Alice in Adaland

Using Ada 2012 in practice

Real-life examples of using Ada 2012 features and a discussion of how they improve software reliability and maintainability:

- Pre- and postconditions
- Static predicates
- **in out** parameters for functions
- Expression functions
- Set notation
- **for** ... **of** ... **loop** notation

The case example is a hosted telephone reception system.

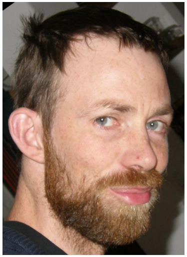
Alice in Adaland

Jacob Sparre Andersen

Jacob Sparre Andersen Research & Innovation
and
AdaHeads K/S

11th June 2013

Jacob Sparre Andersen



Currently:

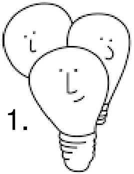
- Independent consultant.
- Co-founder of AdaHeads K/S.
- Co-owner of Koparo Ltd.
- Software architect at AdaHeads.

Background:

- PhD & MSc in experimental physics.
- BSc in mathematics.
- Has taught mathematics, physics and software engineering.
- Worked with bioinformatics, biotechnology and modelling of investments in the financial market.

jacob@jacob-sparre.dk
www.jacob-sparre.dk

AdaHeads K/S



- A software consulting company founded in 2011.
- Four of the owners are active Ada developers:
  - Thomas Løcke
  - Kim Rostgaard Christensen
  - Jacob Sparre Andersen
  - Thomas Pedersen
- Alice is the the core of the first system AdaHeads K/S has been contracted to develop.

tl@adaheads.com krc@adaheads.com
jsa@adaheads.com tp@adaheads.com
www.adaheads.com

Alice, Bob and Chloe

Alice, Bob and Chloe form a hosted telephone reception system being developed by AdaHeads K/S.

Alice	manages where a PBX directs calls when they arrive from the outside and brings Bob live information about the organisations being called.
Bob	is the user interface seen by the receptionists doing the actual work of talking to the callees, taking messages and figuring out where calls should be directed.
Chloe	is the administration interface seen by the staff setting up receptions for new (and existing) customers.

Alice, Bob and Chloe (continued)

- The customer co-funding the development considers the complete system mission critical.
- The customer intends to use it for a long time.
- As Alice is interacting with human callers and receptionists, it is treated as a soft real-time system.

Alltogether we find this a good argument for implementing Alice in Ada.
As our customer wants the user interface to run in (modern) web browsers, we have decided to implement Bob and Chloe in a combination of Dart and HTML.

Tasking and strong typing

Completely forgotten in the abstract – probably because they are a second nature to Ada developers – are:

tasking	We use tasks to manage logically parallel execution. It may speed up the execution, but that is (generally) not why we do it.
typing	Strong typing is a useful tool to avoid mixing up different kinds of objects (even when they are non-composite).

Preconditions

Assuring that we don't accidentally create a reception without at least one end-point:

```
   function Create
     (Title           : in      String;
      Start_At        : in      String;
      End_Points      : in      Receptions.
          End_Point_Collection.Map;
      Decision_Trees  : in      Receptions.
          Decision_Tree_Collection.Map)
     return Instance
     with Pre => (not End_Points.Is_Empty);
```

Postconditions

Telling what changes a subprogram makes to an object:

```
procedure Status_Data
  (Instance : in out Object;
   Request  : in     AWS.Status.Data)
with Post => Instance.Has_Status_Data;
--  Set the client request data. This makes the
    response object aware of
--  Cookies, Sessions, GET/POST request parameters
    and everything else that
--  the AWS.Status.Data object contains.
```

Static predicates

Limiting the length of a string subtype to what our database allocates storage for:

```
    subtype Organization_URI is String
      with Static_Predicate => (Organization_URI'Length
           <= 256);
```

in out parameters for functions

No claim that **in out** parameters are required in this case, but it is there:

```
function Send (Client : in out Client_Type;
               Item   : in     AMI.Packet.Action.
                  Request)
               return AMI.Parser.Packet_Type is
begin
   AMI.Response.Subscribe (Item);
   Client.Send (String (Item.To_AMI_Packet));

   return AMI.Response.Claim (Ticket => Item.
      Action_ID);
end Send;
```

Expression functions

No need to hide the default implementation of this function:

```
   function Clock (PBX : in Instance) return Ada.
        Calendar.Time is
     (Ada.Calendar.Clock);
```

Set notation

Set notation is an easy and readable extension/addition to ranges:

```
function Is_Whitespace (Item : in      Character)
    return Boolean is
   use Ada.Characters.Latin_1;
begin
   return Item in Space | No_Break_Space | HT;
end Is_Whitespace;
```

...

```
          elsif First and then C in '+' | '0' .. '9' then
```

for ... of ... loop notation

Processing characters from a string in order:

```
for C of Item loop
   if Is_Whitespace (C) then
      null; -- removing it
   elsif First and then C in '+' | '0' .. '9' then
      First := False;
      Filled_To := Filled_To + 1;
      Buffer (Filled_To) := C;
   elsif C in '0' .. '9' then
      Filled_To := Filled_To + 1;
      Buffer (Filled_To) := C;
   else
      return Item; -- not a (normal) phone number
   end if;
end loop;
```

External libraries

- AWS – provides the basic HTTP interface implementation.
- GNATcoll – provides database access.
- XMLAda – provides XML parsing.
- Yolk – provides logging, configuration handling and various other utilities on top of AWS and GNATcoll.

New in Ada 2012

Improved checkability:

- Preconditions
- Static predicates

Improved readability:

- Postconditions
- Set notation
- **for ... of ... loop** notation

Not decidable from Alice:

- **in out** parameters for functions
- Expression functions

Contact

Jacob Sparre Andersen
Jacob Sparre Andersen Research & Innovation
`jacob@jacob-sparre.dk`
`http://www.jacob-sparre.dk/`

Jacob Sparre Andersen
+45 21 49 08 04

Source text repositories:

- `https://github.com/AdaHeads/Alice/`
- `https://github.com/AdaHeads/libdialplan/`