Contract-based Programming with Ada 2012

An experience report

Ada 2012 has checked “contracts” and “aspects” for subprograms and types. Some of these are by definition static, while others can be postponed to run-time.

The case example is a hosted telephone reception system, where the core component is written in Ada 2012.

With currently available compilers, the big static analysis benefits of using Ada are related to the basic type system, which also existed in earlier versions of the standard, and the major benefit of switching to Ada 2012 at the moment is in the improved run-time checks.

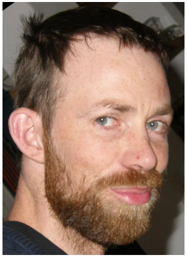
Contract-based Programming with Ada 2012 Experience report

Jacob Sparre Andersen

Jacob Sparre Andersen Research & Innovation
and
AdaHeads K/S

23rd August 2013

Jacob Sparre Andersen



Currently:

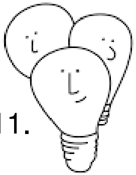
- Independent consultant.
- Co-founder of AdaHeads K/S.
- Co-owner of Koparo Ltd.
- Software architect at AdaHeads.

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
- The case example 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

Raison d'etre for Ada

In the 1970's the US DoD noticed they had a problem with software development.

To solve it they arranged a programming language design competition with:

> *. . . three overriding concerns: program reliability and maintenance, programming as a human activity, and efficiency.*

The result was Ada (1983).

(to make a long story short)

Ada in a single slide

- Procedural programming language.
- Supports object oriented programming (encapsulation, inheritance and dynamic dispatching).
- Modular programming: Packages, child packages and individual subprograms. Generic modules.
- Separates declarations of subprograms and packages in specifications and implementations.
- Concurrent, distributed and real-time programming built in.
- Types distinguished by name (both simple and composite types).

Most recent Ada standard published by ISO in December 2012.

Why we use Ada

The customer co-funding the development considers the complete system mission critical[1], and intends to use it for a long time[2]. As the system is interacting with human callers and receptionists, it is treated as a soft real-time system[3].

And we like creating software in Ada[4].

1 Program reliability.
2 Program maintenance
3 Efficiency
4 Programming as a human activity.

Contracts and aspects in Ada 2012

New in Ada:

- Pre- and postconditions:
  Run-time checks at the call of and return from a subprogram.
- Type invariants:
  Run-time checks on changes to a private type.
- Dynamic subtype predicates:
  Constraints on visible subtypes.
- Static subtype predicates:
  Static constraints on visible subtypes.

Some checks in “old” Ada

- Named “primitive” types:
  No implicit conversions between different named types.
- Parameter passing directions:
  “in”, “out” or “in out” parameters to subprograms checked at compile-time.
- Range checks:
  Array indexing is checked (typically only at run-time).
- Subtypes with ranges:
  Subtype ranges are checked (typically only at run-time).
- Static coverage tests:
  Case statements are checked for full coverage at compile-time.

Preconditions

From the Ada 2012 rationale:

> *A precondition . . . is an obligation on the caller to ensure that it is true before the subprogram is called and it is a guarantee to the implementer of the body that it can be relied upon on entry to the body.*

A reception requires 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

From the Ada 2012 rationale:

> *A postcondition . . . is an obligation on the implementer of the body to ensure that it is true on return from the subprogram and it is a guarantee to the caller that it can be relied upon on return.*

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.
```

Type invariants

From the Ada 2012 rationale:

```
package Places is
   type Disc_Point is private;
   ...   --various operations on disc points
private
   type Disc_Point is
      record
         X, Y: Float range -1.0 .. +1.0;
      end record
      with Type_Invariant
        => Disc_Point.X ** 2 + Disc_Point.Y ** 2 <= 1.0;
end Places;
```

(not used in Alice)

Dynamic subtype predicates

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

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

Static subtype predicates

Inspired by the Ada 2012 rationale:

```
procedure Seasons is
   type Months is (Jan, Feb, Mar, Apr, May, Jun,
                   Jul, Aug, Sep, Oct, Nov, Dec);

   subtype Summer is Months
     with Static_Predicate => Summer in Nov .. Dec |
                                        Jan .. Apr;

   A_Summer_Month : Summer;
begin
   A_Summer_Month := Jul;
end Seasons;
```

The compiler reports:

```
warning: static expression fails static predicate check on "Summer"
```

Coverage check

```
case Level is
   when Debug =>
      System_Message.Debug.Dial_Plan     (Message);
   when Information =>
      System_Message.Info.Dial_Plan      (Message);
   when Notice =>
      System_Message.Notice.Dial_Plan    (Message);
   when Warning =>
      System_Message.Warning.Dial_Plan   (Message);
   when Error =>
      System_Message.Error.Dial_Plan     (Message);
   when Critical =>
      System_Message.Critical.Dial_Plan  (Message);
   when Alert =>
      System_Message.Alert.Dial_Plan     (Message);
   when Emergency =>
      System_Message.Emergency.Dial_Plan (Message);
end case;
```

Static analysis with Ada 2012

Static (compile-time) checking of contracts and aspects is currently only implemented, where it is required by the language standard – and for static subtype predicates.

For now, the big static analysis benefits of using Ada are related to the basic type system, which also existed in earlier versions of the standard, and the major benefit of switching to Ada 2012 at the moment is in the improved run-time checks[5] (and readability).

[5] Although we admittedly haven't yet had cases where Ada 2012 specific run-time checks have found errors in our software.

Contact

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

Source text repositories:

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