

USE: UML Specification Environment

Copyright - laurent.thiry@uha.fr

Installation

```
wget https://freefr.dl.sourceforge.net/project/useocl/USE/6.0.x/use-6.0.0.tar.gz
tar xf use-6.0.0.tar.gz
cd use-6.0.0/bin
```

Diagramme des classes

```
cat > points.use <<EOF
model Figures

class Point
attributes
  x : Integer
  y : Integer
operations
  move(dx:Integer,dy:Integer)
end

class Figure
attributes
operations
  move(dx:Integer,dy:Integer)
end

association contains between
  Figure[1] role f
  Point[1..*] role ps
end
EOF

./use -nr points.use
> help
```

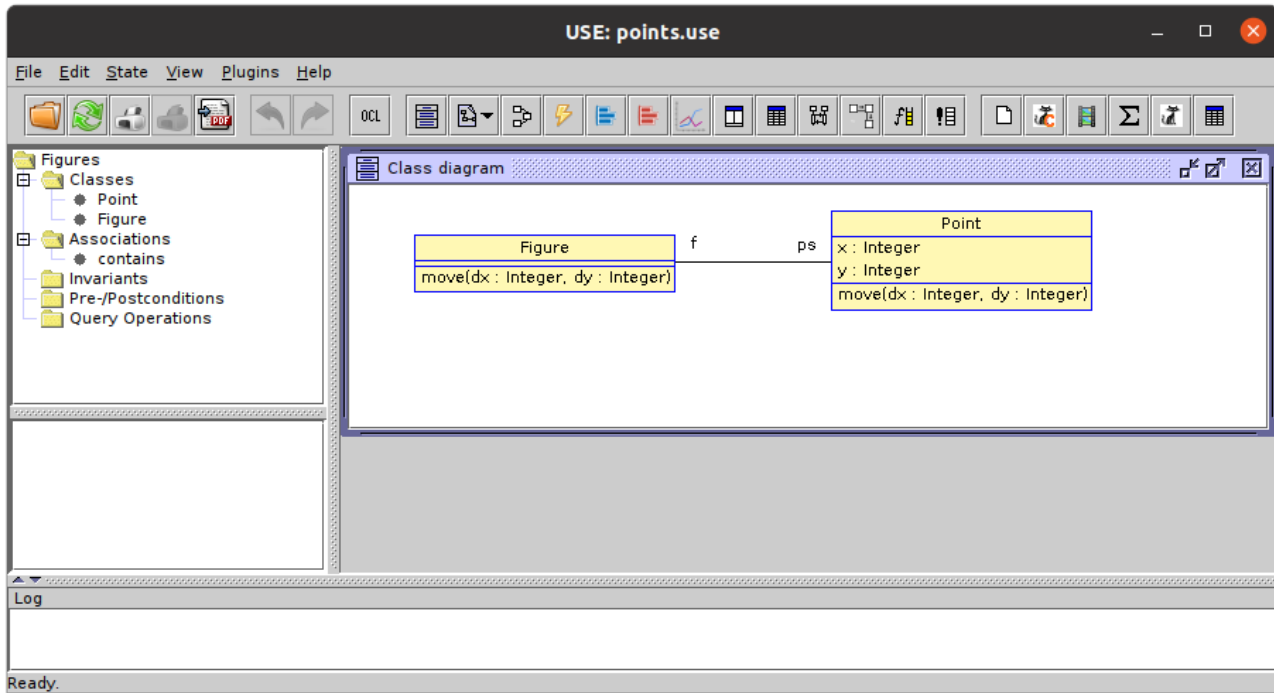
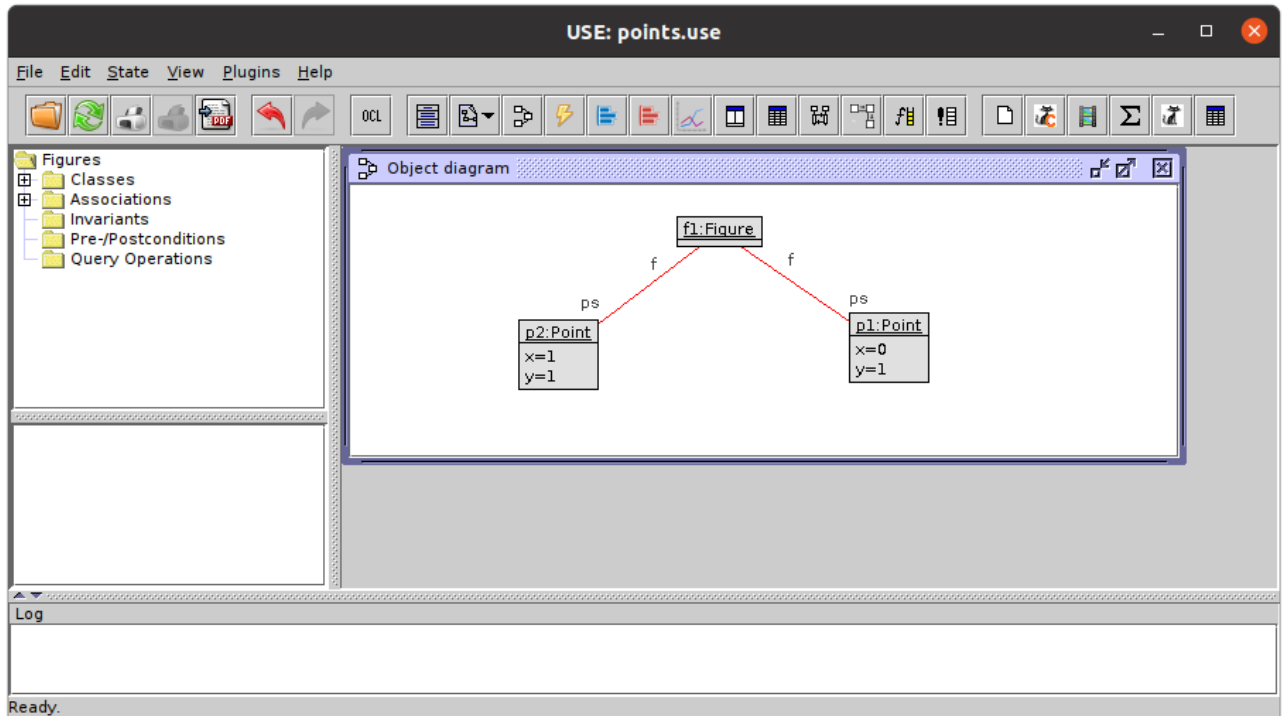


Diagramme d'objets

```

cat > points.cmd << EOF
!create p1 : Point
!set p1.x := 0
!set p1.y := 1
!create p2 : Point
!set p2.x := 1
!set p2.y := 1
!create f1 : Figure
!insert (f1,p1) into contains
!insert (f1,p2) into contains
EOF
> open points.cmd

```



Diagrammes de communication

```

./use -nr points.use
> help
> open points.cmd
> !openter f1 move(1,0)
> !openter p1 move(1,0)
> !opexit
> !openter p2 move(1,0)
> !opexit
> !opexit
> !opexit
> exit

```

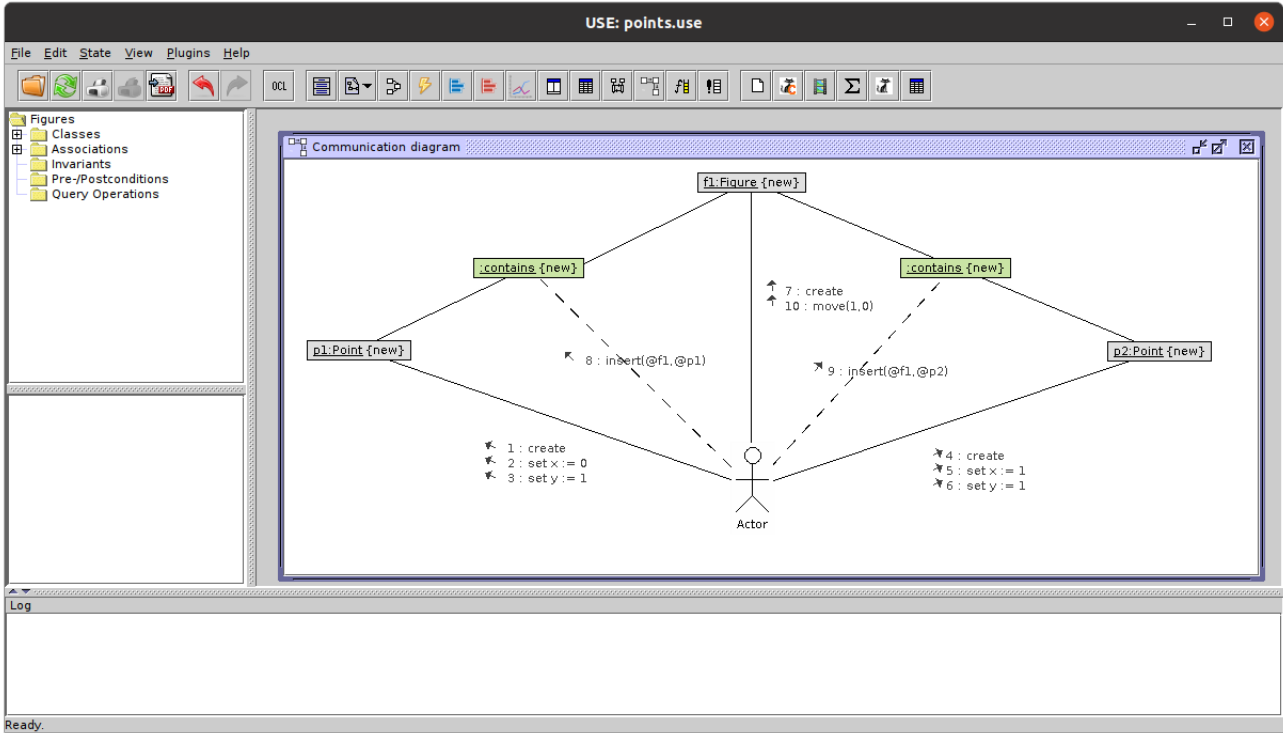
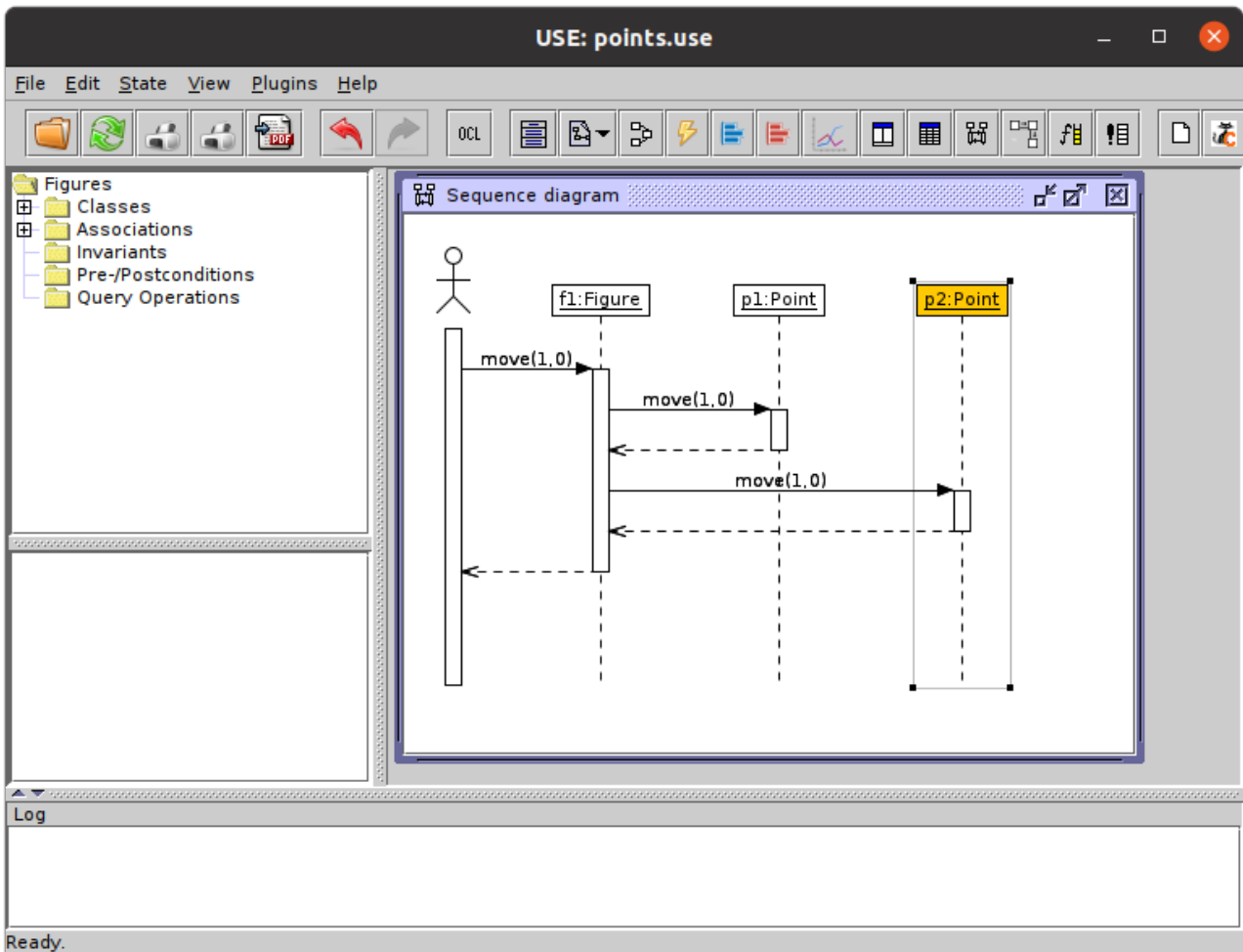
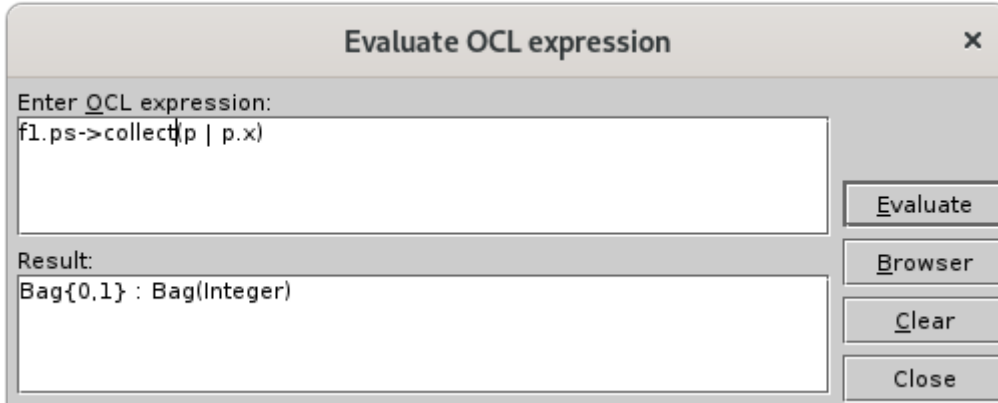


Diagramme de séquence



(Contraintes) OCL

```
f1
f1.ps
f1.ps.collect(p|p.x)
```



```
cat >> points.use <<EOF
constraints
context Point
  inv: (self.x >= 0) and (self.x <=5)
context Point::move(dx:Integer,dy:Integer)
  post: (x - x@pre = dx) and (y -y@pre = dy)
EOF
```

Class invariants

| Invariant | Satisfied |
|-------------|-----------|
| Point::inv1 | false |

1 cnstr. failed. Inherent cnstrs. OK. (1ms) 100 %

Object properties

p2

| Attribute | Value |
|-------------|-------|
| x : Integer | 7 |
| y : Integer | 1 |

Apply Reset

Evaluation browser

context self : Point inv inv1:
((self.x >= 0) and (self.x <= 5))

- Point.allInstances()->forAll(self:Point | ((self.x >= 0) and (self.x <= 5))) = false
 - Point.allInstances() = Set{p1,p2}
 - ((self.x >= 0) and (self.x <= 5)) = true
 - ((self.x >= 0) and (self.x <= 5)) = false
 - (self.x >= 0) = true
 - (self.x <= 5) = false
 - self.x = 7
 - self = @p2

Display options Close

Inheritance, abstraction and polymorphism

```
cat > figures.use <<EOF
model Figures
abstract class Element
  operations
  move(dx:Integer,dy:Integer)
end

class Point < Element
  attributes
  x: Integer
  y:Integer
end

class Figure < Element
  attributes
  ps : Set(Element)
end
EOF
```

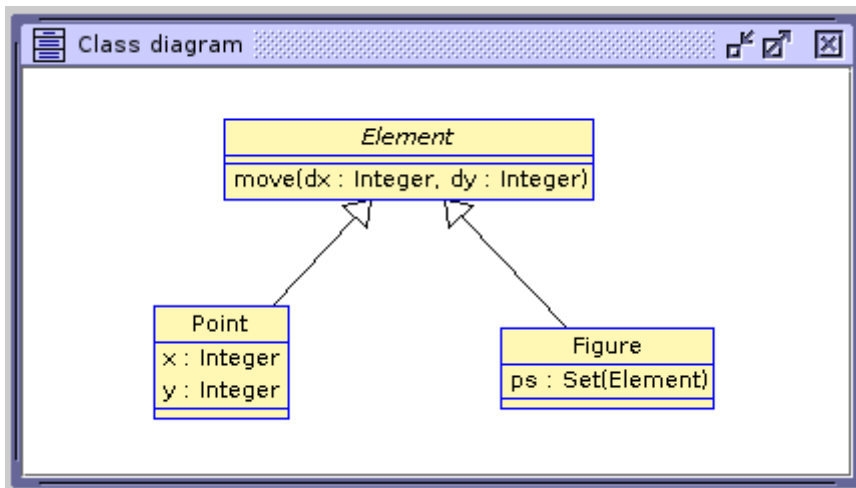


Diagramme d'états

```
cat > light.use <<EOF
model SM
class Light
  operations
    click()
  statemachines
    psm Behavior
  states
    start: initial
    off
    on
  transitions
    start -> off
    off -> on { click }
    on -> off { click }
end
end
EOF
```

