Levon	Peters book	Peters book –modified

type RectangularDomain
  extends Cartesian2D (boundary ={left, bottom, right, top});
  parameter Point p;
  parameter Real w;
  parameter Real h;
  parameter Point p2 = Point(p.x+w, p.y);
  parameter Point p3 = Point(p.x+w, p.y+h);
  parameter Point p4 = Point(p, p.y+h);
  parameter Line bottom = Line(p, p2);
  parameter Line right = Line(p2, p3);
  parameter Line top = Line(p3, p4);
  parameter Line left = Line(p4, p);
end RectangularDomain;
	class DomainRectangle2D
	  extends Domain;
	  parameter Real Lx;
	  parameter Real Ly;
	  parameter Real ax;
	  parameter Real ay;
	  function shapeFunc
	  input Real v1, v2;
	  output Real x = ax + Lx * v1, y = ay + Ly * v2;
	  end shapeFunc;
	  Coordinate x (name = "cartesian");
	  Coordinate y (name = "cartesian");
	  Region2D interior(shape = shapeFunc, interval = {{0,1},{0,1}});
	  Region1D right(shape = shapeFunc, interval = {1,{0,1}});
	  Region1D bottom(shape = shapeFunc, interval = {{0,1},0});
	  Region1D left(shape = shapeFunc, interval = {0,{0,1}});
	  Region1D top(shape = shapeFunc, interval = {{0,1},1});
	end DomainRectangle2D;
		class DomainRectangle2D
		  extends Domain;
		  Coordinate x (name = "cartesian");
		  Coordinate y (name = "cartesian");
		  parameter Real a1; //x-coordinate of left side
		  parameter Real a2; //y-coorinate of lower side
		  parameter Real b1; //x-coordinate of right side
		  parameter Real b2; //y-coorinate of upper side
		  Region2D interior (x in {a1,b1}, y in {a2,b2}); //or rather (x,y) in {a1,b1}@{a2,b2}
		  Region1D right (x = a, y in {a2,b2});
		  Region1D bottom (x in {a1,b1}, y = b1);
		  Region1D left (x = a1, y = {a2,b2});
		  Region1D top (x in {a1,b1}, y = b2);
		end DomainRectangle2D;

type CircularDomain
  extends Cartesian2D(boundary = circle);
  parameter Point center;
  parameter Real radius;
  parameter Circle circle (c = center, r = radius);
end CircularDomain
	class DomainCircular2D
	  extends Domain;
	  parameter Real radius;
	  parameter Real cx = 0;
	  parameter Real cy = 0;
	  function shapeFunc
	  input Real r,v;
	  output Real x,y;
	  algorithm
	  x:=cx + radius * r * cos(2 * C.pi * v)
	  y:=cy + radius * r * sin(2 * C.pi * v);
	  end shapeFunc;
	  Coordinate x (name="cartesian");
	  Coordinate y (name="cartesian";
	  Region2D interior(shape = shapeFunc, interval = {{0,1},{0,1}});
	  Region1D boundary(shape = shapeFunc, interval = {1,{0,1}});
	end DomainCircular2D;
		class DomainCircular2D
		  extends Domain;
		  parameter Real radius = 1;
		  parameter Real cx = 0;
		  parameter Real cy = 0;
		  Coordinate x (name="cartesian");
		  Coordinate y (name="cartesian";
		  Coordinate r (name="polar");
		  Coordinate theta (name="polar");
		  Region2D interior(theta in (0,2*C.pi), r in (0,radius));
		  Region1D boundary(theta in (0,2*C.pi), r = radius);
		equation
		  x = r*cos(theta) + cx;
		  y = r*sin(theta) + cy;
		end DomainCircular2D;

type EquiLateralTriangleDomain
  extends Cartesian2D (boundary = {side_a, side_b, side_c});
  paremeter Real l;
  parameter Line side_a = Line(0, 0);
  parameter Line side_b = Line(l, 0);
  parameter Line side_c = Line(l/2,sqrt(3)/2);
end RectangularDomain;
	class EquiLateralTriangle2D
	  extends Domain;
	  parameter Real l;
	  function shapeFunc
	  input Real p, q;
	  output Real x = p*(1-q)*l + q*l/2, y = q*l*sqrt(3)/2;
	  end shapeFunc;
	  Coordinate x (name = "cartesian");
	  Coordinate y (name = "cartesian");
	  Region2D interior(shape = shapeFunc, interval = {{0,1},{0,1}});
	  Region1D side_a(shape = shapeFunc, interval = {1,{0,1}});
	  Region1D side_b(shape = shapeFunc, interval = {0,{0,1}});
	  Region1D side_c(shape = shapeFunc, interval = {{0,1},0});
	end DomainRectangle2D;
		class EquiLateralTriangle2D
		  extends Domain;
		  paremeter Real l;
		  Coordinate x (name = "cartesian");
		  Coordinate y (name = "cartesian");
		  Coordinate p;
		  Coordinate q;
		  Region2D interior(p in {0,1}, q in {0,1});
		  Region1D side_a(p = 1, q in {0,1});
		  Region1D side_b(p = 0, q in {0,1});
		  Region1D side_c(p in {0,1}, q =0});
		equation
		  x = p*(1-q)*l + q*l/2;
		  y = q*l*sqrt(3)/2;
		end DomainRectangle2D;