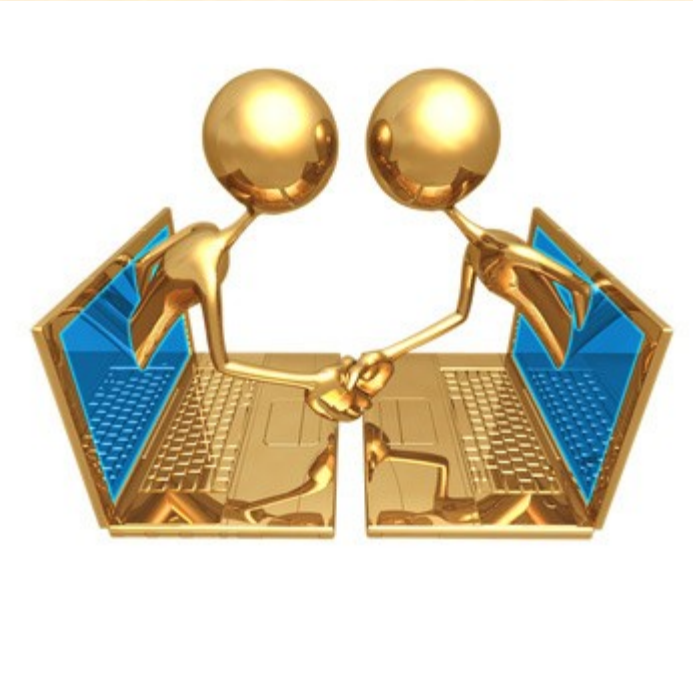
FOSDEM
FREE AND OPEN SOURCE SOFTWARE DEVELOPERS' EUROPEAN MEETING

The contract model of Ada 2012



J-P. Rosen
Adalog
www.adalog.fr

Programming with contracts

- What is it ?
  - With software components, there is a provider of the component who is different from the user of the component
  - For each provided service, define rights and obligations of the user and of the provider of the service
    - A precondition expresses what is required from the user.
    - A postcondition expresses what is promised by the provider.
    - An invariant is a property that always holds (from the POV of the user).
- These conditions are part of the specification
  - Visible !

Programming with contracts

- What are the benefits of language support?
  - ☞Like any check (including simple constraints) :
    - conditions have no effect whatsoever on a working program (presuming such a thing exists).
  - ☞ During program development :
    - Exceptions are raised closer to the cause of the problem
    - Works as kinds of lemma, intermediate steps that help to make program proofs.
  - ☞In service :
    - Protects against faults
    - Facilitates degraded mode

Ada 1983 : constraints

- Restrict the set of possible values of a type

```
type Age is range 0..125;
subtype Adult is Age range 18 .. Age'Last;

procedure Order_Alcohol (Consumer_Age : Adult);
```

Excludes 0

```
function Get_Coefficient return Positive;

…

My_Part := Amount / Get_Coefficient;  -- Necessarily OK
```

Ada 2005 : assertions

- pragma Assert

```
pragma Assert (Condition, Message);
```

- pragma Assertion_Policy
  - ☞Check : if the condition is false, raise Assertion_Error with the given message
  - ☞Ignore : condition not checked

- Enforce invariants, easily removed for production use

Ada 2012 : subtype predicates

- Generalization of the notion of constraint
  - ☞*Static* predicates
    - must be static (!)
    - enjoy many checks at compile time (including full coverage of **case** statements)
  - ☞*Dynamic* predicates
    - no restriction
  - ☞Checked only when Assertion_Policy is Check

```
subtype Even is Integer
   with Dynamic_Predicate => Even mod 2 = 0;

subtype Winter is Month
   with Static_Predicate => Winter in Dec | Jan | Feb;
```

Ada 2012 : Pre and Postconditions

- On subprograms
  - Pre and Post apply to a single type
  - Pre'Class and Post'Class apply also to descendants
  - Checked only when Assertion_Policy is Check
- Special attributes for post-conditions
  - V'Old : value of V on subprogram entrance
  - F'Result : value returned by function F

```
procedure Update_Person (P : in out Person)
   with Post => P.Sex = P.Sex'Old
                and P.Birth_Date = P.Birth_Date'Old;

function Inc(X: Integer) return Integer
   with Pre  => X /= Integer'Last,
        Post => Inc'Result = X'Old+1;
```

Ada 2012 : new expressions

- Quantifiers

```
pragma Assert (for some X in 2 .. N / 2 => N mod X = 0);

-- N is not a prime number...
```

```
type Tab is array (T) of Float;

procedure Sort (A : in out Tab)
   with Post => (A'Length < 2
                 or else (for all I in A'First .. A'Last-1
                                => A (I) <= A (I+1)));
```

- if and case expressions

```
subtype Pet is Animal
   with Static_Predicate =>
     (case Pet is when Cat  | Dog | Horse => True,
                  when Bear | Wolf        => False);
```

- expression functions

Ada 2012 : type invariants

- Only for private types
- Apply only outside the package
  - ☞may be temporarily violated by services inside the package

```
package Places is
   type Disc_Point is private
      with Type_Invariant => Check_In(Disc_Pt);

   function Check_In(D: Disc_Point) return Boolean;
   ...   -- various operations on disc points

private
   type Disc_Point is
      record
         X, Y: Float range -1.0 .. +1.0;
      end record;

   function Check_In (D: Disc_Point) return Boolean is
            (D.X**2 + D.Y**2 <= 1.0)
      with Inline;
end Places;
```

The next step : SPARK

- A formally provable language
  - ☞restricted to a subset of Ada (95)
  - ☞augmented with an assertion language (special comments)
  - ☞compiles with an Ada compiler
- Oriented towards static analysis
  - ☞"Correctness by construction"
  - ☞Availability of proof-making tools
    - free versions available from Praxis/AdaCore
- Demonstrator: Tokeneer project (EAL5)

```
procedure Inc (X : in out Integer);
        --# global in out CallCount;
        --# pre  X < Integer'Last and
        --#      CallCount < Integer'Last;
        --# post X = X~ + 1 and
        --#      CallCount = CallCount~ + 1;
```


Questions?