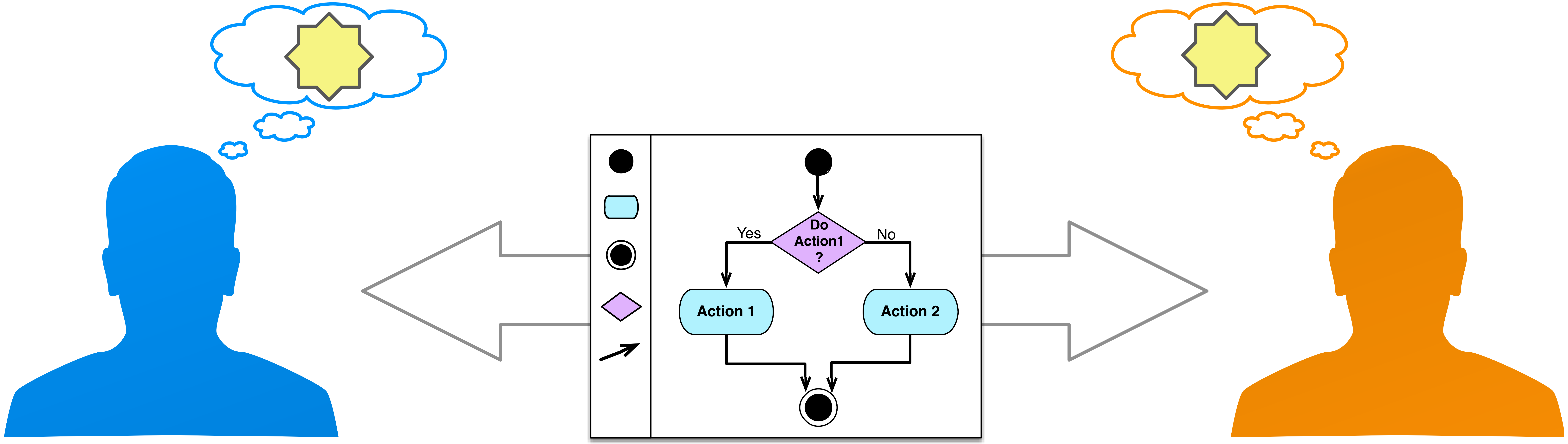
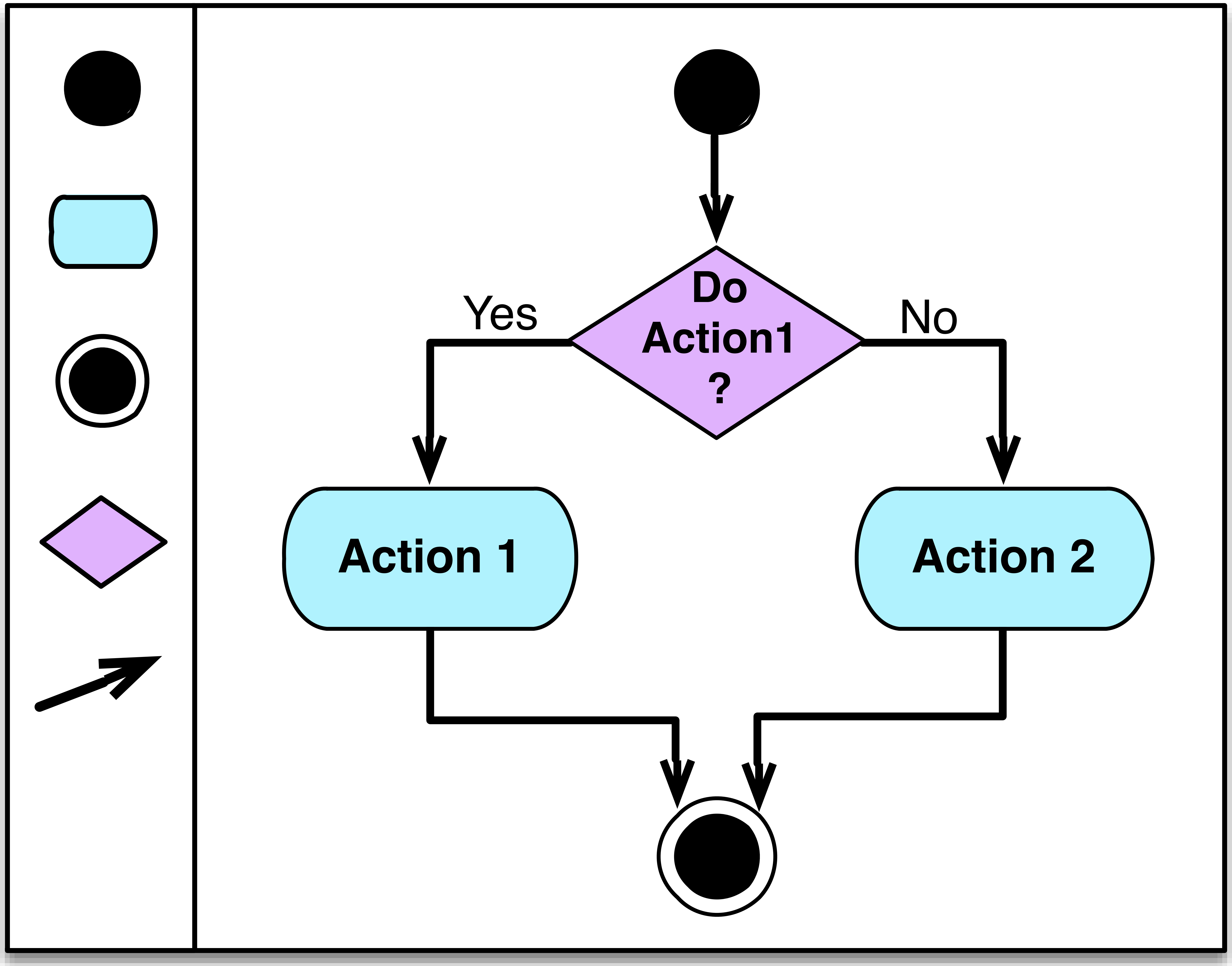
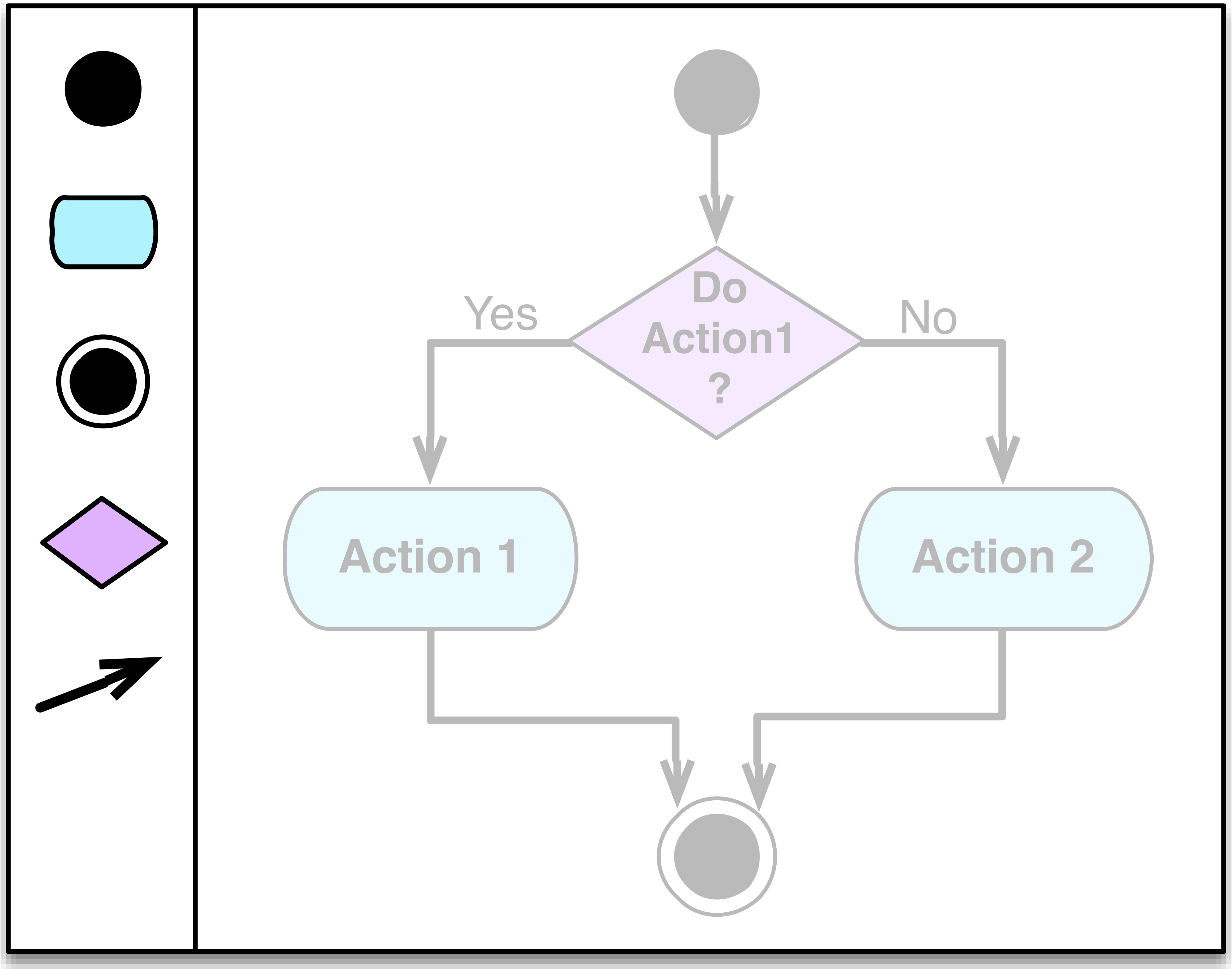
Definition Methods and Implementation of Domain-Specific Modeling Language Tools

Renārs Liepiņš

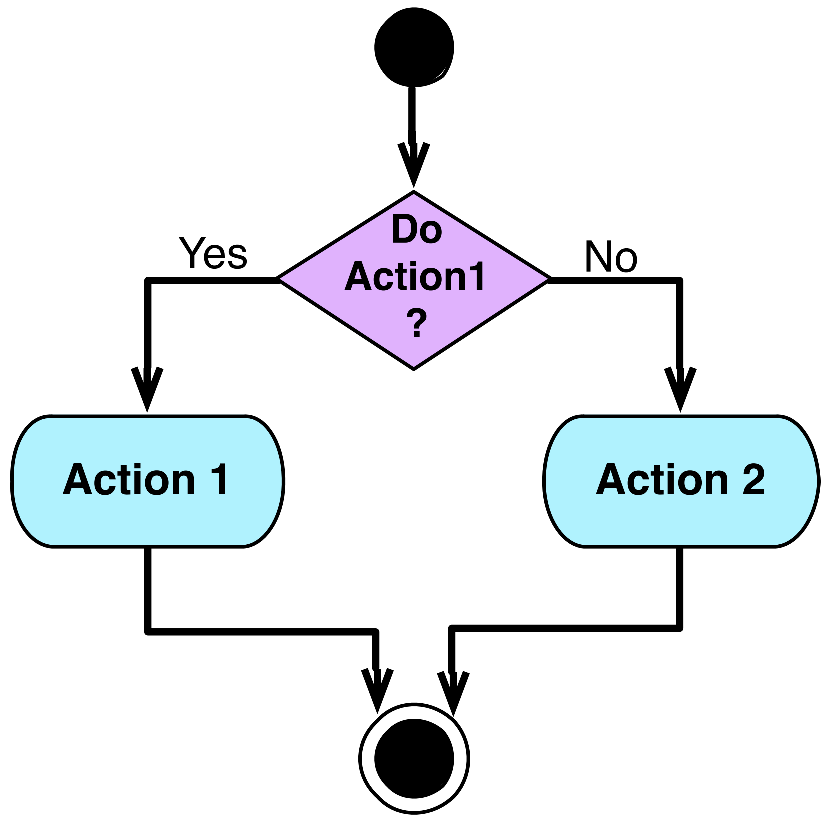


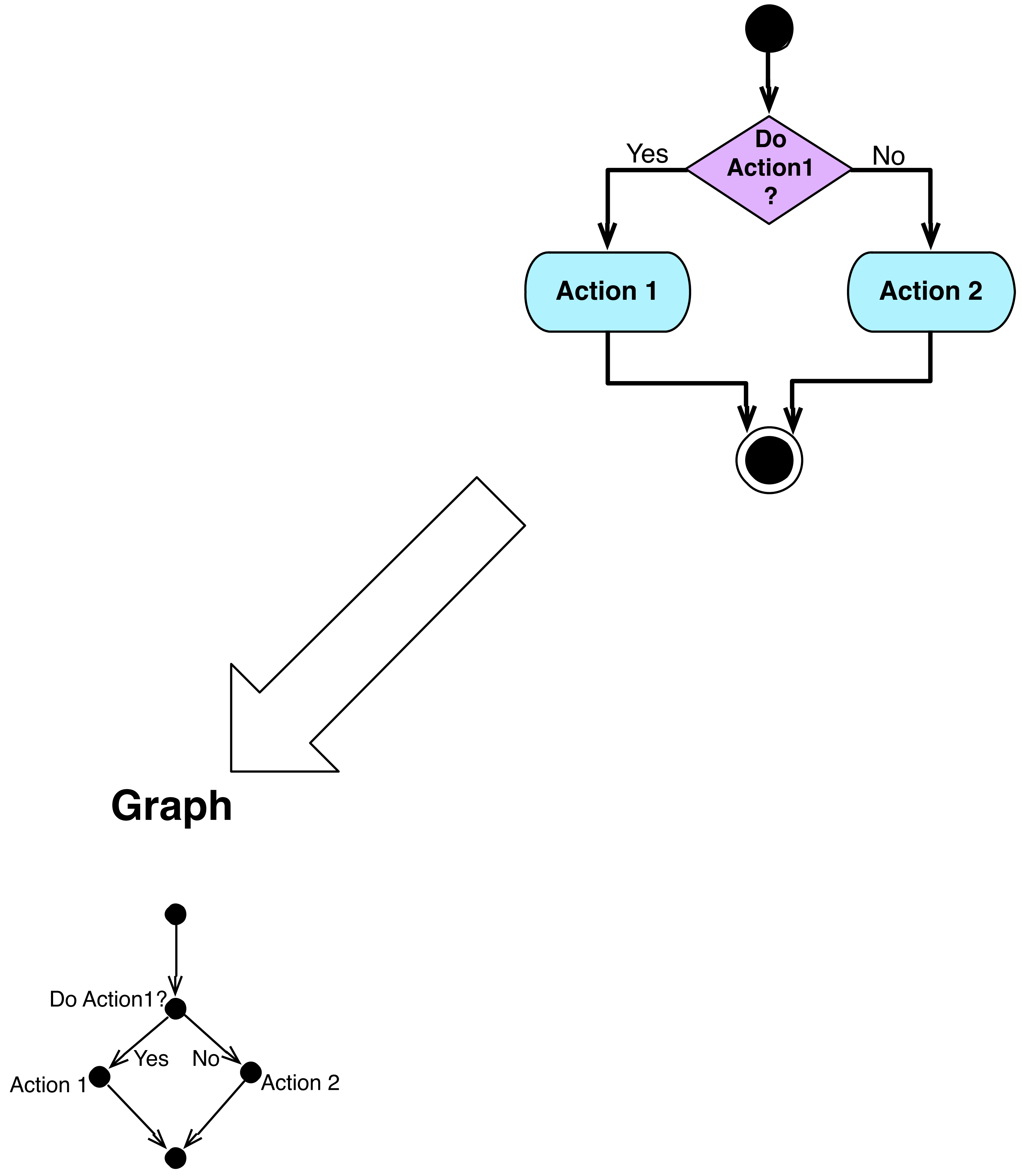

Do Action1 ?
Yes
No
Action 1
Action 2

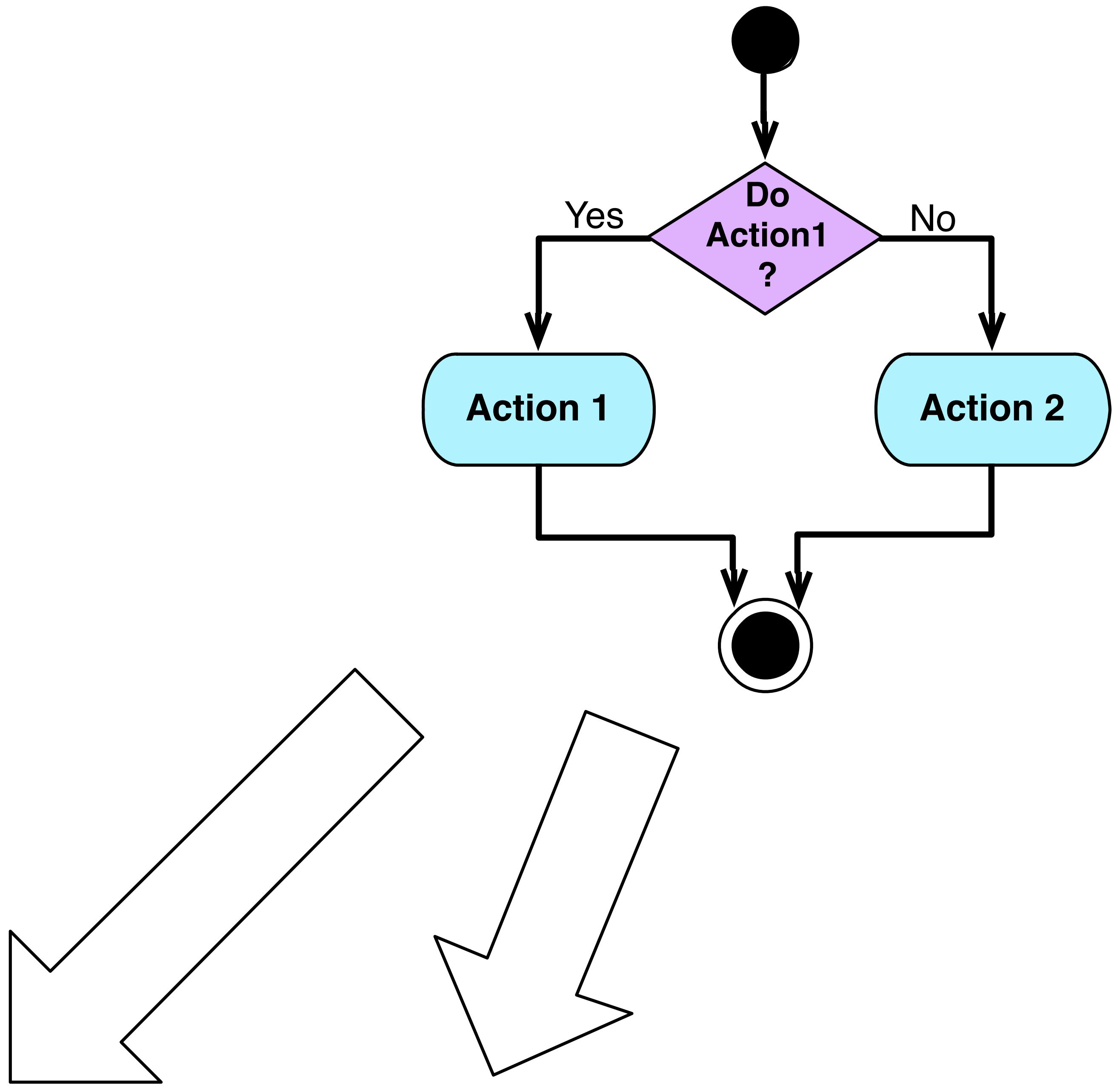

Do Action1 ?
Yes
No
Action 1
Action 2



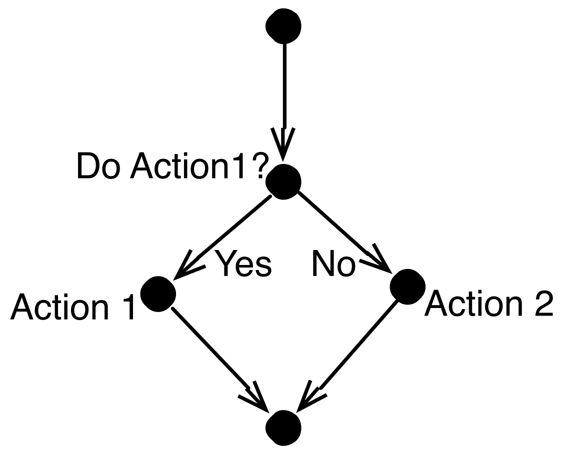
How to build such tools?


Do Action1 ?
Yes
No
Action 1
Action 2


Do Action1 ?
Yes
No
Action 1
Action 2
Graph
Do Action1?
Yes
No
Action 1
Action 2

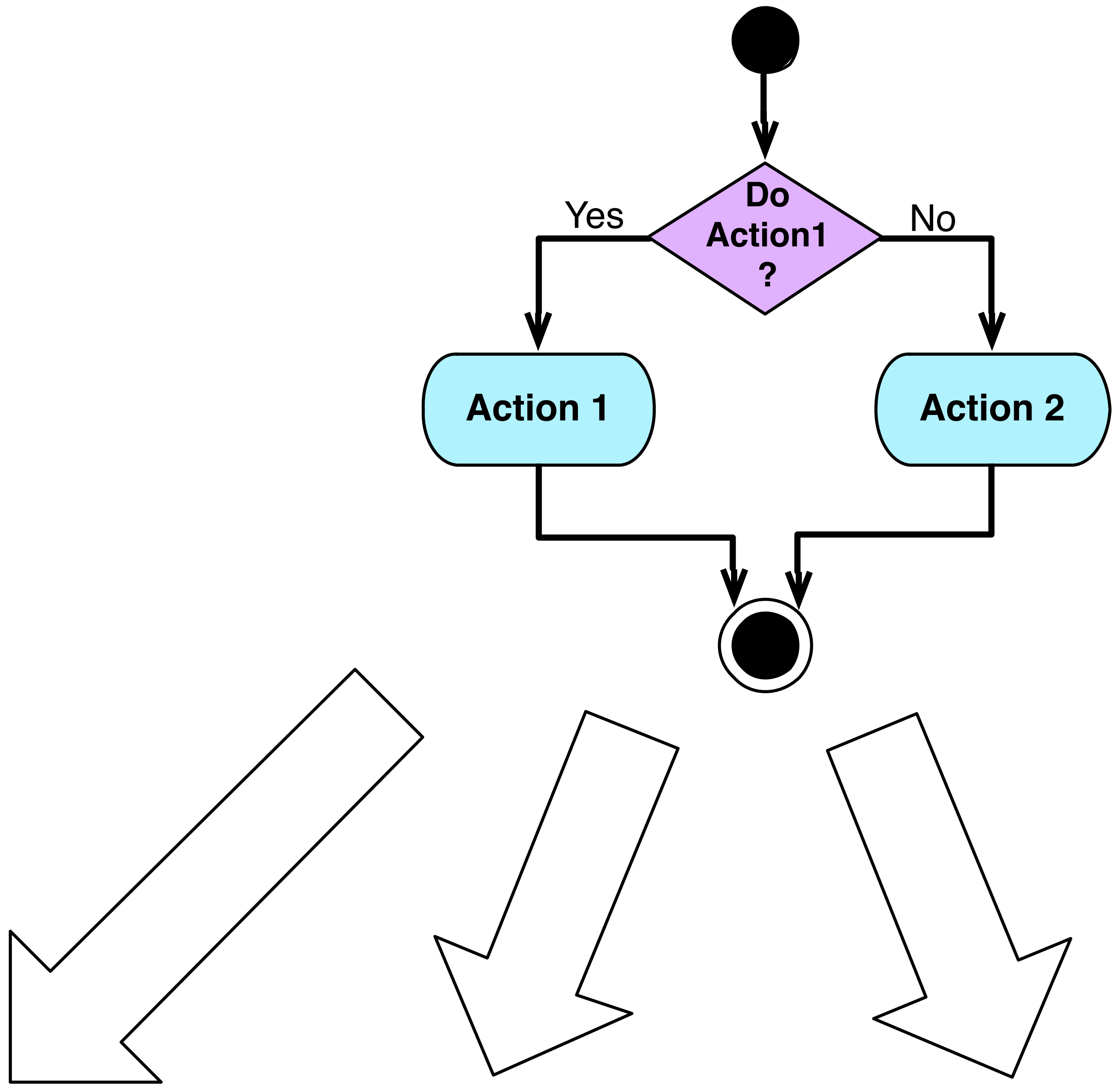


Graph

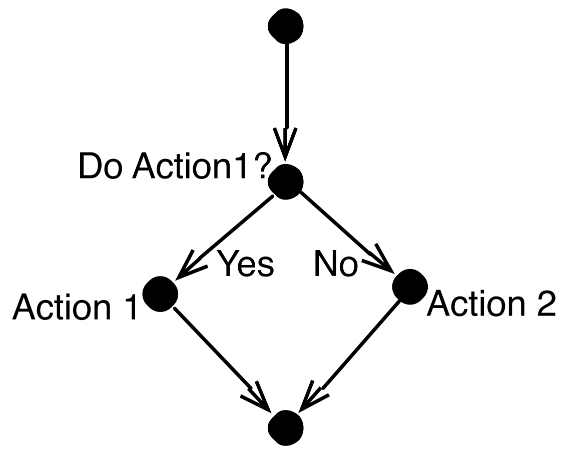


Types

```
FlowChart
Start
End
Decision
Action
Flow
```



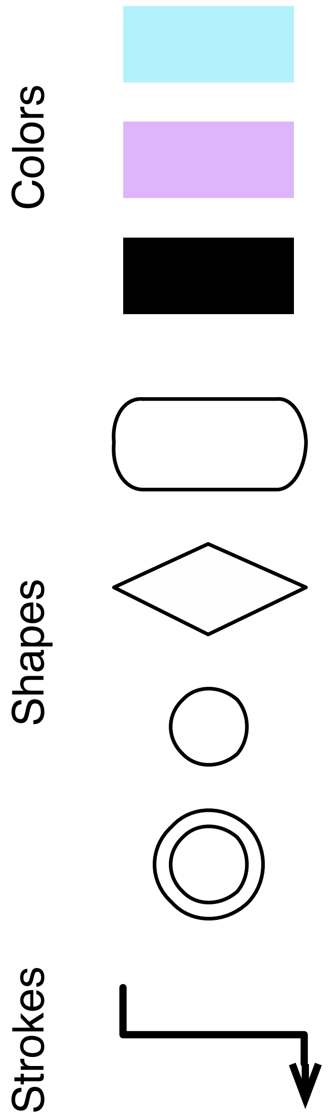
Graph



Types

```
FlowChart
Start
End
Decision
Action
Flow
```

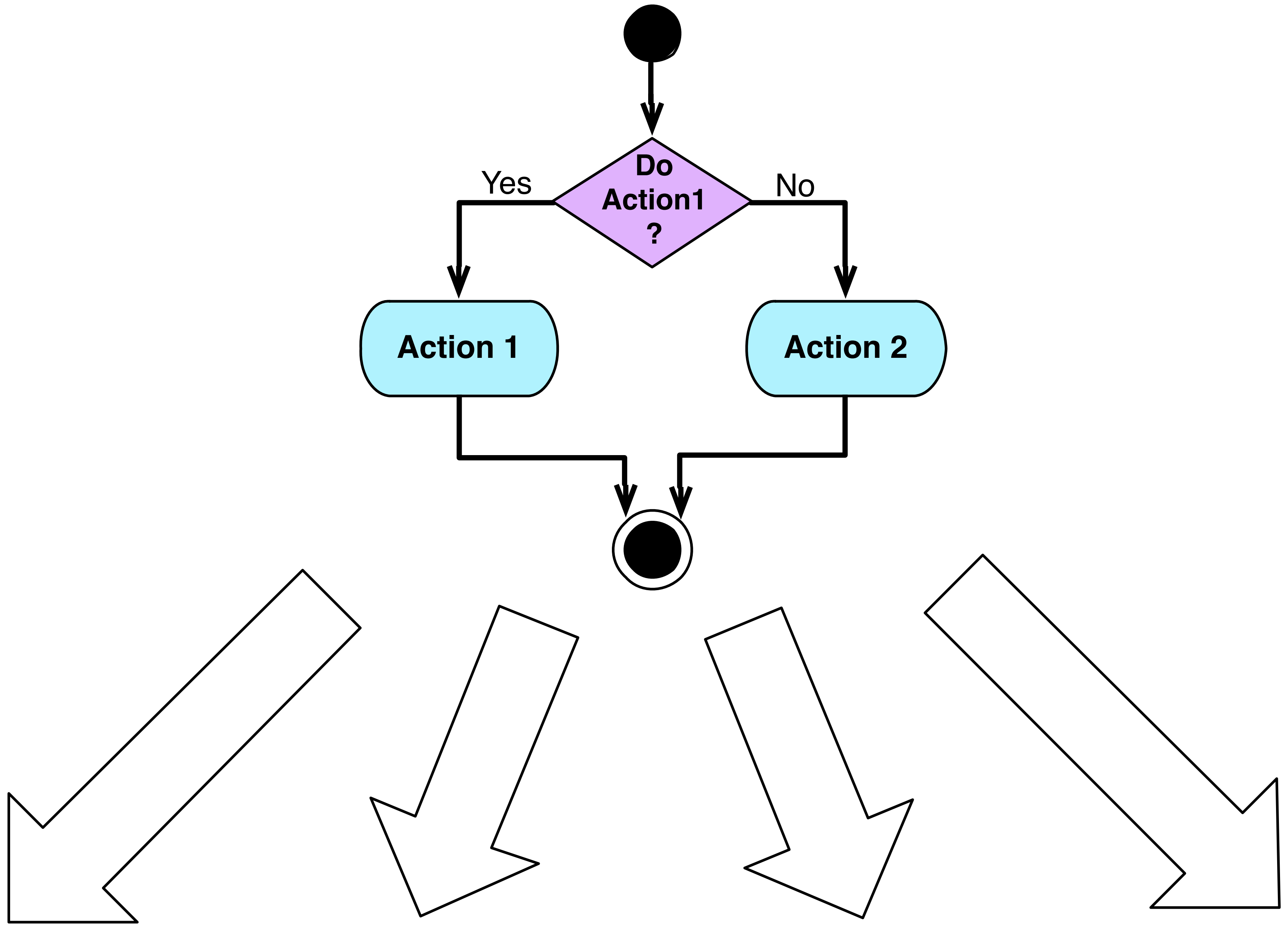
Styles



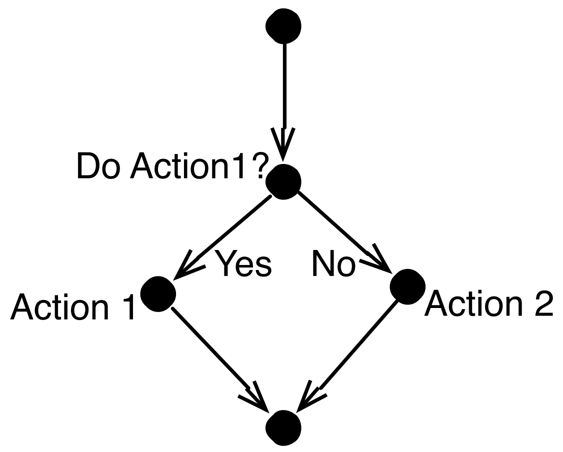
Fonts

Helvetica Bold 12pt

Helvetica Regular 12pt



Graph



Types

`FlowChart`

`Start`

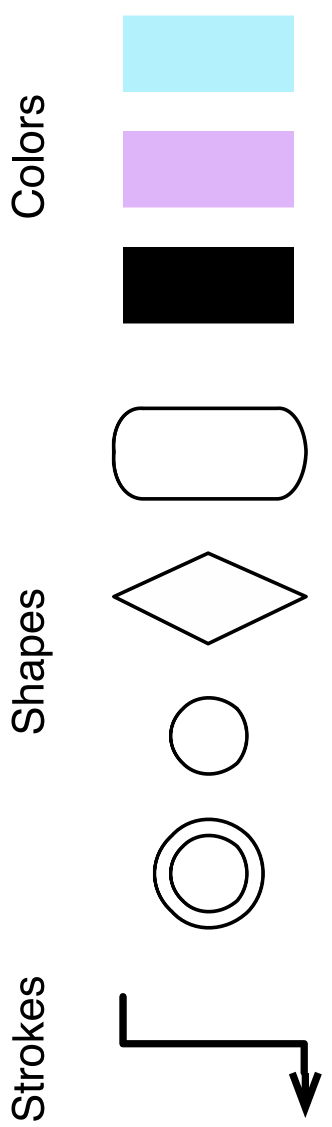
`End`

`Decision`

`Action`

`Flow`

Styles



Fonts

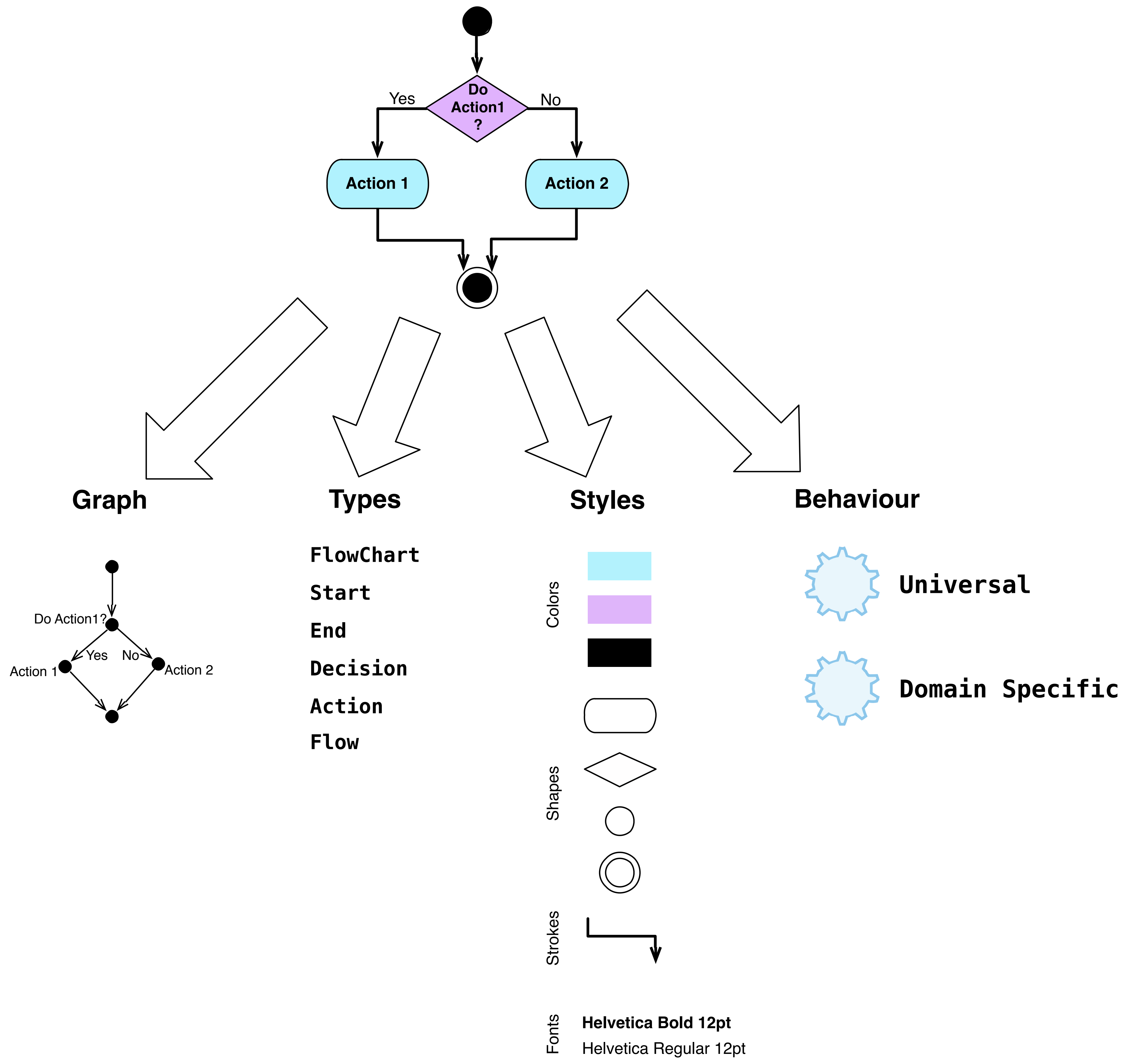
Helvetica Bold 12pt

Helvetica Regular 12pt

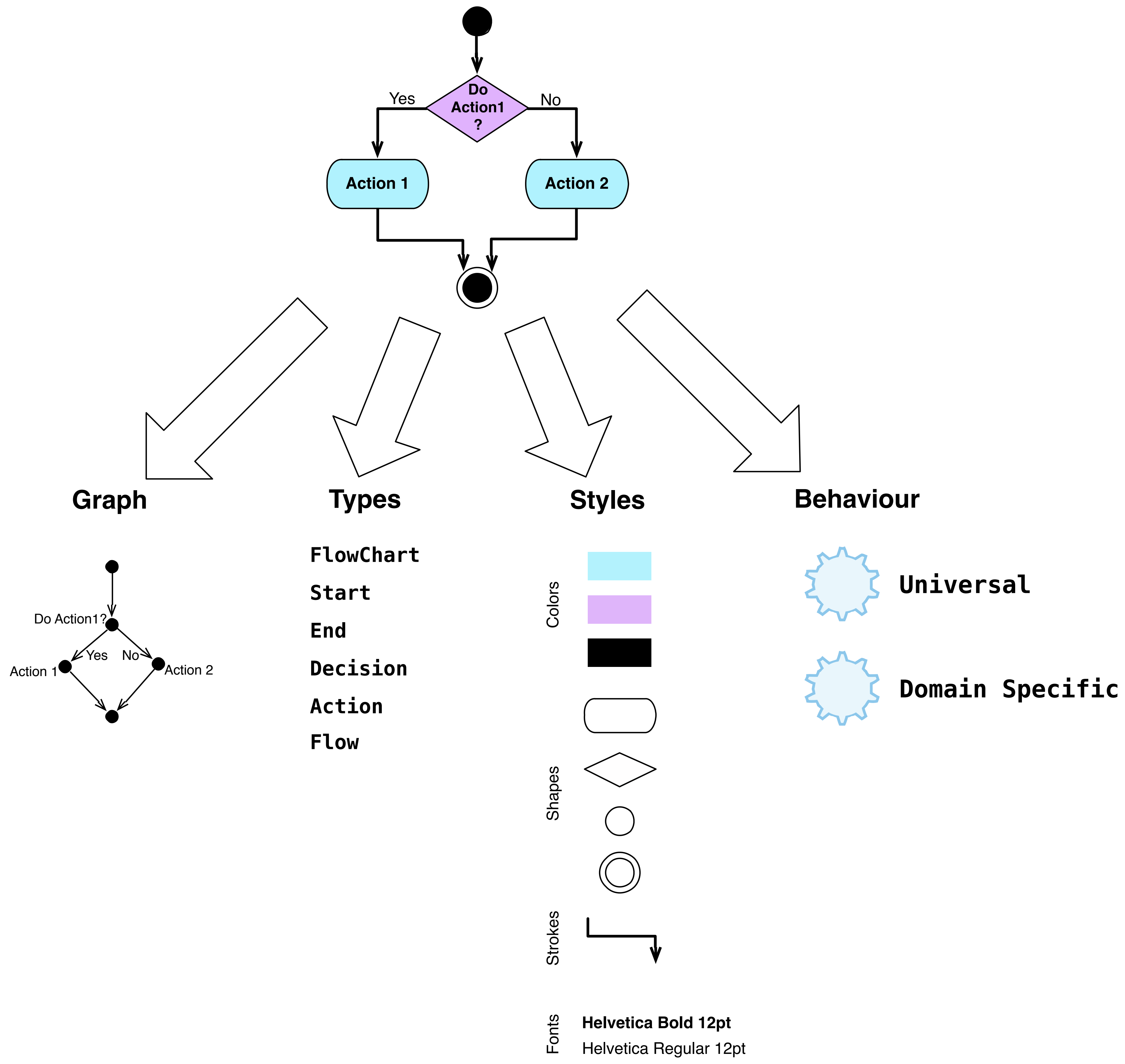
Behaviour

`Universal`

`Domain Specific`

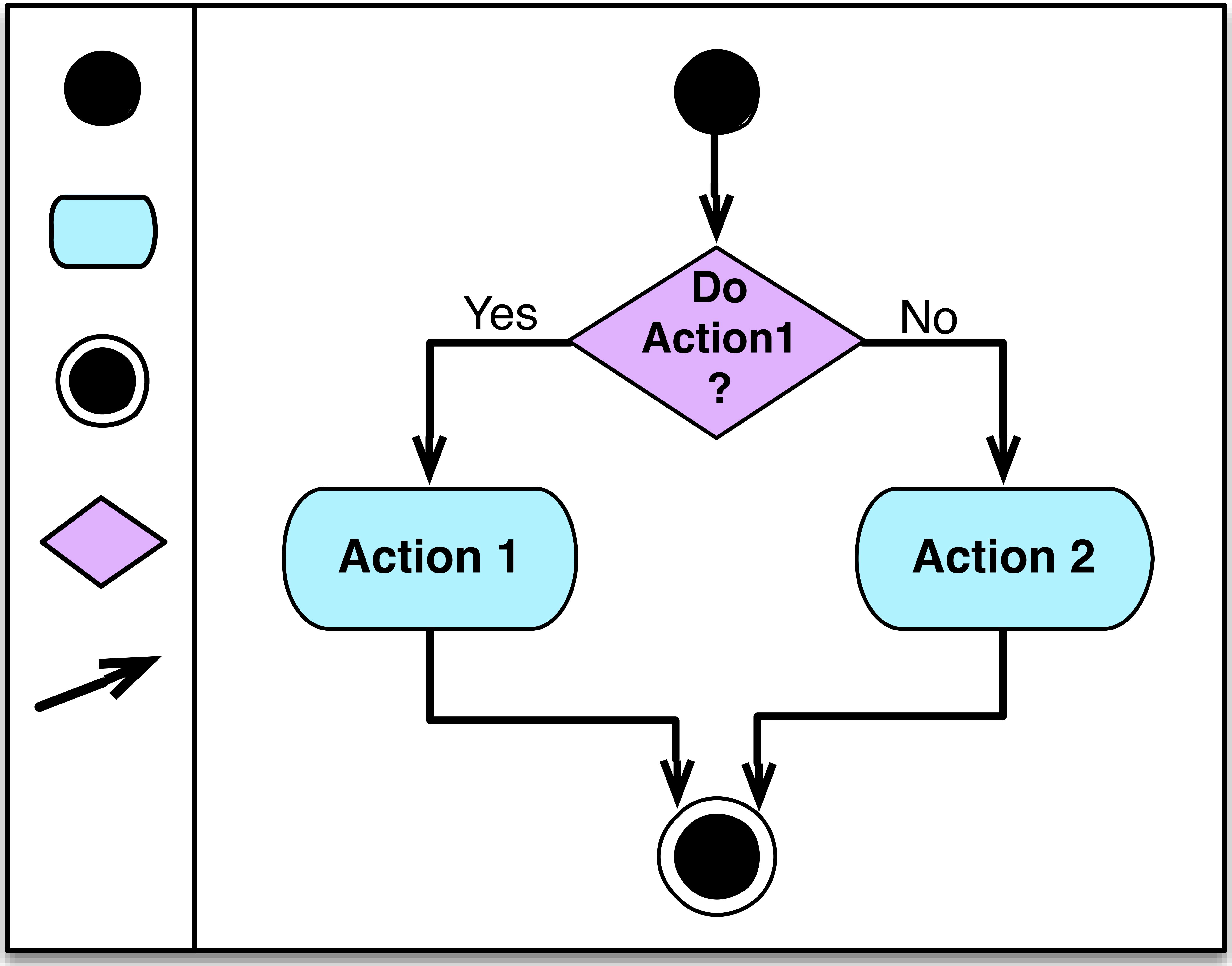


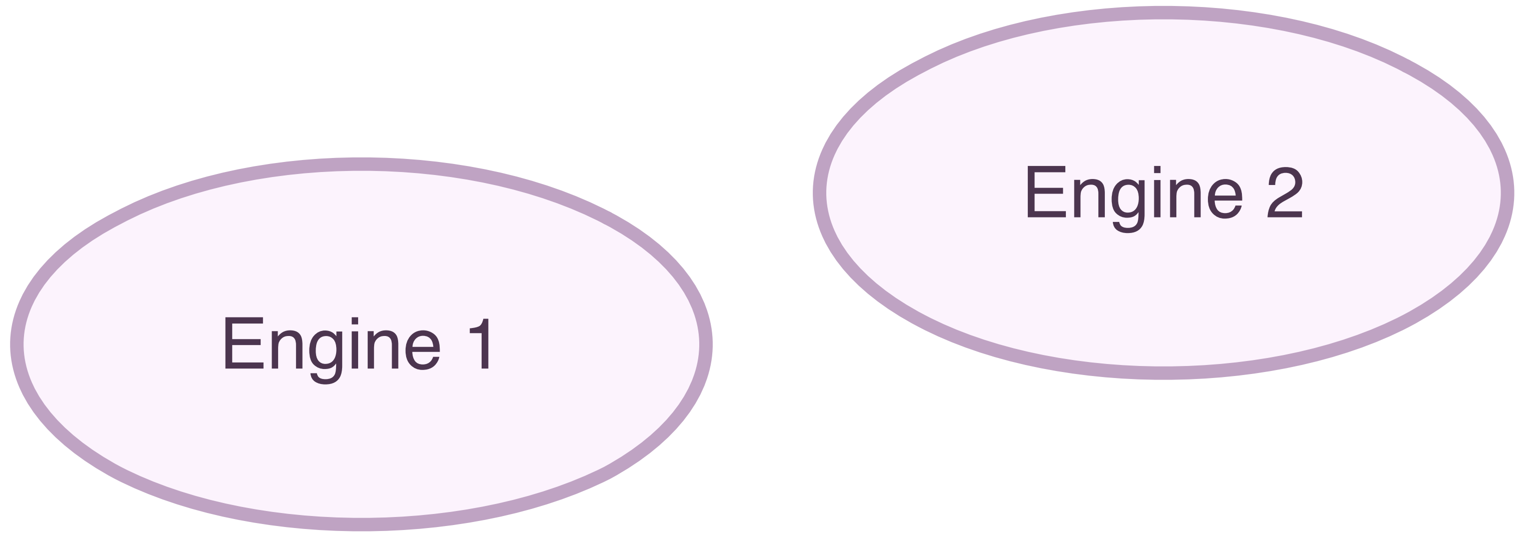
- Model Based Approach



- Model Based Approach
- Ontology Based Approach

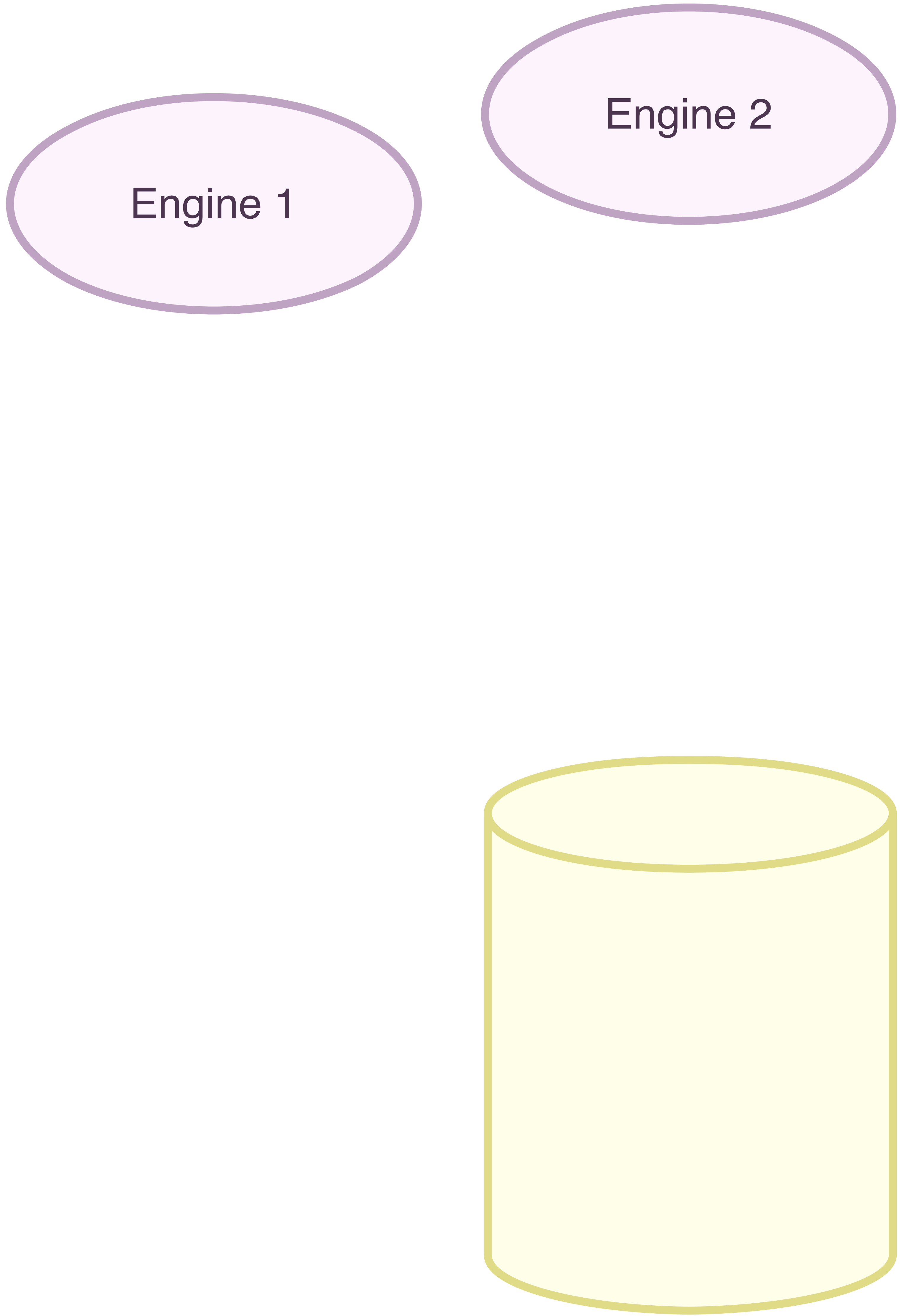
Model Based Tool Building


Do Action1 ?
Yes
No
Action 1
Action 2

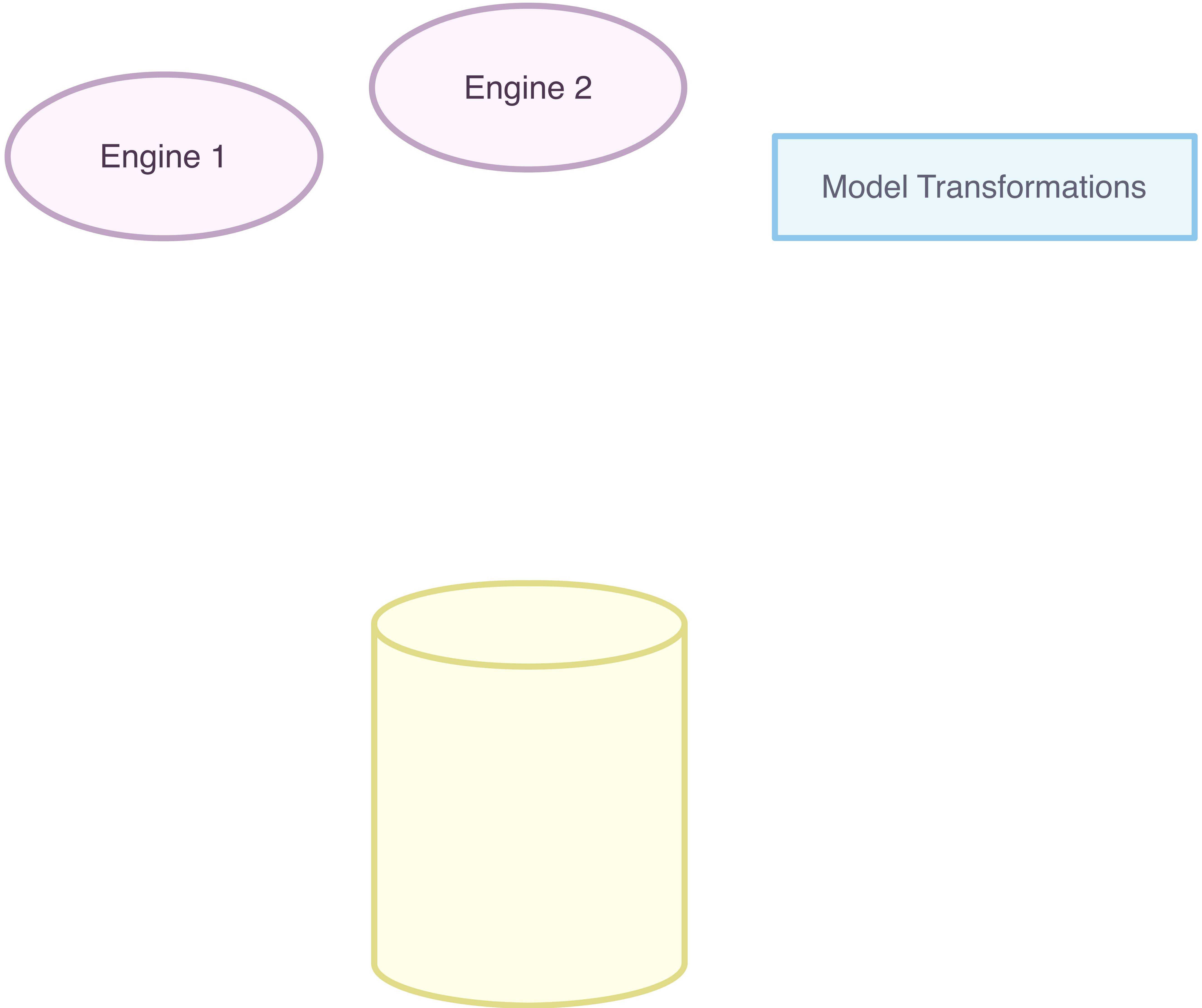


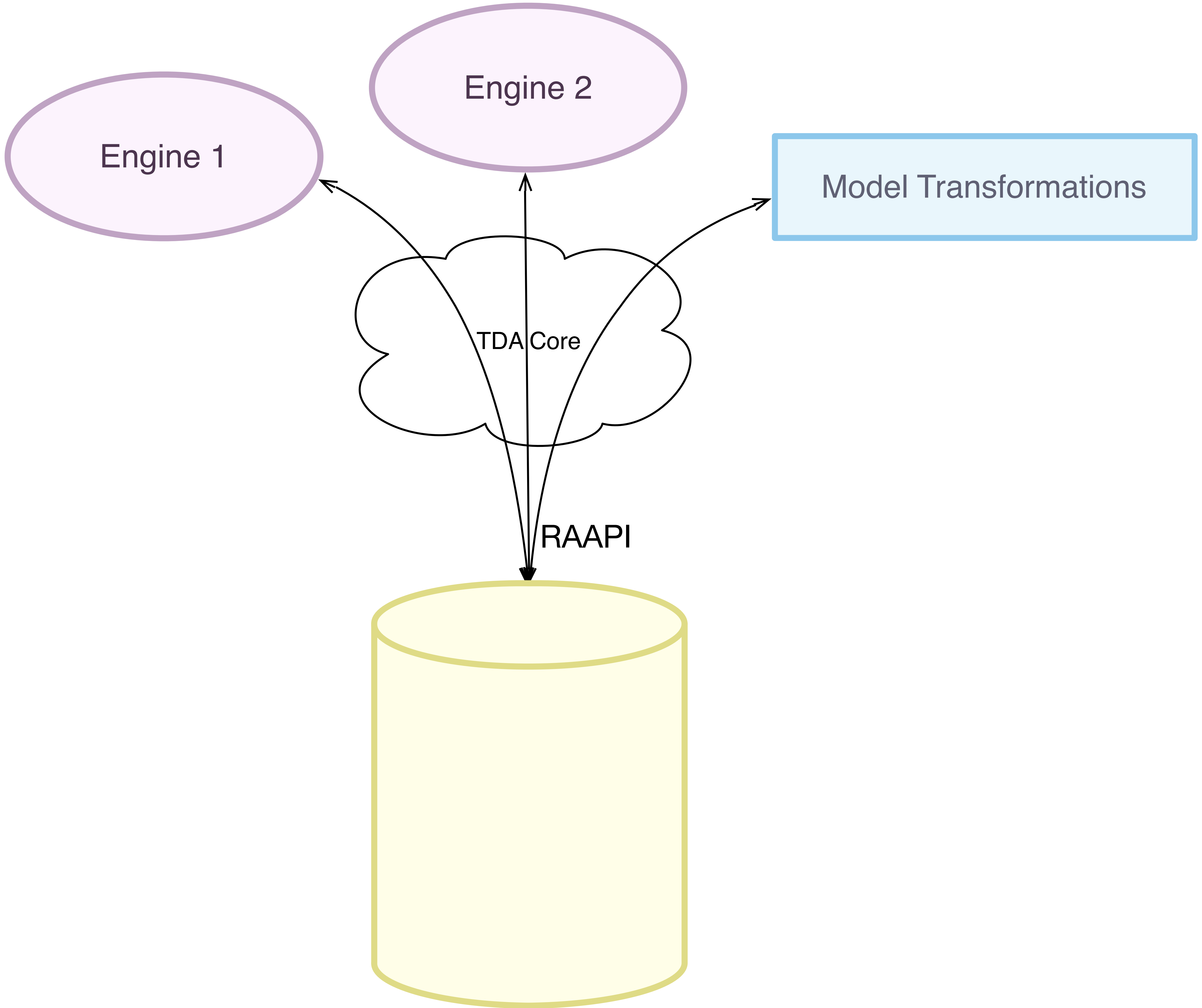
Engine 1

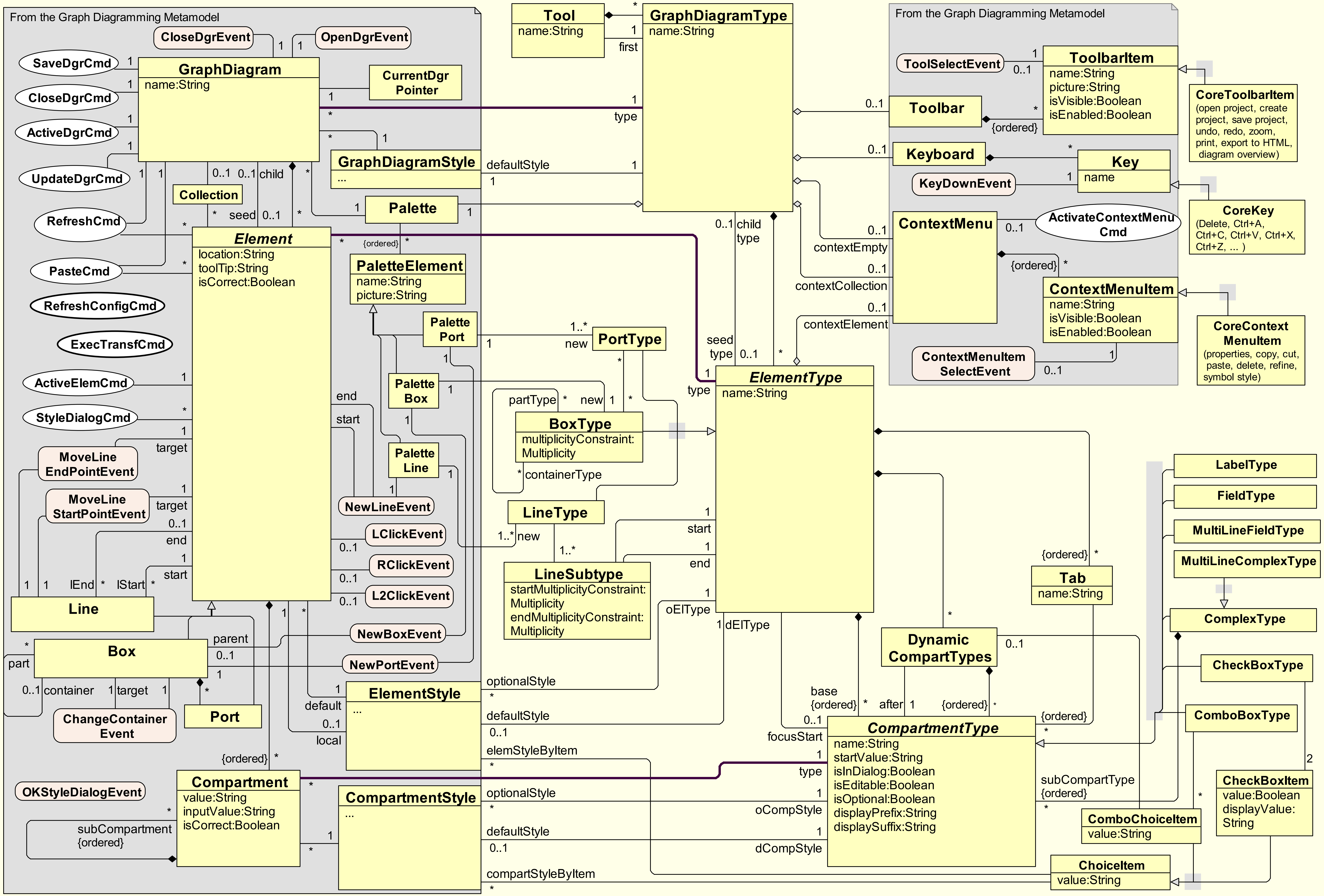
Engine 2



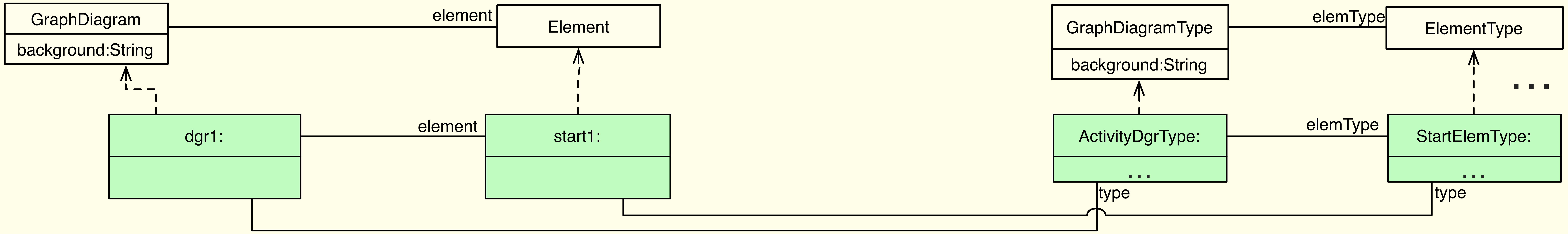
Engine 1
Engine 2


Engine 2
Engine 1
Model Transformations


Engine 2
Engine 1
Model Transformations
TDA Core
RAAPI



From the Graph Diagramming Metamodel
CloseDgrEvent
OpenDgrEvent
SaveDgrCmd
CloseDgrCmd
ActiveDgrCmd
UpdateDgrCmd
RefreshCmd
PasteCmd
RefreshConfigCmd
ExecTransfCmd
ActiveElemCmd
StyleDialogCmd
MoveLine EndPointEvent
MoveLine StartPointEvent
GraphDiagram
name:String
CurrentDgr Pointer
GraphDiagramStyle
...
Collection
Palette
PaletteElement
name:String
picture:String
Palette Port
Palette Box
Palette Line
Element
location:String
toolTip:String
isCorrect:Boolean
NewLineEvent
LClickEvent
RClickEvent
L2ClickEvent
NewBoxEvent
NewPortEvent
Line
Box
Port
ChangeContainer Event
ElementStyle
...
OKStyleDialogEvent
Compartment
value:String
inputValue:String
isCorrect:Boolean
CompartmentStyle
...
subCompartment {ordered}
seed
child
target
end
start
lEnd
lStart
parent
part
container
default
local
{ordered}
Tool
name:String
first
GraphDiagramType
name:String
type
defaultStyle
PortType
new
BoxType
multiplicityConstraint: Multiplicity
partType
containerType
LineType
LineSubtype
startMultiplicityConstraint: Multiplicity
endMultiplicityConstraint: Multiplicity
ElementType
name:String
seed type
child type
oElType
dElType
optionalStyle
elemStyleByItem
compartStyleByItem
oCompStyle
dCompStyle
focusStart
From the Graph Diagramming Metamodel
ToolSelectEvent
ToolbarItem
name:String
picture:String
isVisible:Boolean
isEnabled:Boolean
CoreToolbarItem
(open project, create project, save project, undo, redo, zoom, print, export to HTML, diagram overview)
Toolbar
Keyboard
Key
name
KeyDownEvent
CoreKey
(Delete, Ctrl+A, Ctrl+C, Ctrl+V, Ctrl+X, Ctrl+Z, ...)
ContextMenu
ActivateContextMenu Cmd
ContextMenuItem
name:String
isVisible:Boolean
isEnabled:Boolean
CoreContext MenuItem
(properties, copy, cut, paste, delete, refine, symbol style)
ContextMenuItem SelectEvent
contextEmpty
contextCollection
contextElement
Tab
name:String
Dynamic CompartTypes
base {ordered}
after
CompartmentType
name:String
startValue:String
isInDialog:Boolean
isEditable:Boolean
isOptional:Boolean
displayPrefix:String
displaySuffix:String
subCompartType {ordered}
LabelType
FieldType
MultiLineFieldType
MultiLineComplexType
ComplexType
CheckBoxType
ComboBoxType
CheckBoxItem
value:Boolean
displayValue: String
ComboChoiceItem
value:String
ChoiceItem
value:String



GraphDiagram

background:String

element

Element

dgr1:

element

start1:

GraphDiagramType

background:String

elemType

ElementType

. . .

ActivityDgrType:

. . .

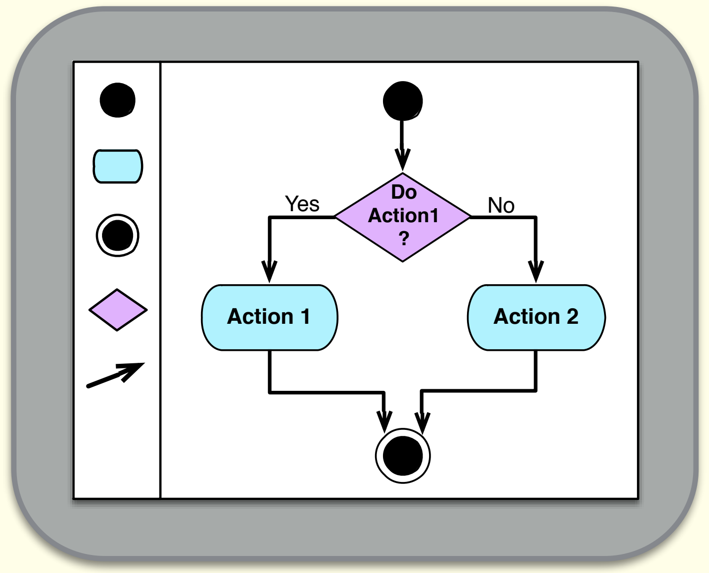
type

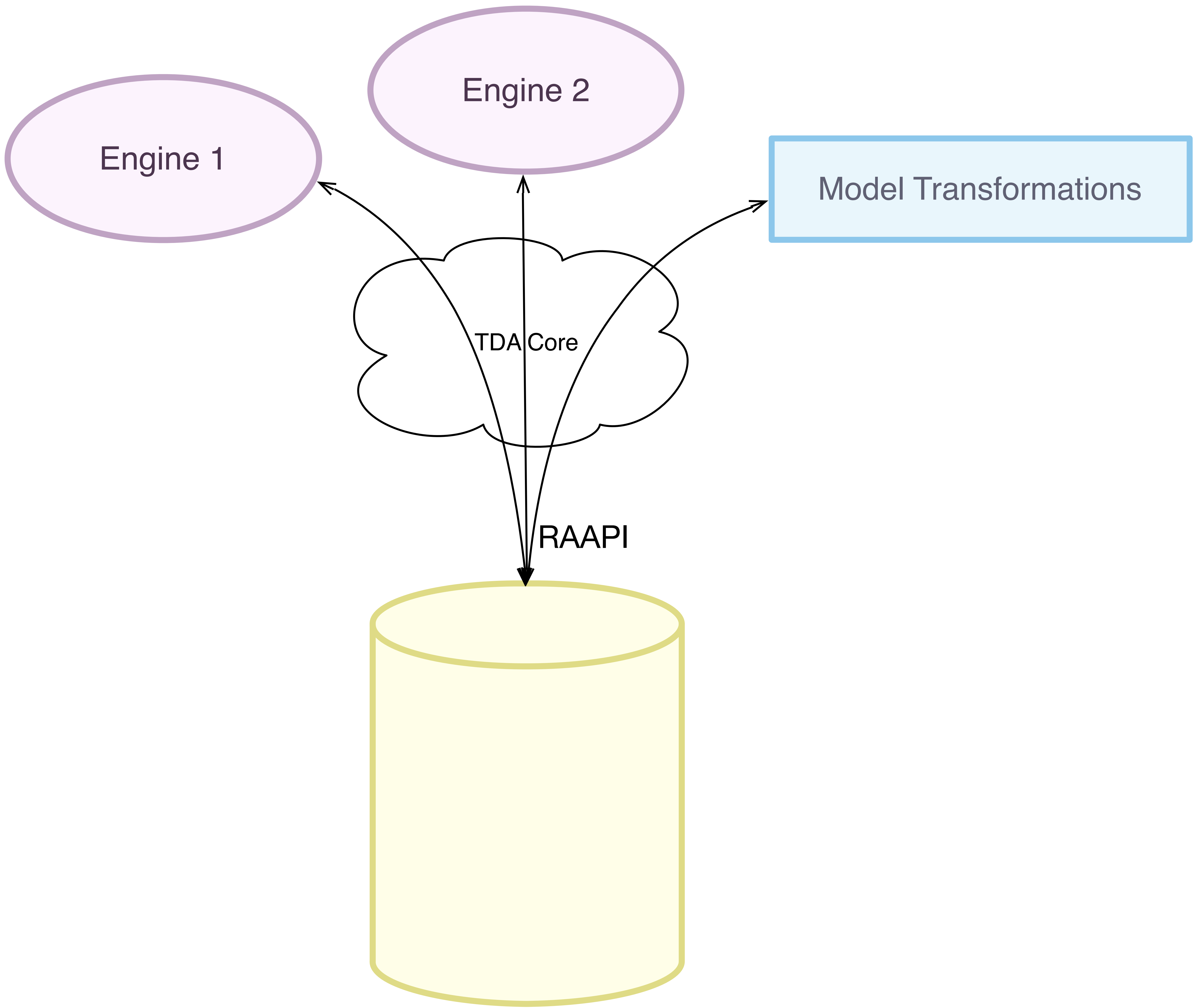
elemType

StartElemType:

. . .

type

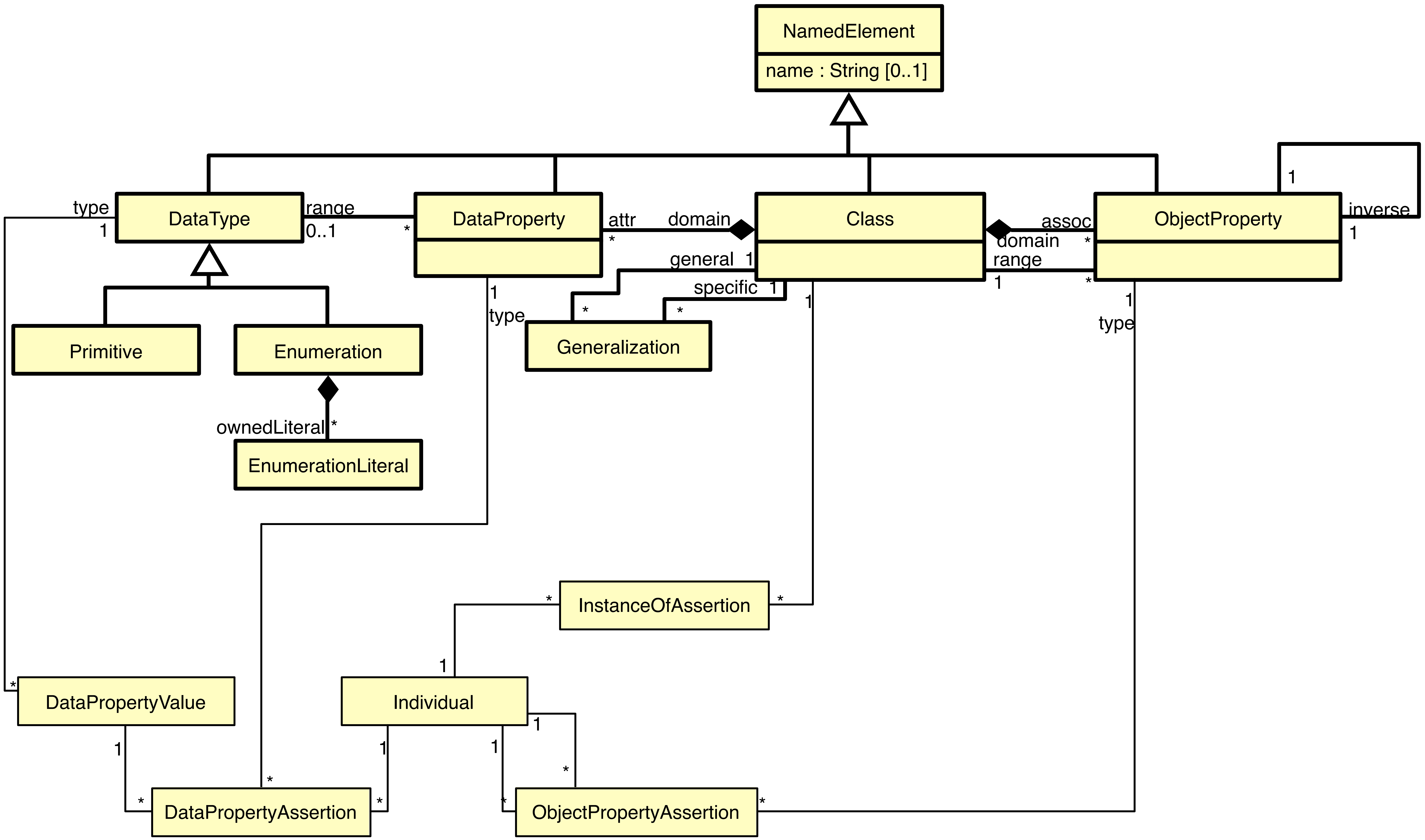



Engine 1
Engine 2
Model Transformations
TDA Core
RAAPI

Language for

Model Transformations

IQuery


NamedElement
name : String [0..1]
type
1
DataType
range
0..1
*
DataProperty
attr
*
domain
Class
assoc
domain
*
range
1
*
ObjectProperty
1
inverse
1
general
1
specific
1
*
*
Generalization
1
type
1
1
type
Primitive
Enumeration
ownedLiteral
*
EnumerationLiteral
*
InstanceOfAssertion
*
1
Individual
*
DataPropertyValue
1
1
1
1
*
*
DataPropertyAssertion
*
*
*
ObjectPropertyAssertion
*

One object → **Collections**
of objects

Design

- Primitive Selectors
- Selector Combinators
- lQuery Selector Shorthand

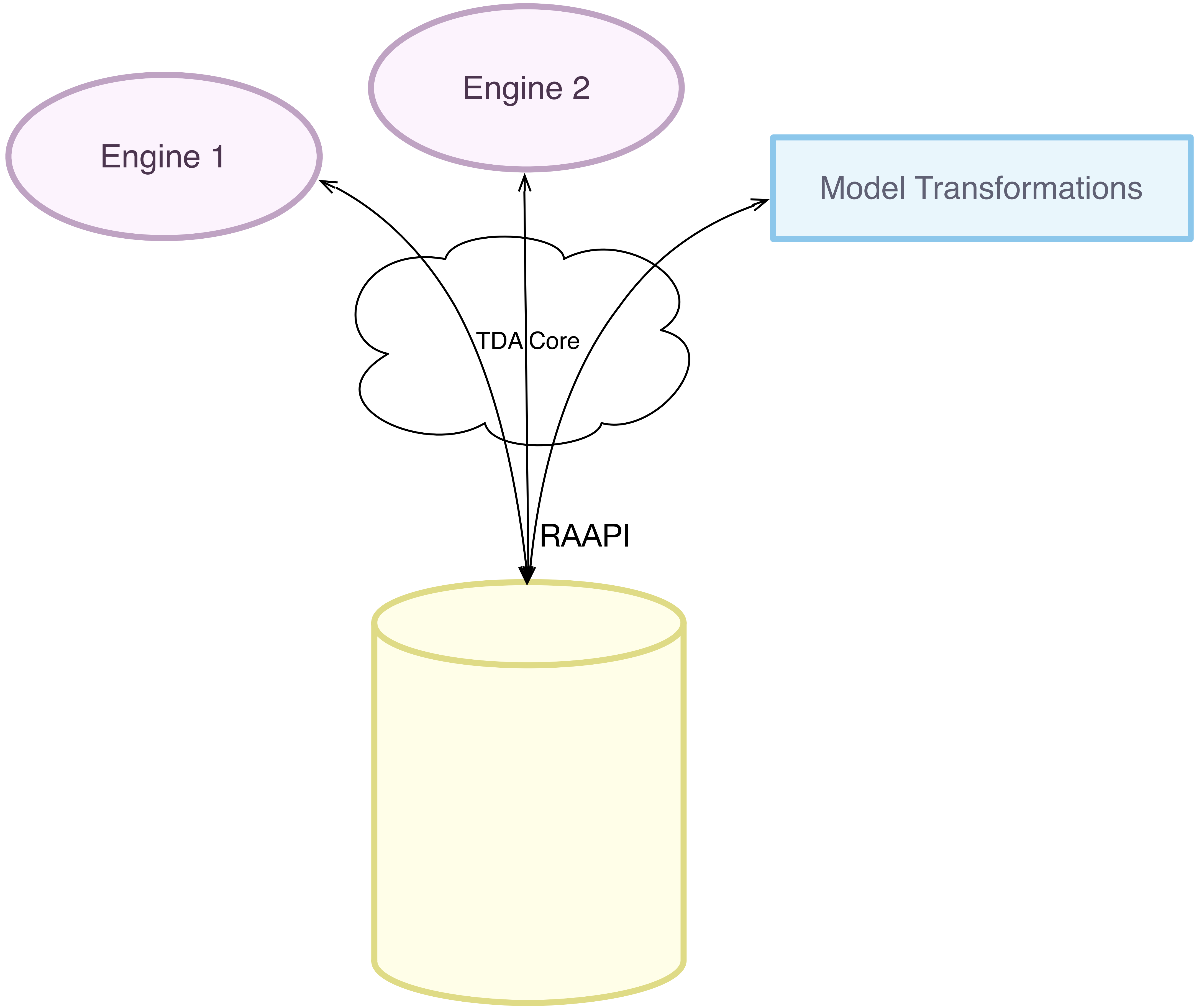
```
obj_col
   :find(“Node /compartment [input = C2]”)
   :attr(“input”, “Class2”)
```

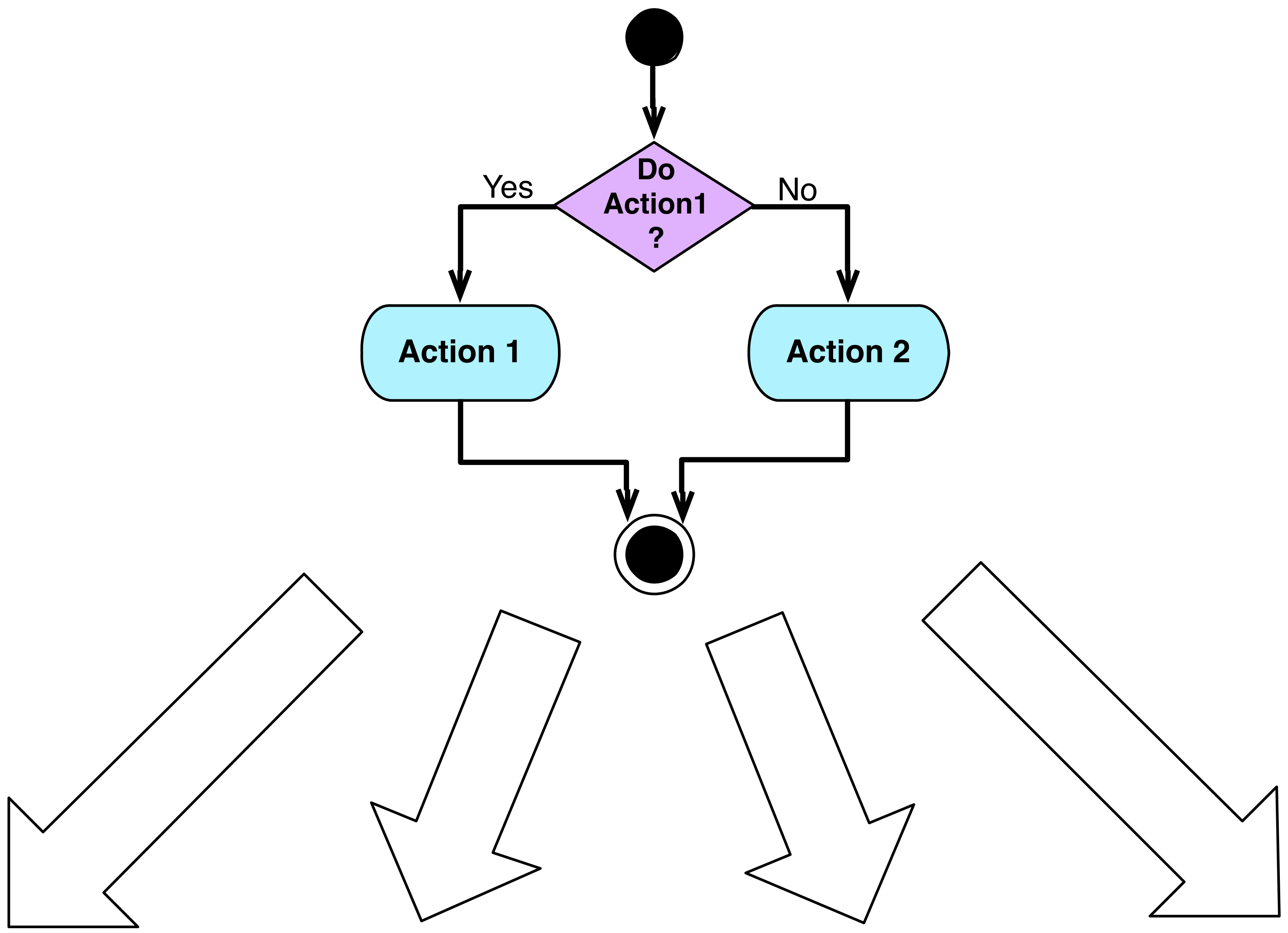
Results

- Transformation Language for Tool Building
- Extensively validated

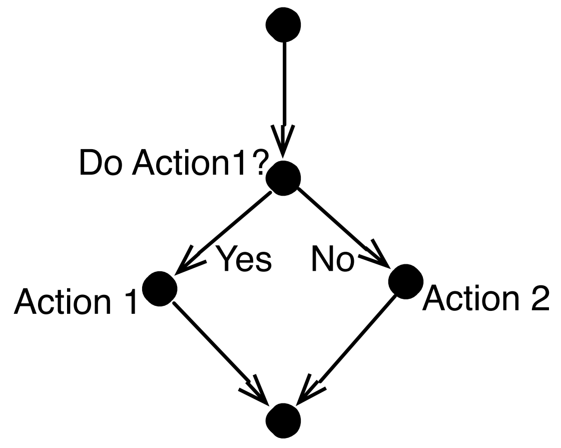
Model Based Tool Building – Results

- Fast & easy tool definition
- Multiple tools used in industry





Graph



Types

FlowChart

Start

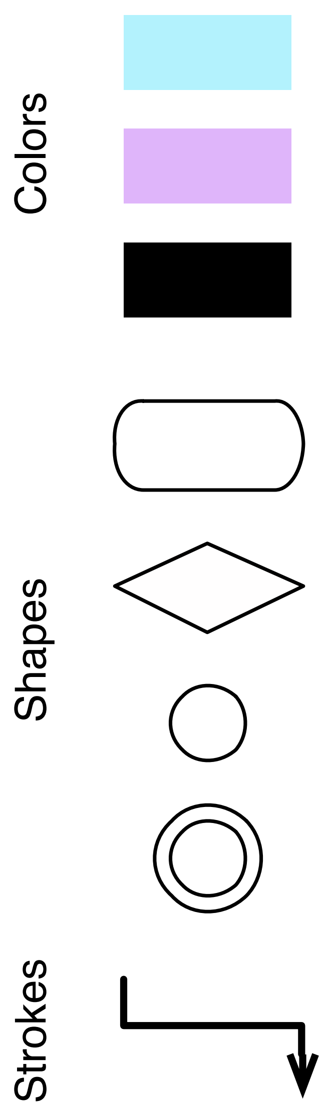
End

Decision

Action

Flow

Styles



Fonts

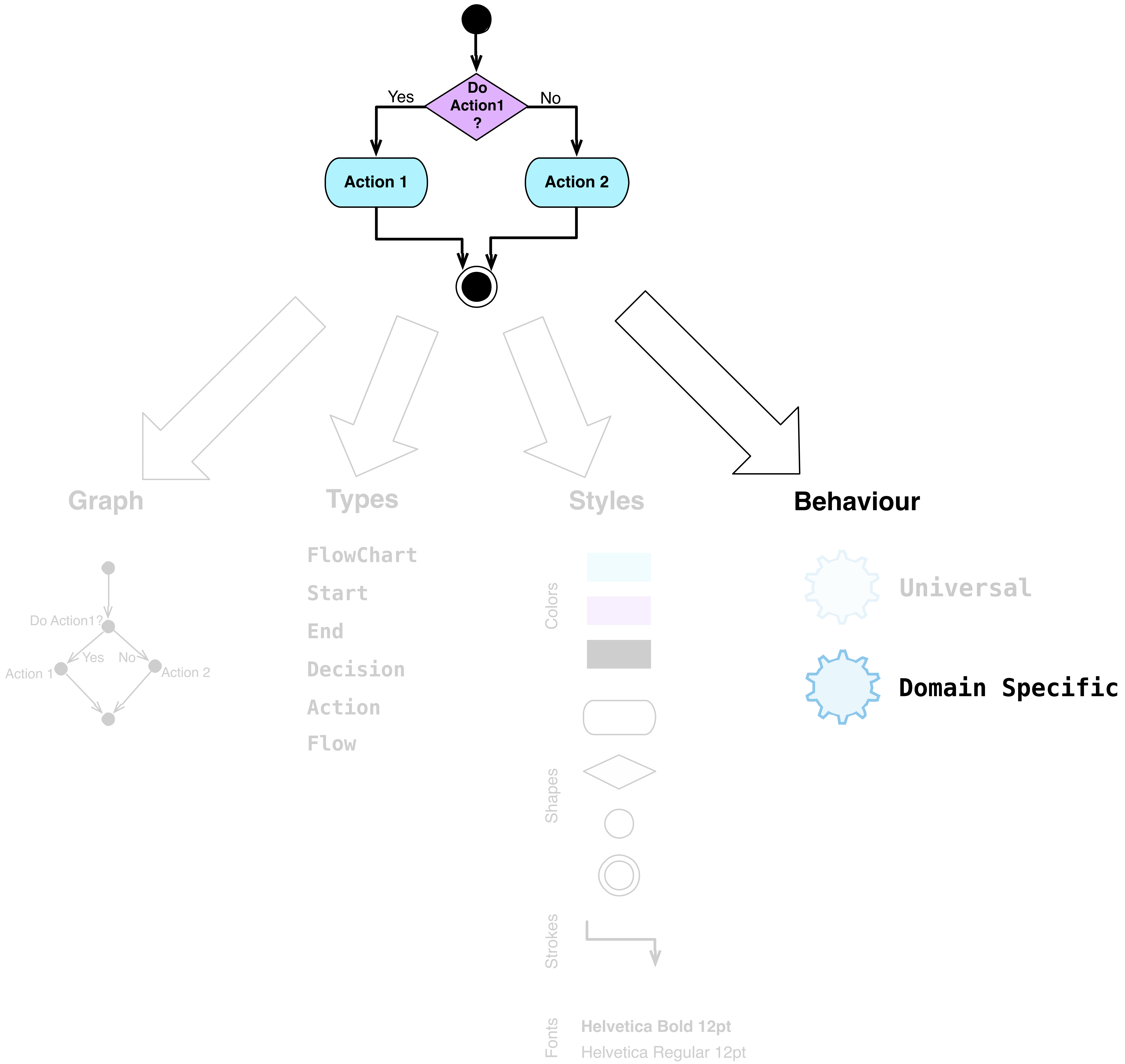
Helvetica Bold 12pt

Helvetica Regular 12pt

Behaviour

Universal

Domain Specific



Graph

Types

- FlowChart
- Start
- End
- Decision
- Action
- Flow

Styles

Helvetica Bold 12pt

Helvetica Regular 12pt

Behaviour

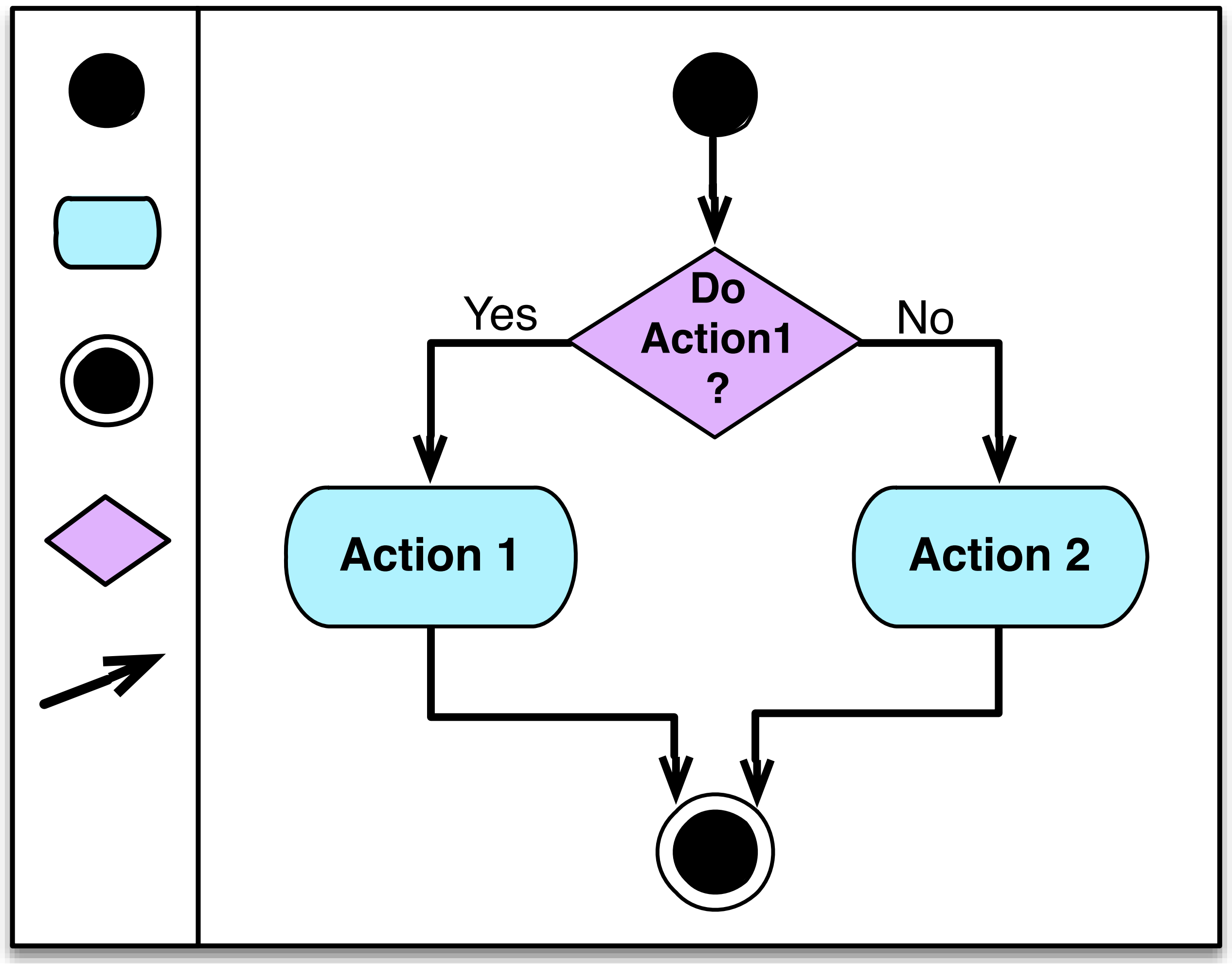
Universal

Domain Specific

Domain Specific Behavior

- Dynamic styles
- Semantic validation

Basic Semantic Validation Example



Every **activity diagram** has exactly one **start** element.

Every **activity diagram** has exactly one **end** element.

Ontology Based
Tool Building

What is Needed

- UML based notation and metamodel for OWL

What is Needed

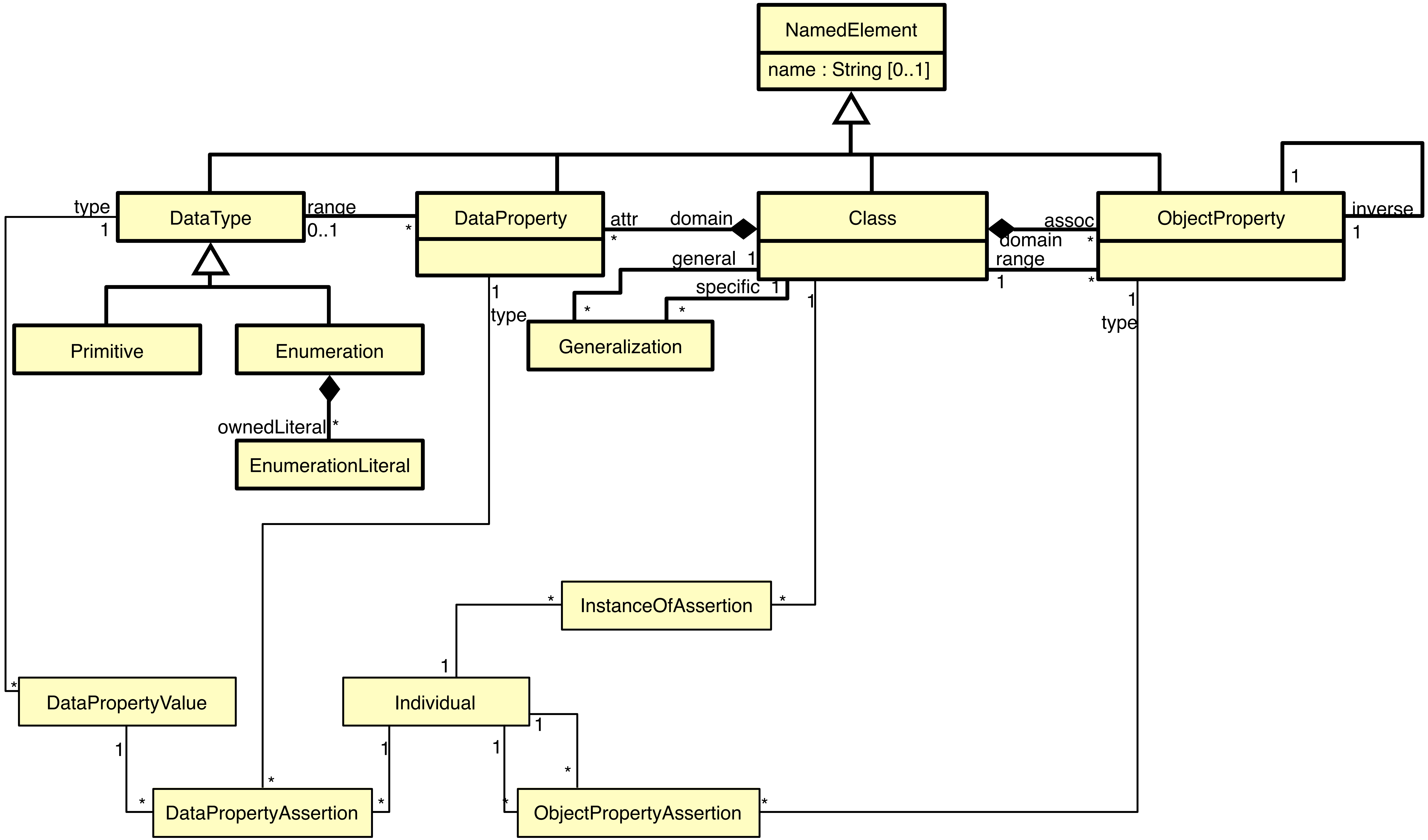
- UML based notation and metamodel for OWL
- Integration of lQuery with OWL

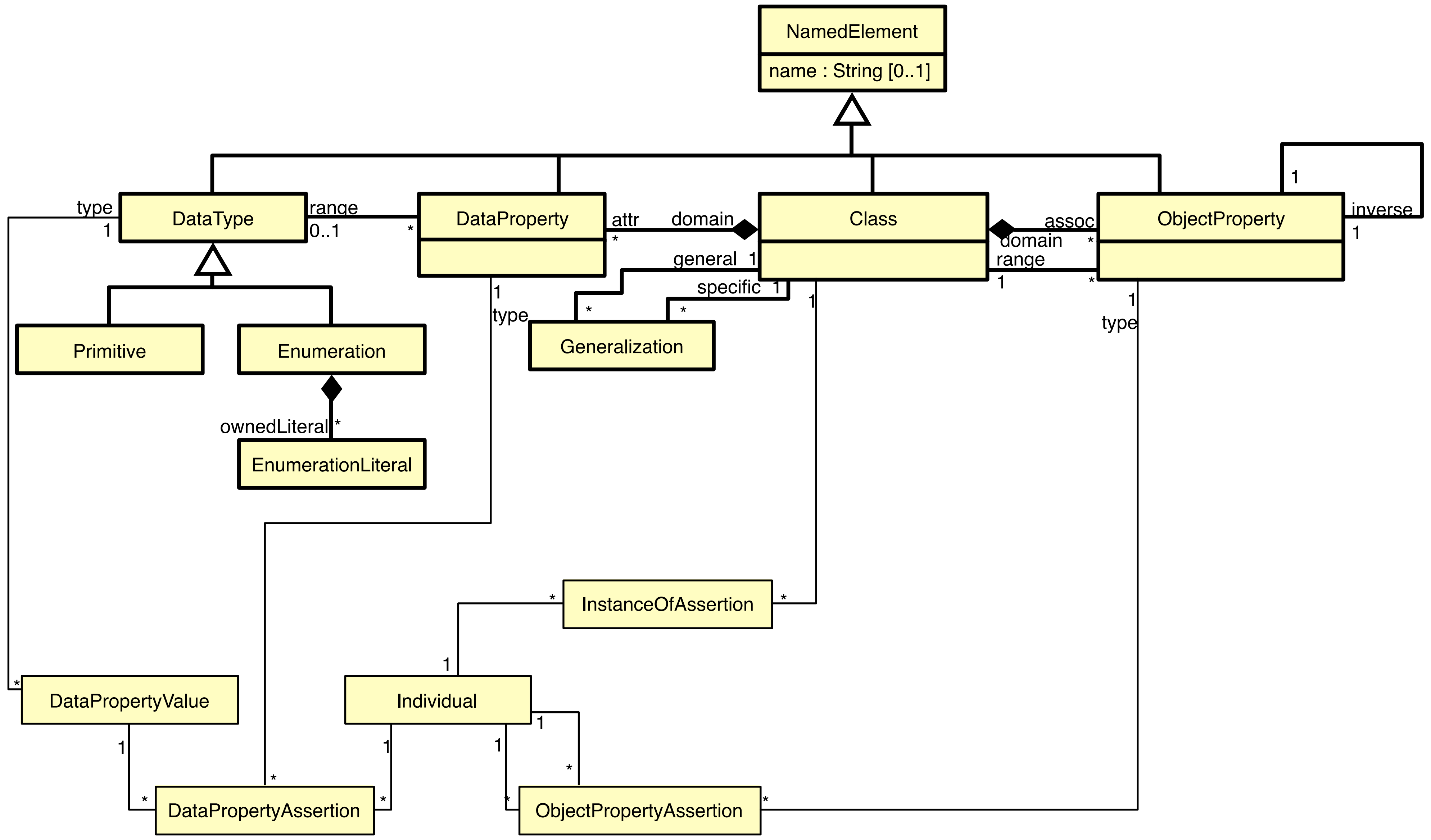
What is Needed

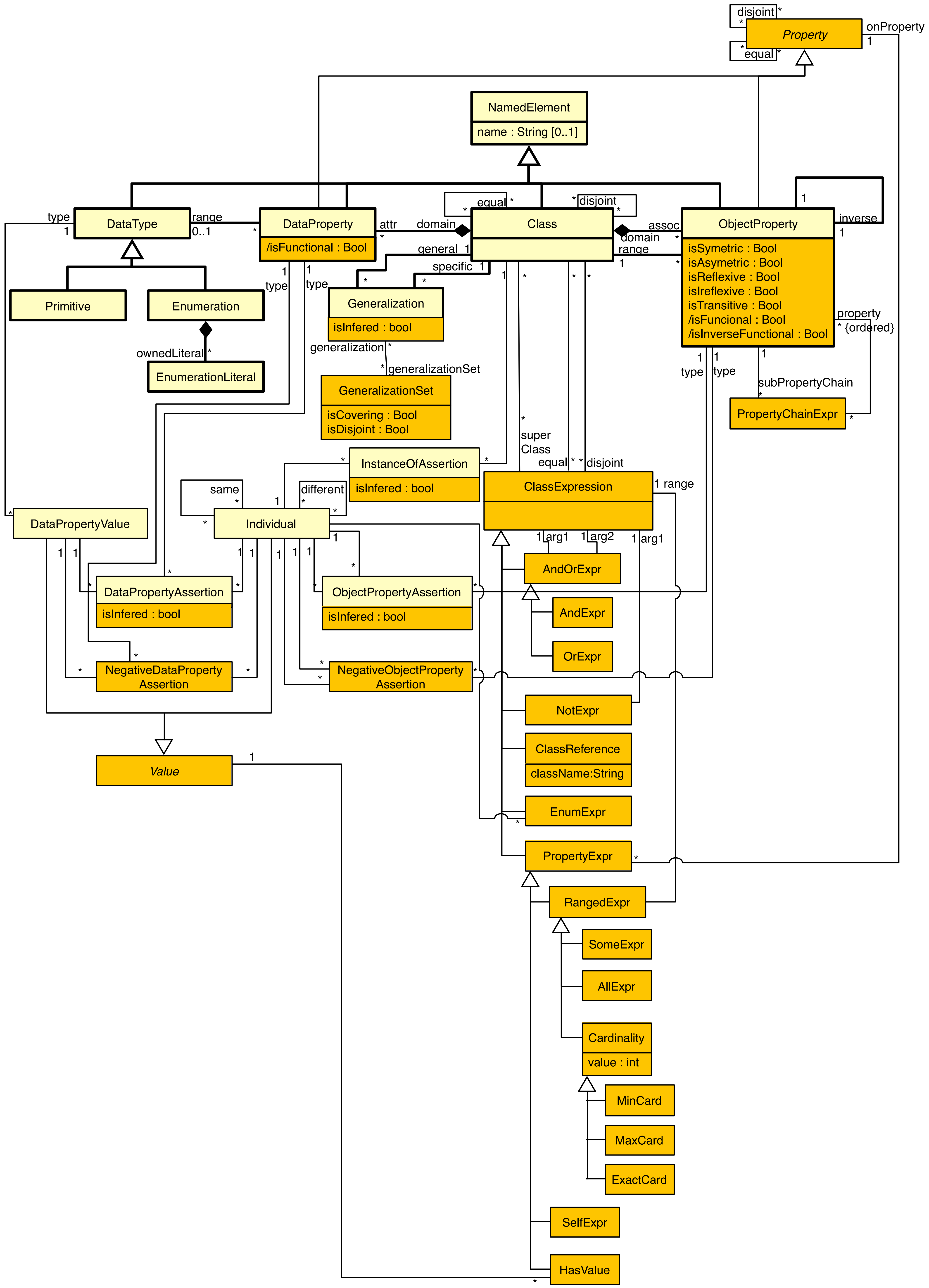
- UML based notation and metamodel for OWL
- Integration of IQuery with OWL
- Architecture

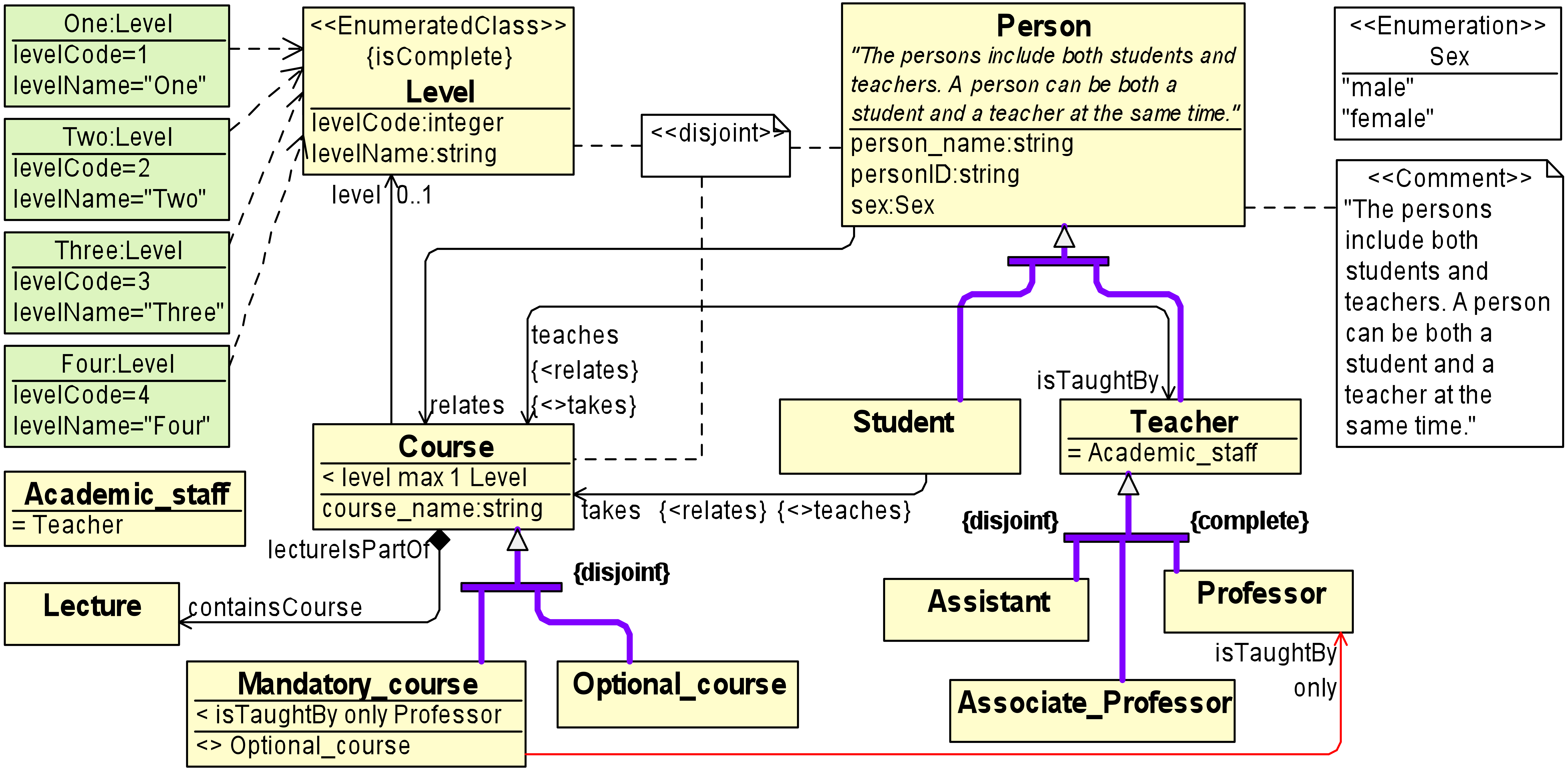
UML based notation and editor for OWL

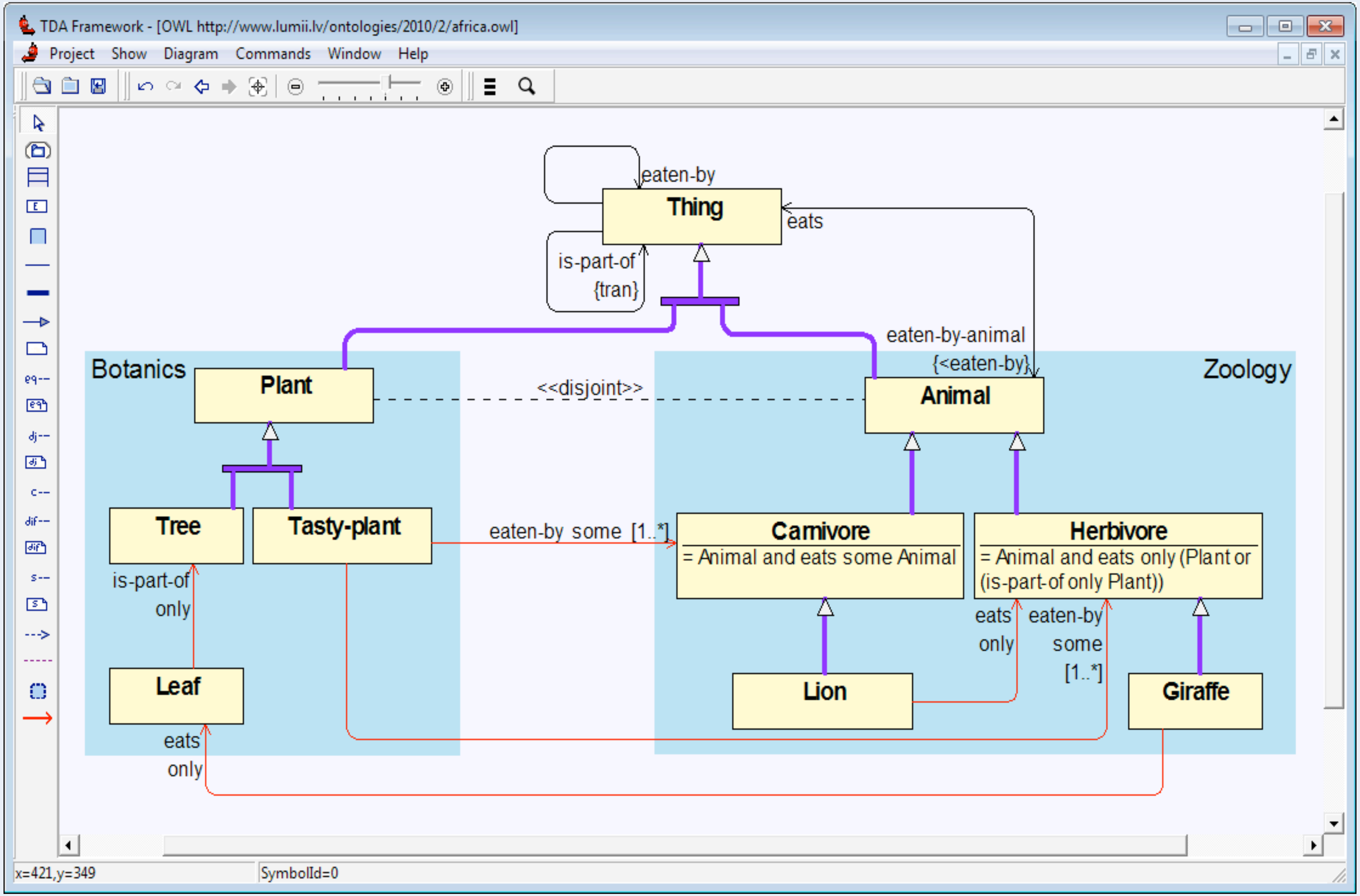
- Metamodel extension
- Notation
- Editor

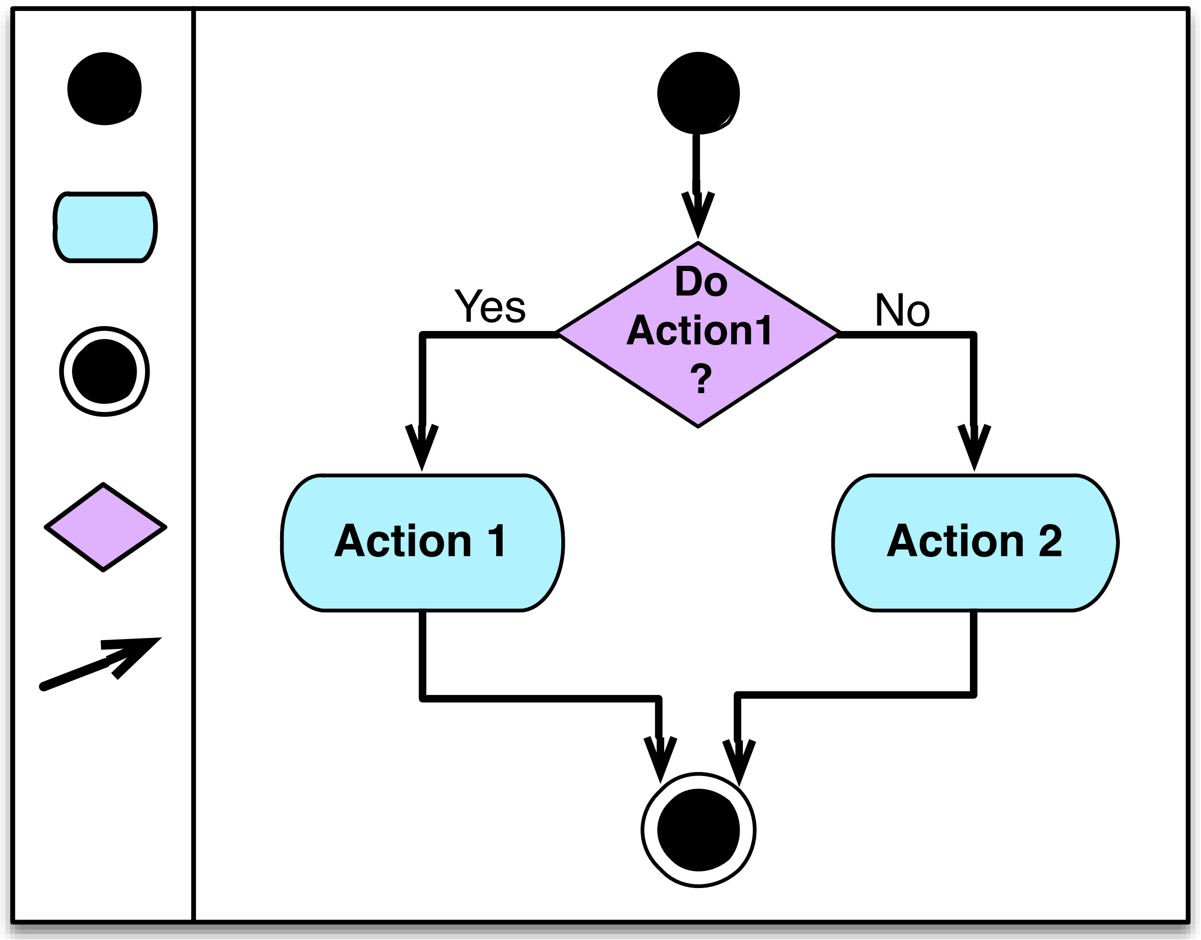

NamedElement
name : String [0..1]
DataType
type
1
range
0..1
*
DataProperty
attr
*
domain
Class
assoc
domain
*
range
1
*
ObjectProperty
1
inverse
1
general 1
specific 1
*
*
Generalization
1
type
1
1
type
Primitive
Enumeration
ownedLiteral *
EnumerationLiteral
*
InstanceOfAssertion
*
1
Individual
*
DataPropertyValue
1
1
1
1
*
*
*
*
*
DataPropertyAssertion
ObjectPropertyAssertion
*


NamedElement
name : String [0..1]
DataType
type
1
range
0..1
*
DataProperty
attr
*
domain
Class
assoc
domain
*
range
1
*
ObjectProperty
1
inverse
1
general
1
specific
1
*
*
Generalization
1
type
1
1
type
Primitive
Enumeration
ownedLiteral
*
EnumerationLiteral
*
InstanceOfAssertion
*
1
Individual
*
DataPropertyValue
1
*
DataPropertyAssertion
1
*
*
1
1
*
*
ObjectPropertyAssertion
*

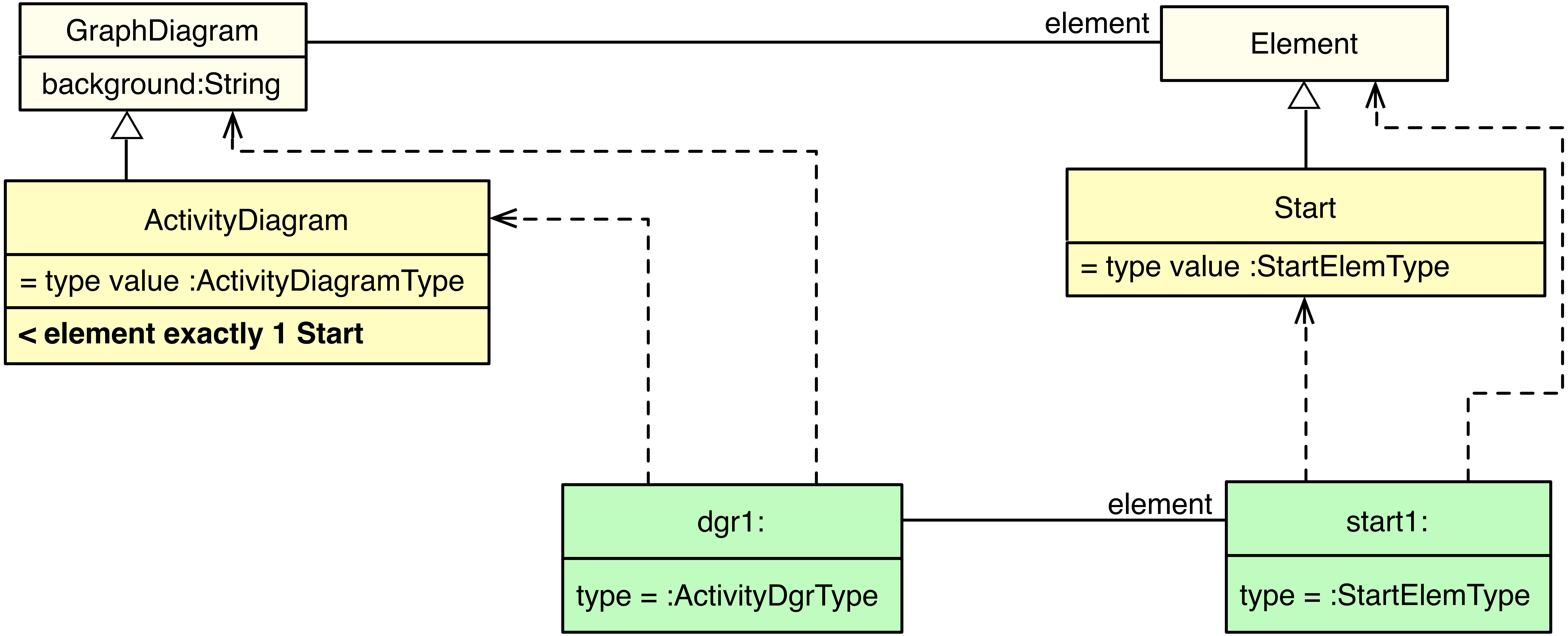

disjoint
Property
onProperty
equal
NamedElement
name : String [0..1]
DataType
type
range
0..1
DataProperty
/isFunctional : Bool
attr
domain
Class
equal
disjoint
ObjectProperty
isSymetric : Bool
isAsymetric : Bool
isReflexive : Bool
isIrreflexive : Bool
isTransitive : Bool
/isFuncional : Bool
/isInverseFunctional : Bool
assoc
domain
range
inverse
general
specific
Primitive
Enumeration
ownedLiteral
EnumerationLiteral
Generalization
isInfered : bool
generalization
generalizationSet
GeneralizationSet
isCovering : Bool
isDisjoint : Bool
property
{ordered}
subPropertyChain
PropertyChainExpr
InstanceOfAssertion
isInfered : bool
super
Class
ClassExpression
range
same
different
DataPropertyValue
Individual
arg1
arg2
AndOrExpr
AndExpr
OrExpr
DataPropertyAssertion
isInfered : bool
ObjectPropertyAssertion
isInfered : bool
NegativeDataProperty
Assertion
NegativeObjectProperty
Assertion
NotExpr
ClassReference
className:String
Value
EnumExpr
PropertyExpr
RangedExpr
SomeExpr
AllExpr
Cardinality
value : int
MinCard
MaxCard
ExactCard
SelfExpr
HasValue


One:Level
levelCode=1
levelName="One"
Two:Level
levelCode=2
levelName="Two"
Three:Level
levelCode=3
levelName="Three"
Four:Level
levelCode=4
levelName="Four"
Academic_staff
= Teacher
Lecture
<<EnumeratedClass>>
{isComplete}
Level
levelCode:integer
levelName:string
level
0..1
<<disjoint>>
Person
"The persons include both students and teachers. A person can be both a student and a teacher at the same time."
person_name:string
personID:string
sex:Sex
<<Enumeration>>
Sex
"male"
"female"
<<Comment>>
"The persons include both students and teachers. A person can be both a student and a teacher at the same time."
teaches
{<relates}
{<>takes}
relates
isTaughtBy
Course
< level max 1 Level
course_name:string
Student
Teacher
= Academic_staff
takes
{<relates}
{<>teaches}
{disjoint}
{complete}
lectureIsPartOf
{disjoint}
containsCourse
Assistant
Professor
Mandatory_course
< isTaughtBy only Professor
<> Optional_course
Optional_course
Associate_Professor
isTaughtBy
only


TDA Framework - [OWL http://www.lumii.lv/ontologies/2010/2/africa.owl]
Project
Show
Diagram
Commands
Window
Help
eaten-by
Thing
eats
is-part-of
{tran}
eaten-by-animal
{<eaten-by}
Botanics
Zoology
Plant
<<disjoint>>
Animal
Tree
Tasty-plant
eaten-by some [1..*]
Carnivore
= Animal and eats some Animal
Herbivore
= Animal and eats only (Plant or (is-part-of only Plant))
is-part-of
only
eats
only
eaten-by
some
[1..*]
Leaf
Lion
Giraffe
eats
only
x=421,y=349
SymbolId=0



Every **activity diagram** has exactly one **start** element.



GraphDiagram

background:String

element

Element

ActivityDiagram

= type value :ActivityDiagramType

< element exactly 1 Start

Start

= type value :StartElemType

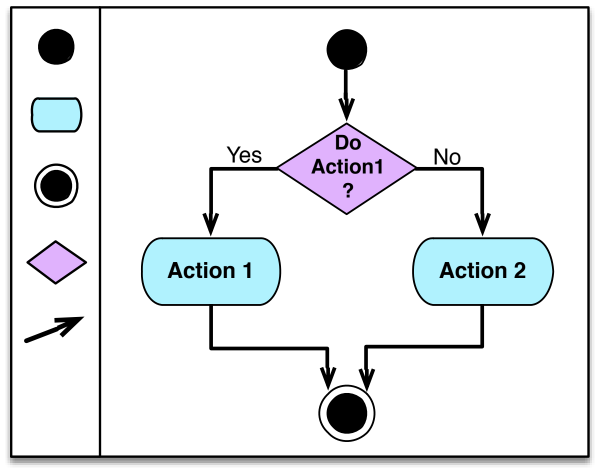
dgr1:

type = :ActivityDgrType

element

start1:

type = :StartElemType



Every **activity diagram** has exactly one **start** element.

What is needed

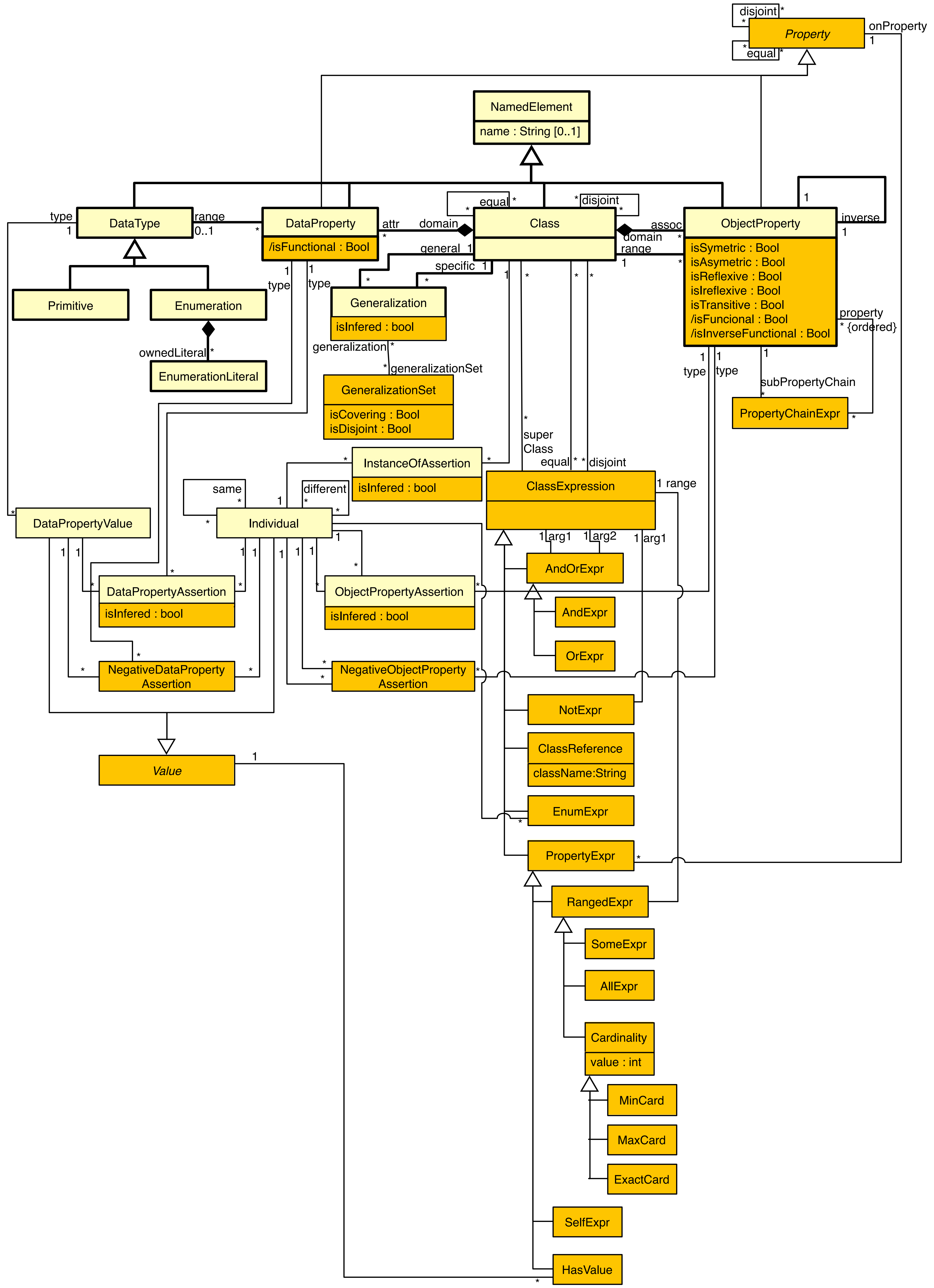
✓ UML based notation and metamodel for OWL

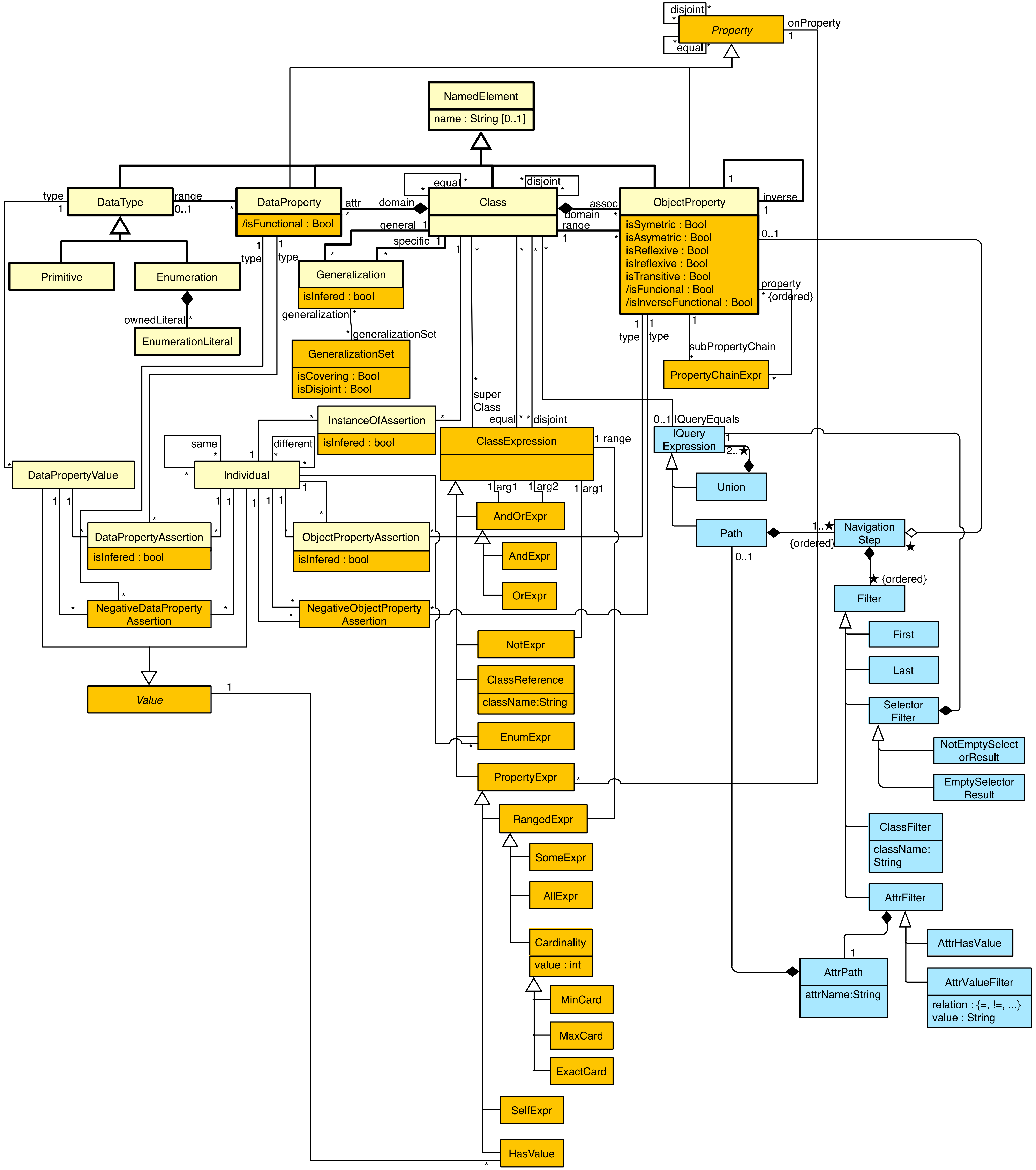
➡ **Integration of IQuery with OWL**

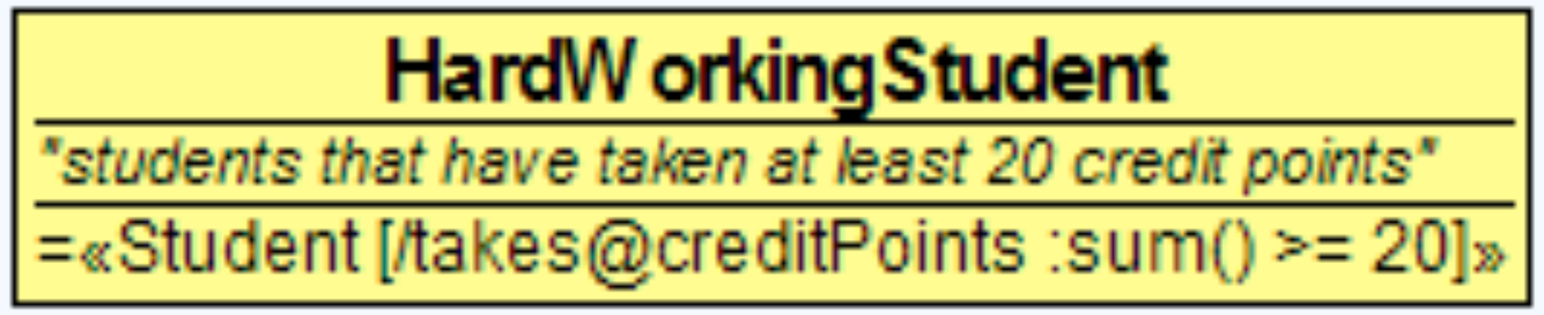
- Architecture

Extending OWL with lQuery selectors

- Metamodel extension
- Graphical notation extension
- Integration with semantic reasoners


Property
disjoint
equal
onProperty
NamedElement
name : String [0..1]
DataType
type
range
0..1
DataProperty
/isFunctional : Bool
attr
domain
equal
disjoint
Class
assoc
domain
range
general
specific
ObjectProperty
isSymetric : Bool
isAsymetric : Bool
isReflexive : Bool
isIreflexive : Bool
isTransitive : Bool
/isFuncional : Bool
/isInverseFunctional : Bool
inverse
property
{ordered}
subPropertyChain
PropertyChainExpr
Primitive
Enumeration
ownedLiteral
EnumerationLiteral
Generalization
isInfered : bool
generalization
generalizationSet
GeneralizationSet
isCovering : Bool
isDisjoint : Bool
InstanceOfAssertion
isInfered : bool
super
Class
equal
disjoint
ClassExpression
range
same
different
Individual
DataPropertyValue
DataPropertyAssertion
isInfered : bool
ObjectPropertyAssertion
isInfered : bool
arg1
arg2
arg1
AndOrExpr
AndExpr
OrExpr
NegativeDataProperty Assertion
NegativeObjectProperty Assertion
NotExpr
ClassReference
className:String
EnumExpr
Value
PropertyExpr
RangedExpr
SomeExpr
AllExpr
Cardinality
value : int
MinCard
MaxCard
ExactCard
SelfExpr
HasValue


disjoint
equal
Property
onProperty
NamedElement
name : String [0..1]
DataType
type
range
DataProperty
/isFunctional : Bool
attr
domain
equal
Class
disjoint
assoc
domain
range
ObjectProperty
isSymetric : Bool
isAsymetric : Bool
isReflexive : Bool
isIreflexive : Bool
isTransitive : Bool
/isFuncional : Bool
/isInverseFunctional : Bool
inverse
general
specific
Primitive
Enumeration
ownedLiteral
EnumerationLiteral
Generalization
isInfered : bool
generalization
generalizationSet
GeneralizationSet
isCovering : Bool
isDisjoint : Bool
property
{ordered}
subPropertyChain
PropertyChainExpr
type
super
Class
equal
disjoint
InstanceOfAssertion
isInfered : bool
same
different
DataPropertyValue
Individual
ClassExpression
range
0..1
IQueryEquals
IQuery
Expression
2..★
Union
arg1
arg2
AndOrExpr
AndExpr
OrExpr
DataPropertyAssertion
isInfered : bool
ObjectPropertyAssertion
isInfered : bool
Path
1..★
{ordered}
Navigation
Step
★ {ordered}
Filter
NegativeDataProperty
Assertion
NegativeObjectProperty
Assertion
NotExpr
ClassReference
className:String
EnumExpr
PropertyExpr
Value
First
Last
Selector
Filter
NotEmptySelect
orResult
EmptySelector
Result
RangedExpr
SomeExpr
AllExpr
Cardinality
value : int
MinCard
MaxCard
ExactCard
SelfExpr
HasValue
ClassFilter
className:
String
AttrFilter
AttrHasValue
AttrPath
attrName:String
AttrValueFilter
relation : {=, !=, ...}
value : String



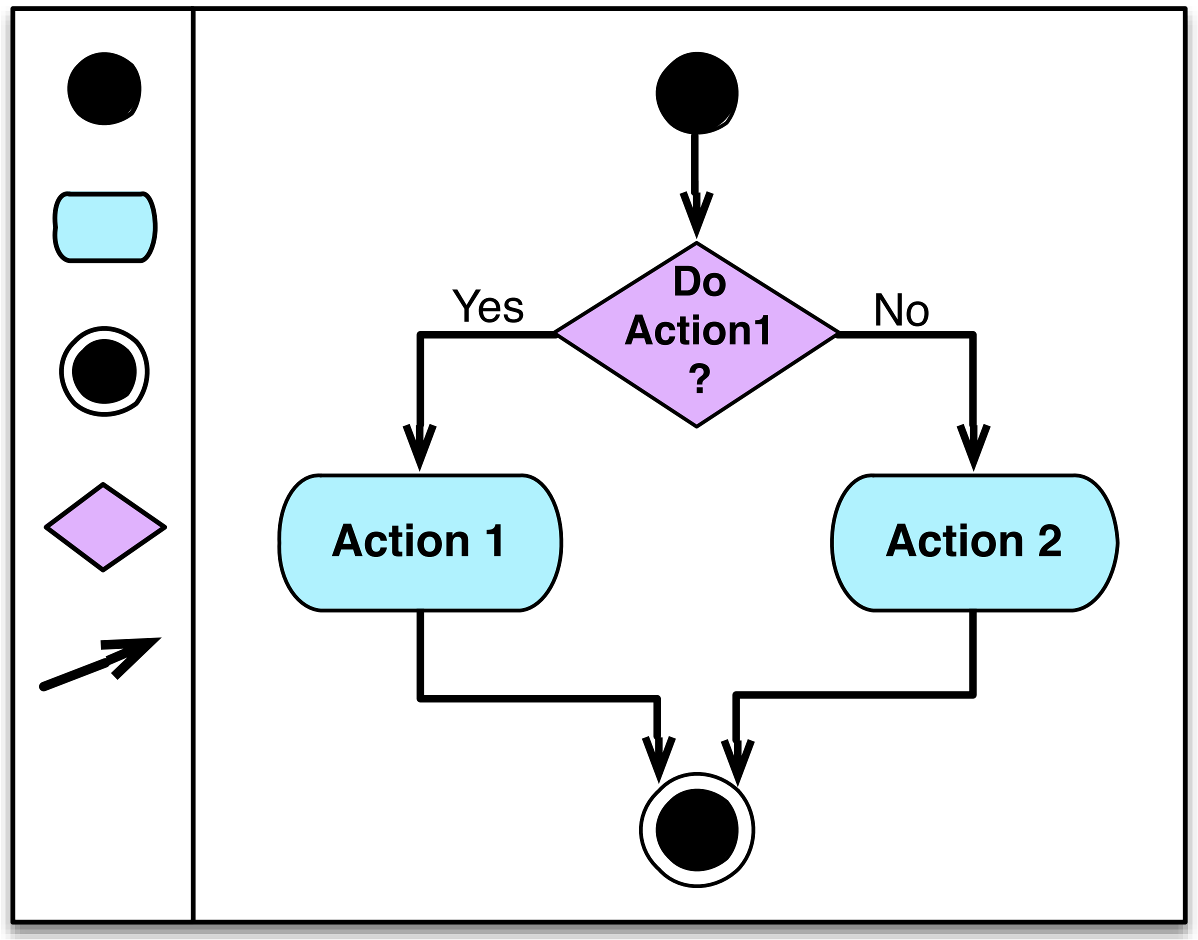
HardWorkingStudent

"students that have taken at least 20 credit points"

=«Student [/takes@creditPoints :sum() >= 20]»

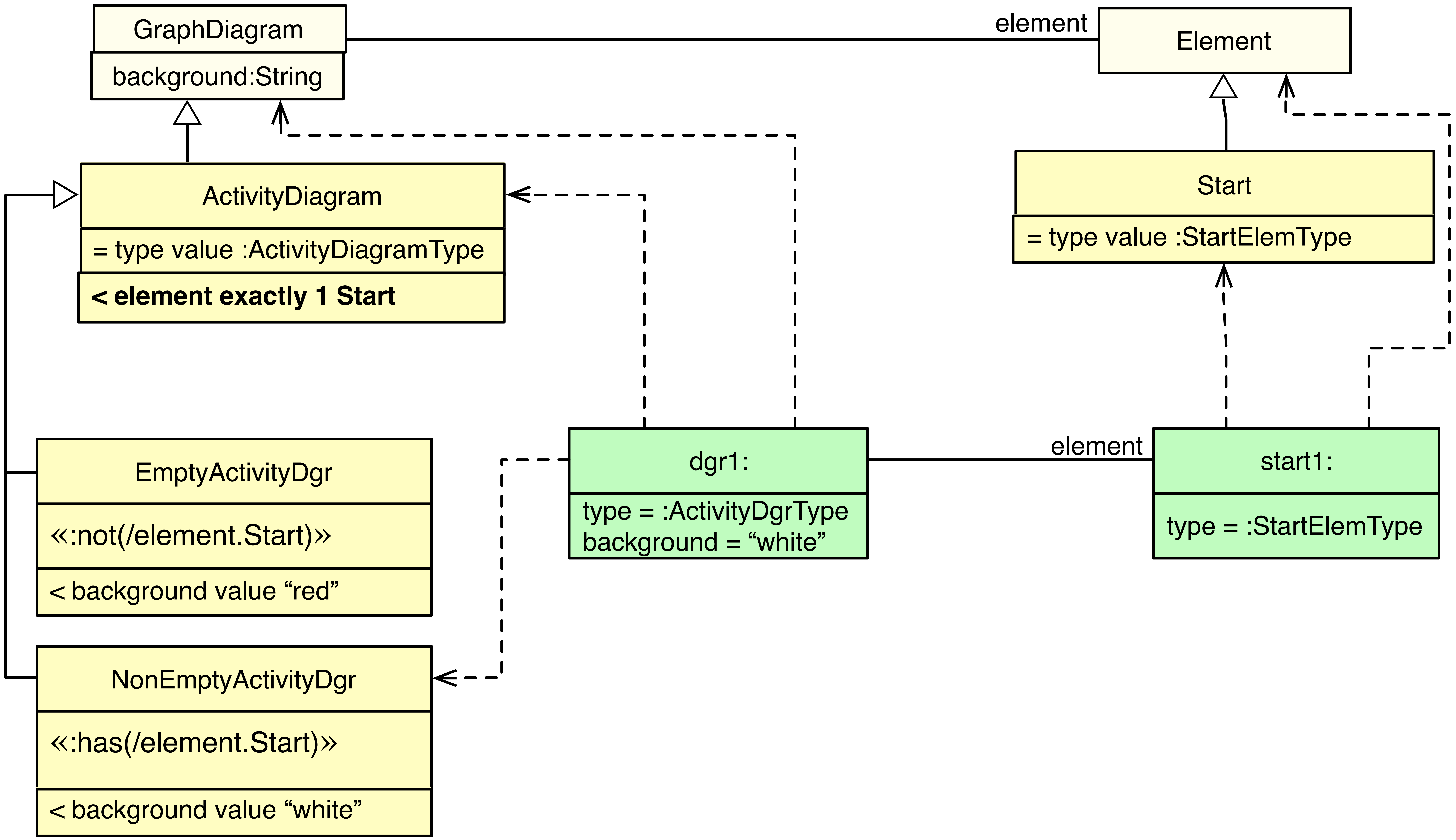
Integration Algorithm

```
1. Classify with OWL & materialize

2. Classify with lQuery & materialize

3. If new inferences
       goto step 1
   else
       finish
```



Activity diagram without a **start** element is **red**.

Activity diagram with a **start** element is **white**.



GraphDiagram

background:String

element

Element

ActivityDiagram

= type value :ActivityDiagramType

< element exactly 1 Start

Start

= type value :StartElemType

EmptyActivityDgr

«:not(/element.Start)»

< background value “red”

NonEmptyActivityDgr

«:has(/element.Start)»

< background value “white”

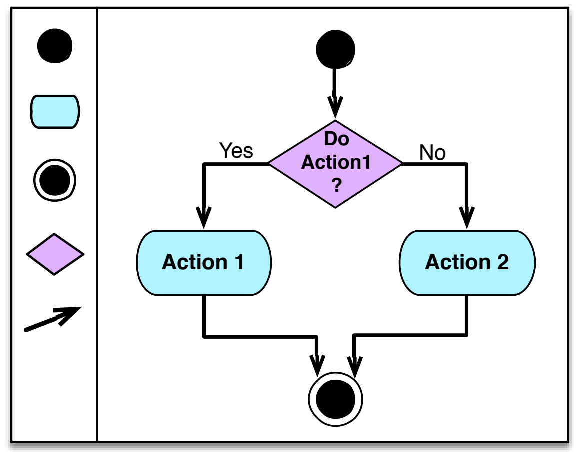
dgr1:

type = :ActivityDgrType
background = “white”

element

start1:

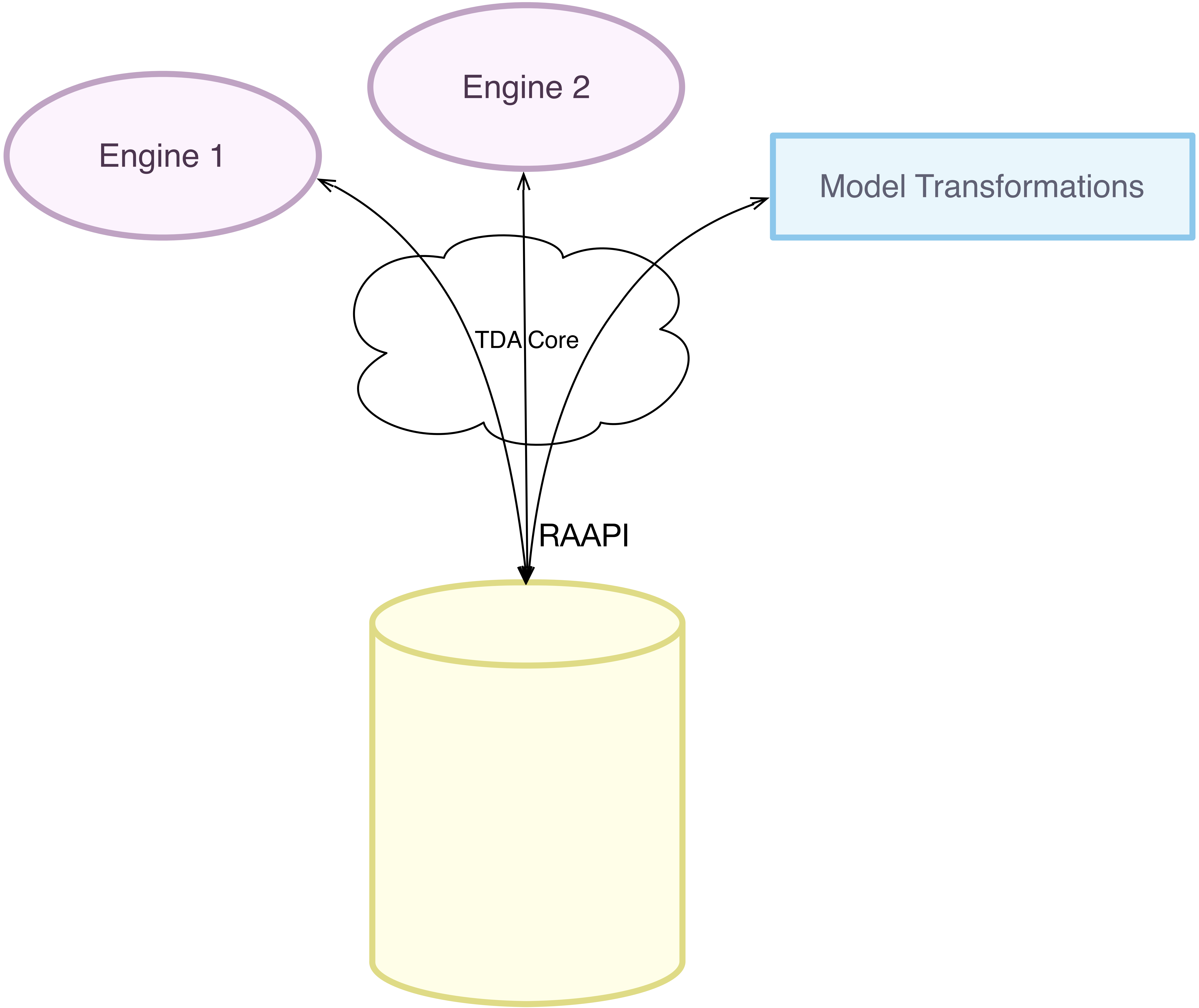
type = :StartElemType

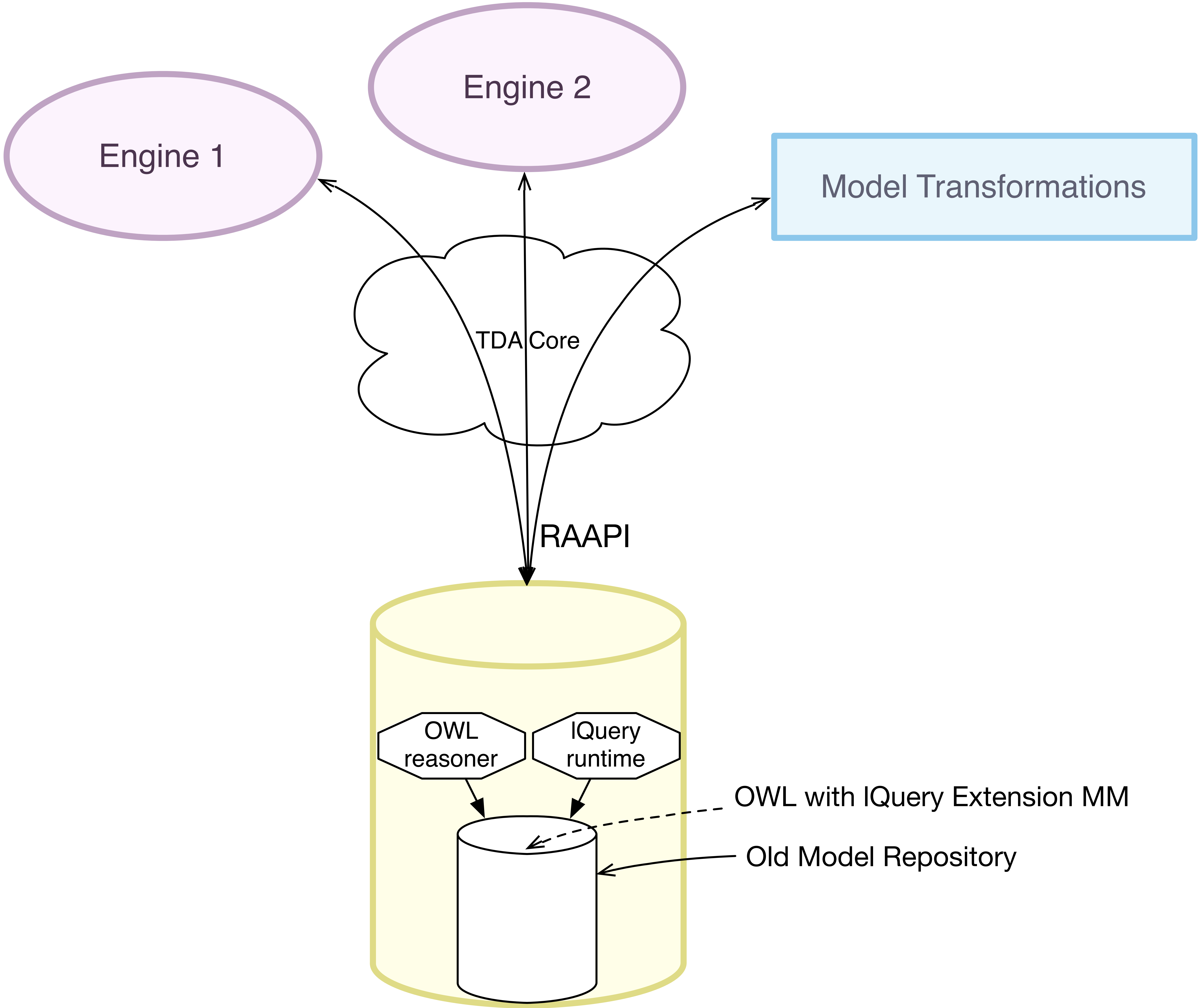


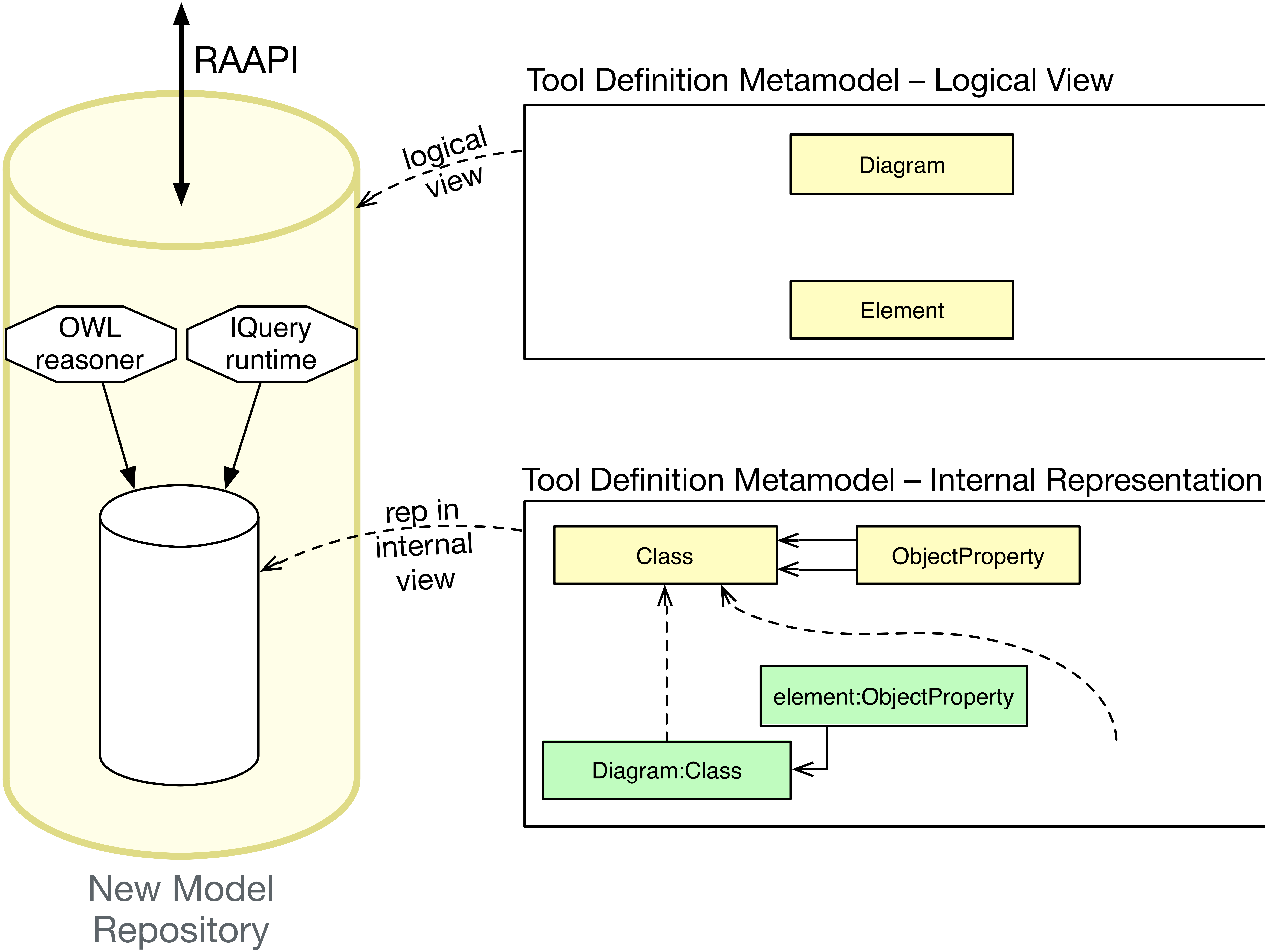
Activity diagram without a **start** element is **red**.
Activity diagram with a **start** element is **white**.

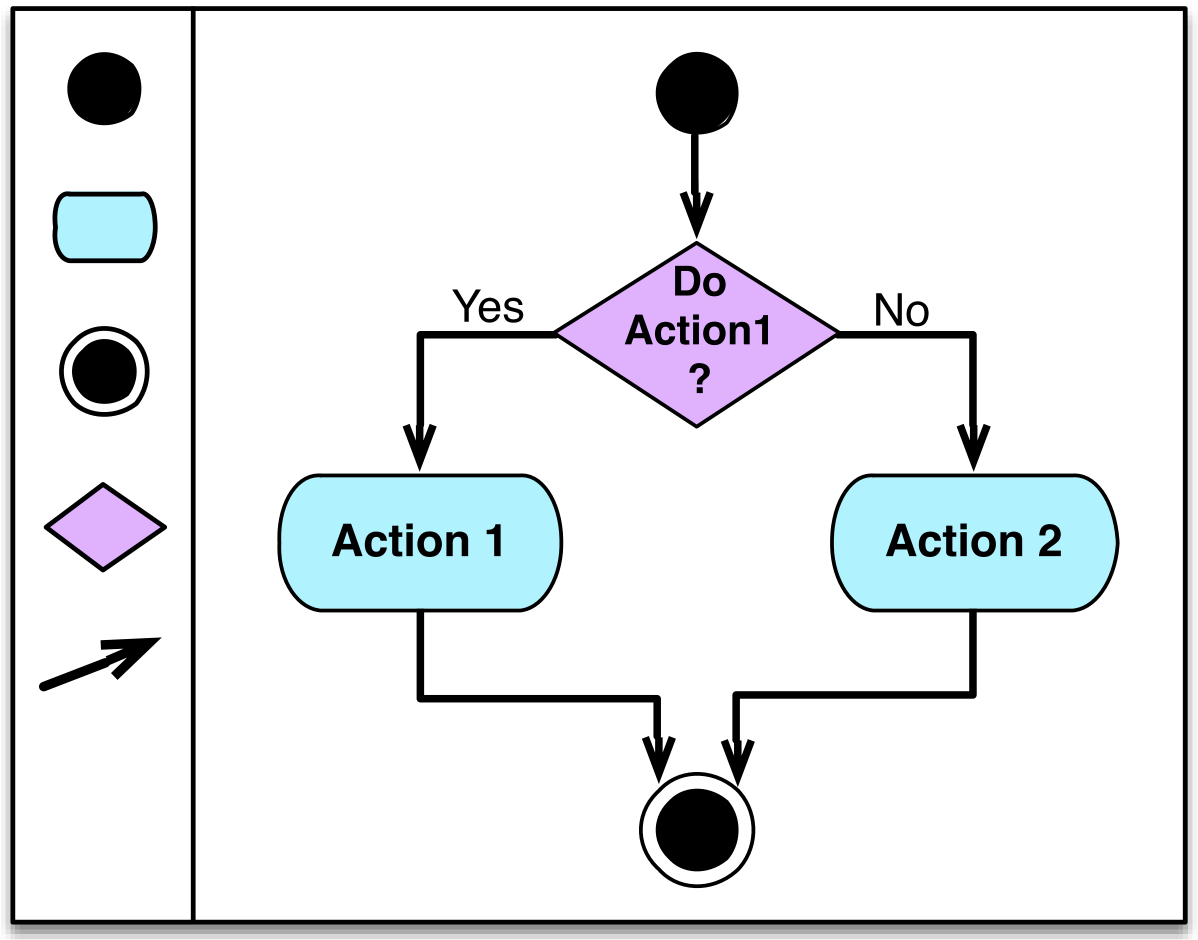
What is needed

- ✓ UML based notation and metamodel for OWL
- ✓ Integration of IQuery with OWL
- ➡ **Architecture**


Engine 2
Engine 1
Model Transformations
TDA Core
RAAPI


Engine 2
Engine 1
Model Transformations
TDA Core
RAAPI
OWL reasoner
IQuery runtime
OWL with IQuery Extension MM
Old Model Repository

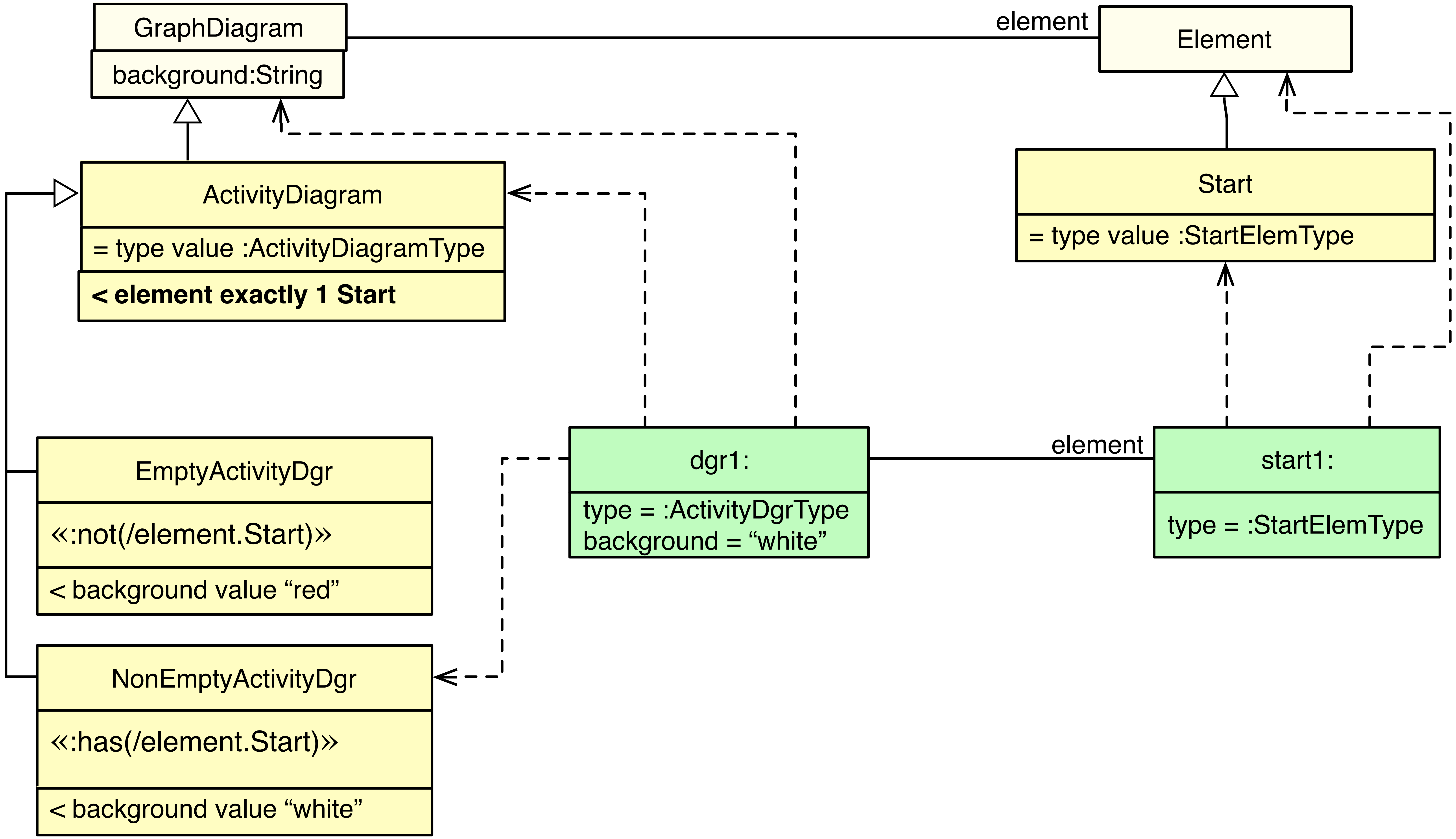

RAAPI
Tool Definition Metamodel – Logical View
logical view
Diagram
Element
OWL reasoner
IQuery runtime
Tool Definition Metamodel – Internal Representation
rep in internal view
Class
ObjectProperty
element:ObjectProperty
Diagram:Class
New Model Repository



Every **activity diagram** has exactly one **start** element.

Activity diagram without a **start** element is **red**.

Activity diagram with a **start** element is **white**.



GraphDiagram

background:String

element

Element

ActivityDiagram

= type value :ActivityDiagramType

< element exactly 1 Start

Start

= type value :StartElemType

EmptyActivityDgr

«:not(/element.Start)»

< background value “red”

NonEmptyActivityDgr

«:has(/element.Start)»

< background value “white”

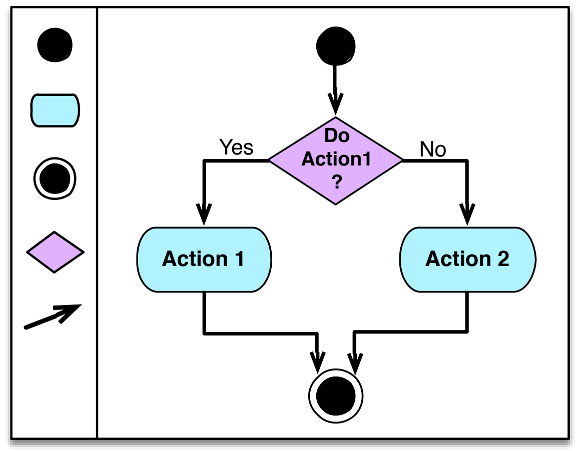
dgr1:

type = :ActivityDgrType
background = “white”

element

start1:

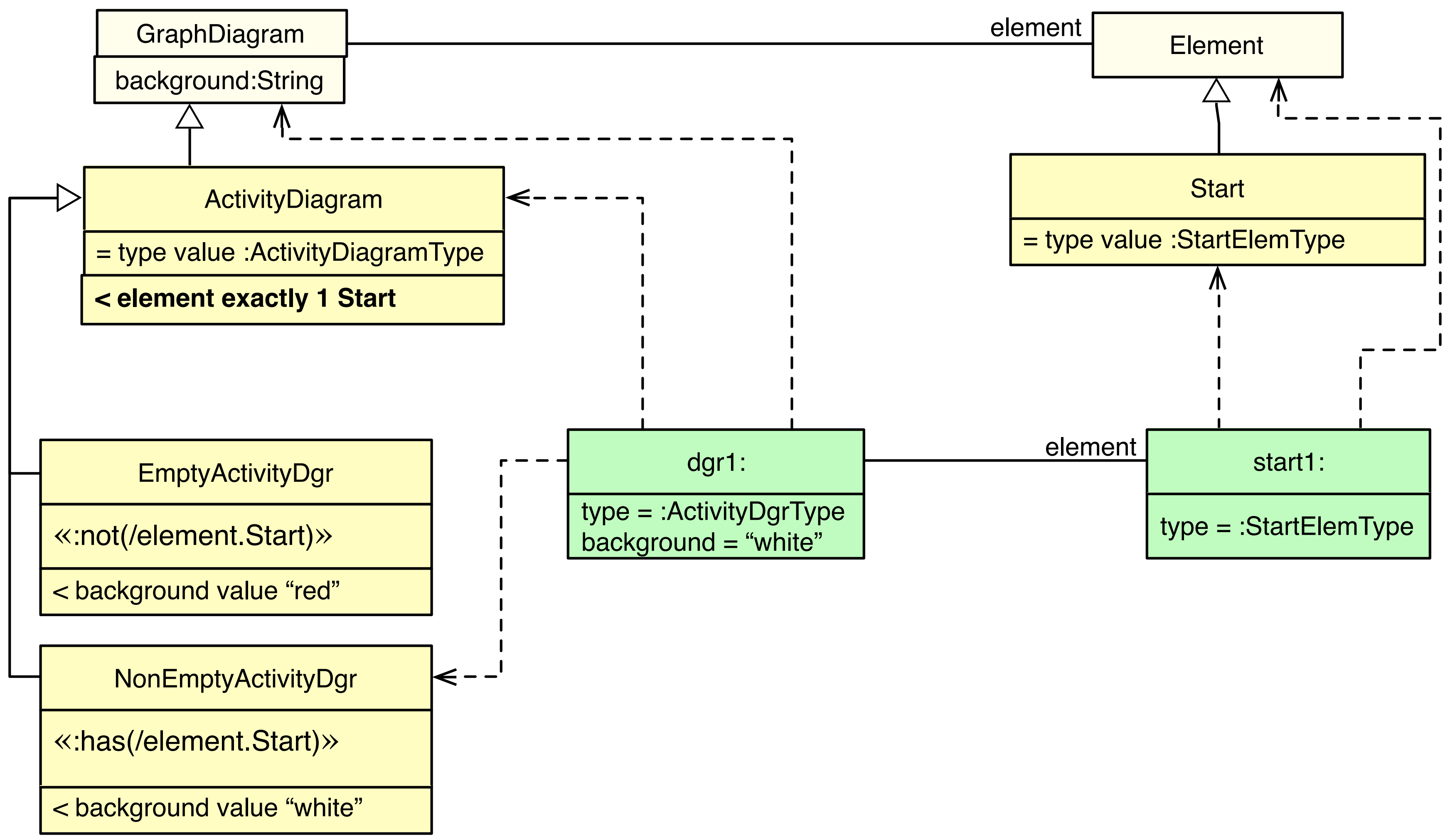
type = :StartElemType

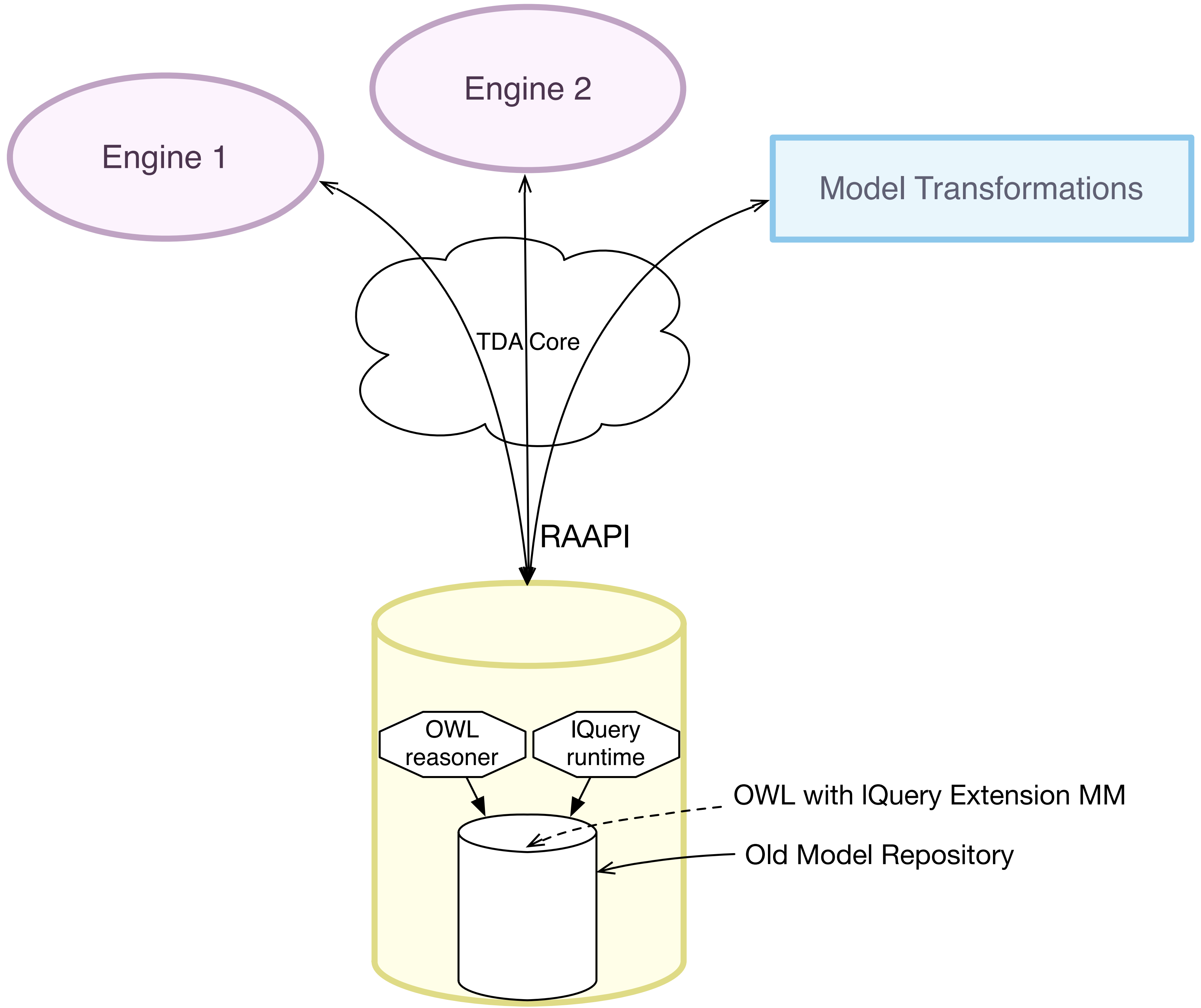


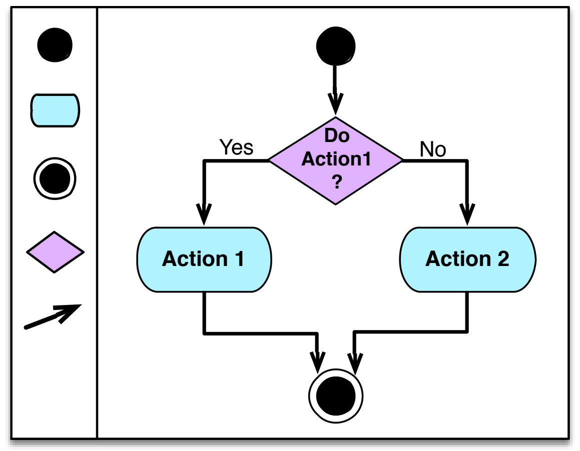
Every **activity diagram** has exactly one **start** element.

Activity diagram without a **start** element is **red**.

Activity diagram with a **start** element is **white**.



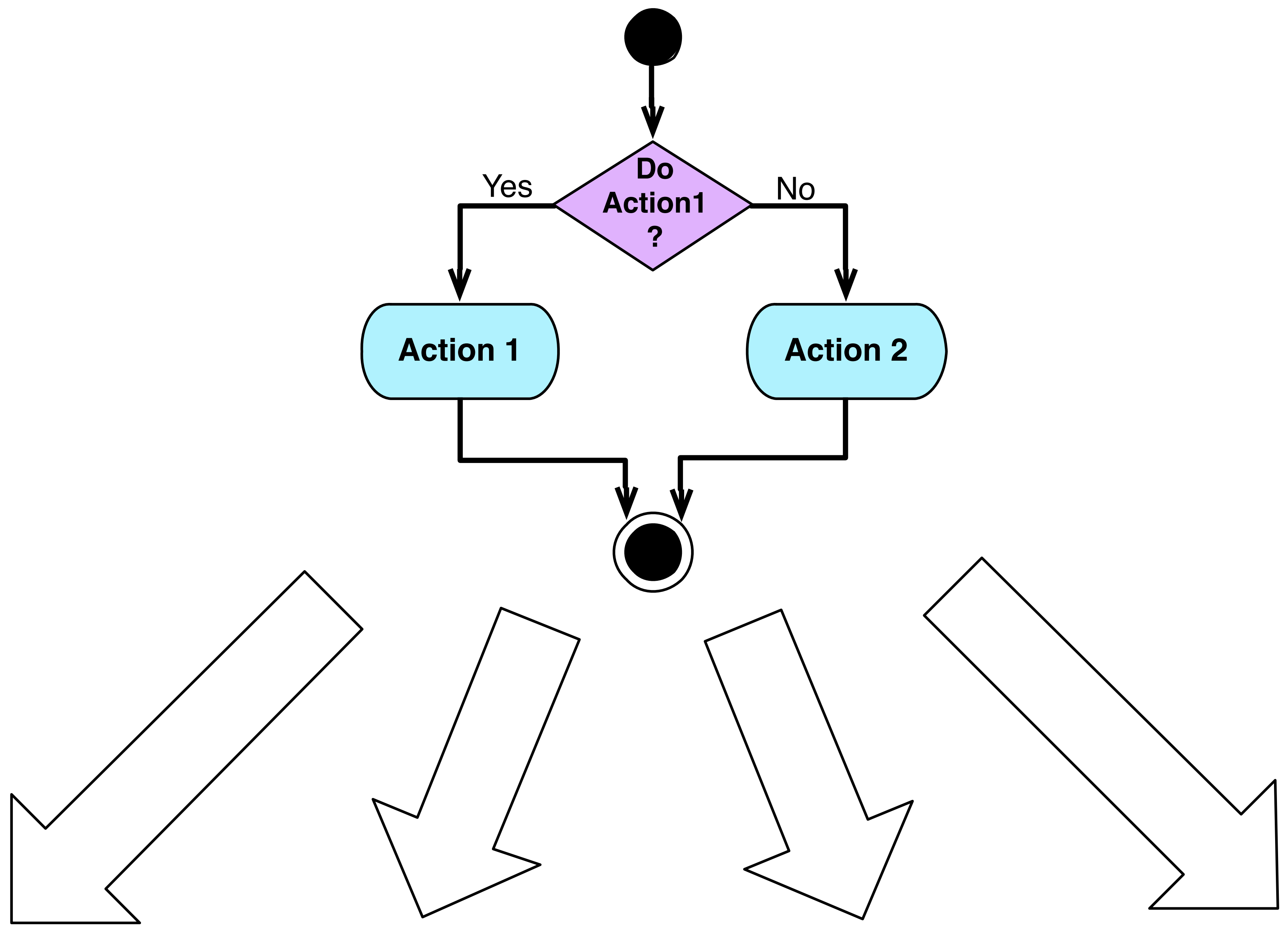




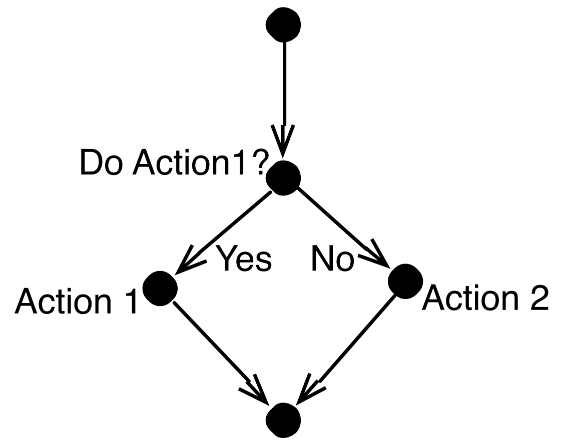
Every **activity diagram** has exactly one **start** element.

Activity diagram without a **start** element is **red**.

Activity diagram with a **start** element is **white**.



Graph



Types

FlowChart

Start

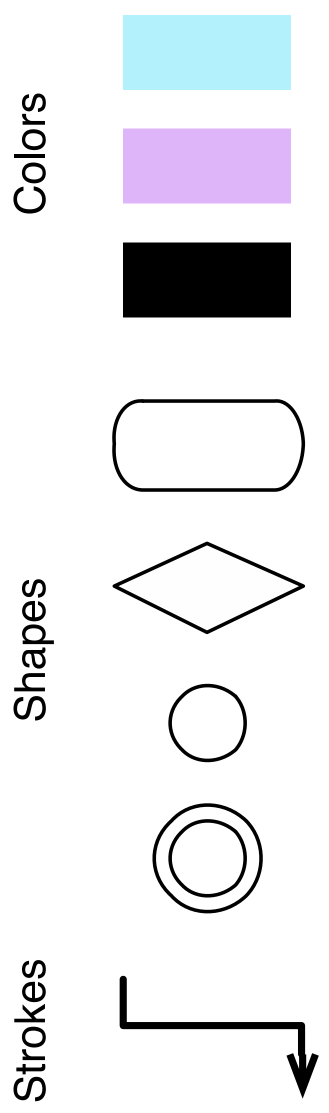
End

Decision

Action

Flow

Styles



Fonts

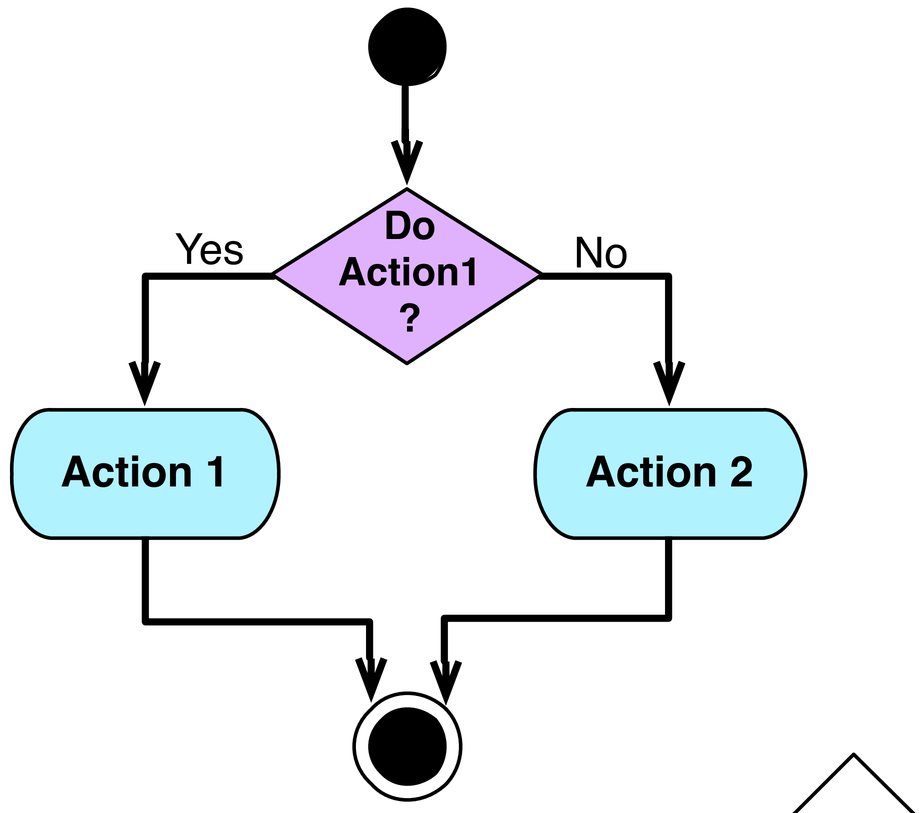
Helvetica Bold 12pt

Helvetica Regular 12pt

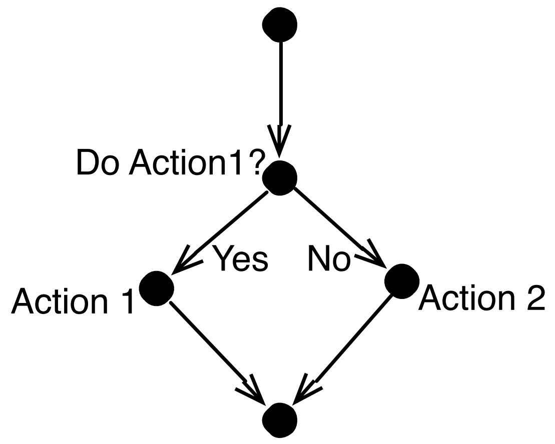
Behaviour

Universal

Domain Specific



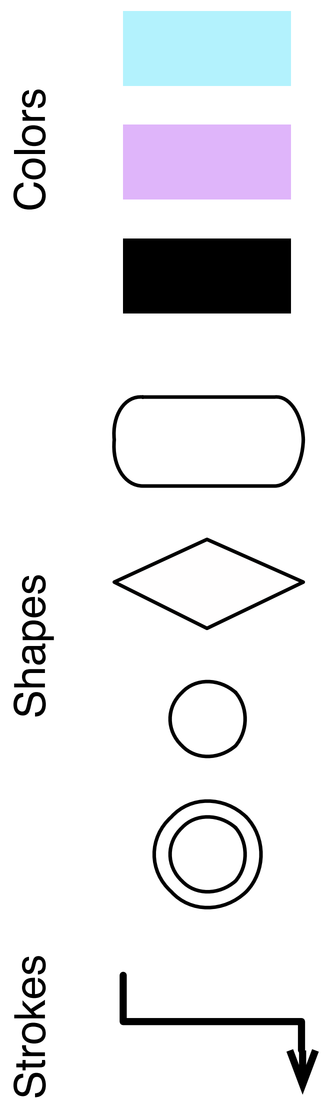
Graph



Types

```
FlowChart
Start
End
Decision
Action
Flow
```

Styles



Fonts

Helvetica Bold 12pt

Helvetica Regular 12pt

Behaviour

`Universal`

`Domain Specific`

Semantic Validation & Styles

`Ontology Based`

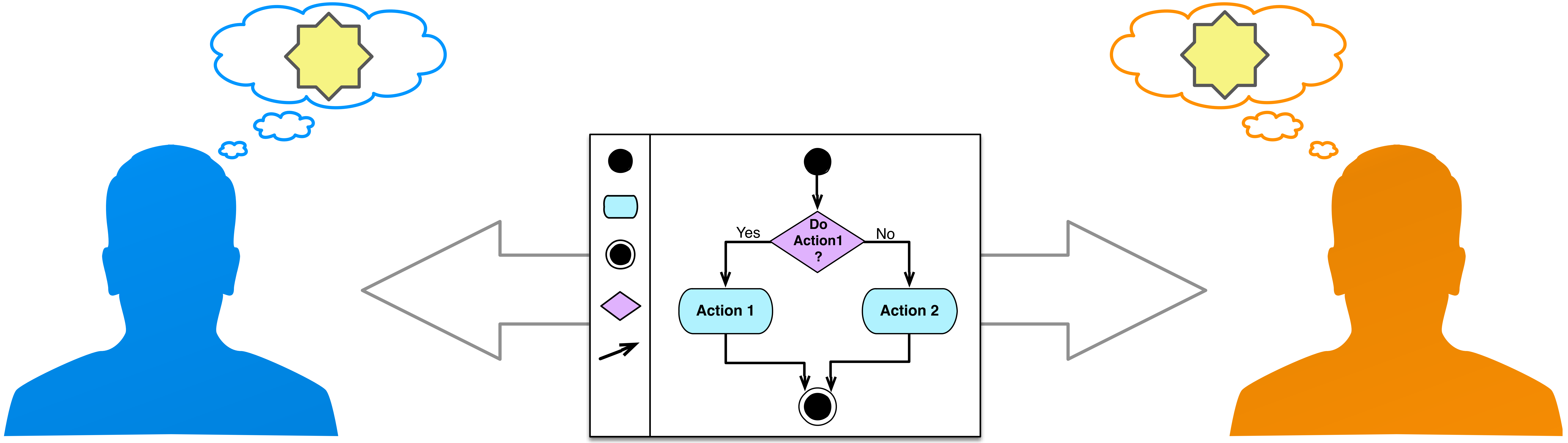
Summary

Summary

- Model Based Tool Building
    - Transformation Language: IQuery

Summary

- Model Based Tool Building
  - Transformation Language: lQuery
- Ontology Based Tool Building
  - UML based model and notation for OWL
  - OWL extension with lQuery selectors
  - Architecture


Do
Action1
?
Yes
No
Action 1
Action 2

Thank you!