Ada and Mini-Ada

A solution to the two-language problem

https://alire.ada.dev/crates/hac

Dr Gautier de Montmollin

FOSDEM 2025

“Breaking the two-language wall”

Prototyping

data processing,
code generation,
simulations,
shell scripts,
plug-ins,
exercises, …

Traditionally: dynamic, duck typing; interactive workflow with global, persistent data

Large applications

Production code

Static typing (for predictability and performance);

data loaded and managed by the application

Domain specific

e.g. SQL

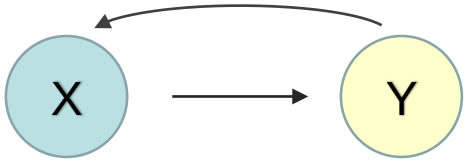
HAC* ← **Ada** → “full Ada” system

*) **HAC** = **H**AC **A**da **C**ompiler

Two-language cases

Case 1: the Prototype – Implementation cycle

- Sketch in pseudo-code or prototype in scripting language **X**. Motivation: abstraction, "clean hands".
- Implementation in **Y**. Motivation: run-time speed; team **Y** cares about bugs.



Issues: Complex process; may require two separate teams.

Solution:

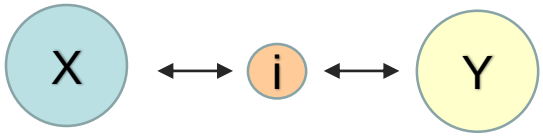
Prototype with **HAC** (**H**AC **A**da **C**ompiler), use "**Full Ada**" * for the implementation. Few rewrites if any.

* e.g. GNAT (the GNU Ada compiler), or another Ada compiler

Two-language cases

Case 2: Coexistence

- Part in scripting language **X** (frequent adjustments).
- Part with "heavy-lifting" in **Y**.
- Interface **i** between both parts.



Issue: May require *three* teams.

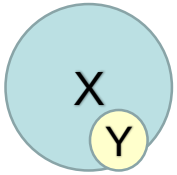
Solution:

- Use "**Full Ada"** or possibly **HAC** for **X,**
- Use only "**Full Ada"** for **Y**.

Two-language cases

Case 3: Embedding

- Application in **X** (industrial software, game, …).
- Scripting embedded in the application, language **Y**.



Issues: For an internal software, needs knowledge of both **X** and **Y** for some people.

Solution:

Embed **HAC** in the "**Full Ada"** application.

Embedding **HAC** in a "**Full Ada**" application

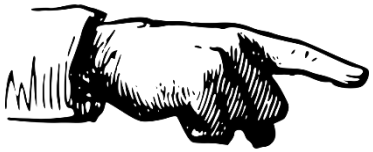
Advantages:

- **Same language** ⟶ +/- same people
- **Code sharing**: Ada packages shared by the HAC program and its host
- **Same type system** (subtypes, enums, arrays, records, …)

Embedding **HAC** in a "**Full Ada**" application

```
with Ada.Command_Line, HAC_Sys.Builder, HAC_Sys.PCode.Interpreter;

procedure HAC_Mini is
  use Ada.Command_Line, HAC_Sys.PCode.Interpreter;
  BD : HAC_Sys.Builder.Build_Data;
  post_mortem : Post_Mortem_Data;
begin
  if Argument_Count > 0 then

    BD.Build_Main_from_File (Argument (1));

    if BD.Build_Successful then
      Interpret_on_Current_IO (BD, 1, "", post_mortem);
    end if;

  end if;
end HAC_Mini;
```

Embedding **HAC** in a "**Full Ada**" application



Live demo, with data exchange

In the HAC project: ./demo/data_exchange_simple