

- Specification of Rectangle with Uncertain values (UReal) using UML and OCL models
- (file: "Rectangle.ocl")

Rectangle
+x : UReal +y : UReal +h : UReal +w : UReal
+area() : UReal +fitsIn(base : Rectangle) : Boolean +move(deltaH : UReal, deltaW : UReal)

context Rectangle:: area() :UReal
post: **result** = (self.h).mult(self.w)

context Rectangle::fitsIn(base :Rectangle) :**Boolean**
post: **result** = base.x.lessEq(self.x) and
self.x.add(self.w).lessEq(base.x.add(base.w)) and
base.y.lessEq(self.y) and
self.y.add(self.h).lessEq(base.y.add(base.h))

context Rectangle:: move (deltaH :UReal, deltaW :UReal) : Rectangle
post: result.x = self.x.add(deltaW) **and**
result.y = self.y.add(deltaH) **and**
result.h = self.h **and**
result.w = self.w

-- Specification of Rectangle with Uncertain values (UReal) using OCL/USE (SOIL) models.
-- (file: Rectangle.use)

```
model Rectangle
class Rectangle
attributes
  x: UReal -- position X of left-down corner
  y: UReal -- position Y of left-down corner
  h: UReal -- height
  w: UReal -- width
operations
  area() :UReal
  begin
    result:= (self.h).mult(self.w);
  end

  fitsIn(base :Rectangle) :Boolean
  begin
    declare XW : UReal, BW : UReal, YH : UReal, BH : UReal;
    XW:=new UReal;
    BW:=new UReal;
    YH:=new UReal;
    BH:=new UReal;

    XW :=(self.x).add(self.w);
    BW :=(base.x).add(base.w);
    YH :=(self.y).add(self.h);
    BH :=(base.y).add(base.h);
    result := (base.x).lessEq(self.x) and
              XW.lessEq(BW) and
              (base.y).lessEq(self.y) and
              YH.lessEq(BH);
  end

  move (deltaH :UReal, deltaW :UReal) : Rectangle
  begin
    declare aux : Rectangle;
    aux := new Rectangle;
    aux.x := self.x.add(deltaW);
    aux.y := self.y.add(deltaH);
    aux.h := self.h;
    aux.w := self.w;
    result := aux;
  end
end -- class Rectangle
class UReal
...
end
end --model
```

-- Example of model simulation, using USE. (file: "Rectangle-Execution.use")

```
!new UReal('x1')
!new UReal('y1')
!new UReal('w1')
!new UReal('h1')
!new UReal('dh')
!new UReal('dw')
!new UReal('arear')
!new UReal('areas')
!new UReal('areat')
!new Boolean ("b")
!new Rectangle('r')
!new Rectangle('s')
!new Rectangle('t')

!x1.x :=0.0; -- auxiliary variable inicializacion
!x1.u :=0.0001;
!y1:=x1;
!w1.x:=10.0;
!w1.u :=0.0001;
!h1.x :=20.0;
!h1.u :=0.0001;

!r.x :=x1; -- first rectangle "r"
!r.y :=y1;
!r.w :=w1;
!r.h :=h1;

!s.x :=x1; -- second rectangle "s"
!s.y :=y1;
!s.w :=w1.add(w1);
!s.h :=h1;

!arear:=r.area(); -- area of r
?arear.x
?arear.u

!areas:=s.area(); -- area of s
?areas.x
?areas.u

!b:=arear.equals(arear); -- same areas?
?b

!b:=r.fitsIn(s); -- fitsIn?
?b
!b:=s.fitsIn(r);
?b
```

```

-- Let's move "r"
-- first, define the deltas
!dw.x := 0.1;
!dw.u := 0.0001;
!dh.x := 0.1;
!dh.u := 0.0001;

!t:=r.move(dw,dh); -- move r (and get "t")
!b:=t.fitsIn(s); -- does it fit anymore?
?b
!areat:=t.area(); -- area of t
?areat.x
?areat.u
!b:=areat.equals(arear); -- has area changed?
?b
--let's see the moved rectangle
!x1:=t.x;
!y1:=t.y;
!h1:=t.h;
!w1:=t.w;
?x1.x
?x1.u
?y1.x
?y1.u
?w1.x
?w1.u
?h1.x
?h1.u

```

NOTES:

- Auxiliary variables are needed because the current implementation of SOIL does not permit the concatenation of operations, i.e. `r.x.u` or `a.add(b).mult(c)`
- To simulate the model with USE, open first the `Rectangle.use` specification (with, e.g. the USE graphical interface) to load the model, and then open the `Rectangle-Execution.use` file using the USE command line (separate window).