

Porting to Ada 2005

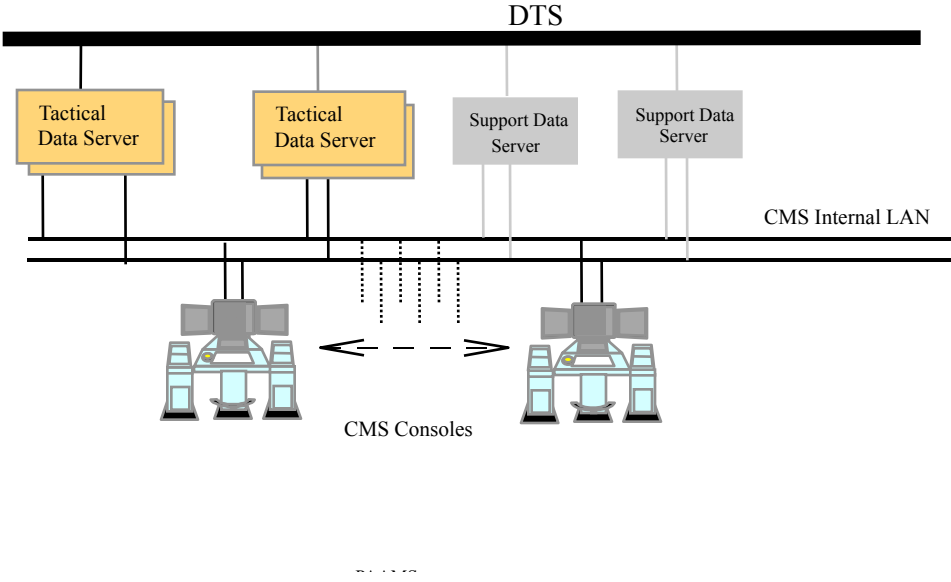
Jeff Cousins



CMS-1 family of Command Systems

- Originally developed for Type 45 destroyers
- Also fitted to HMS Clyde (Falklands patrol vessel)
- and RFA Argus (helicopter training and casualty reception)
- Now in development for retro-fit to Type 23 frigates

Basic CMS-1 Architecture (1)



Basic CMS-1 Architecture (2)

- Dual redundant servers
- Dual or triple redundant ship's highway (DTS)
- Dual redundant local LAN

- Servers mostly Ada
- Consoles mostly C++

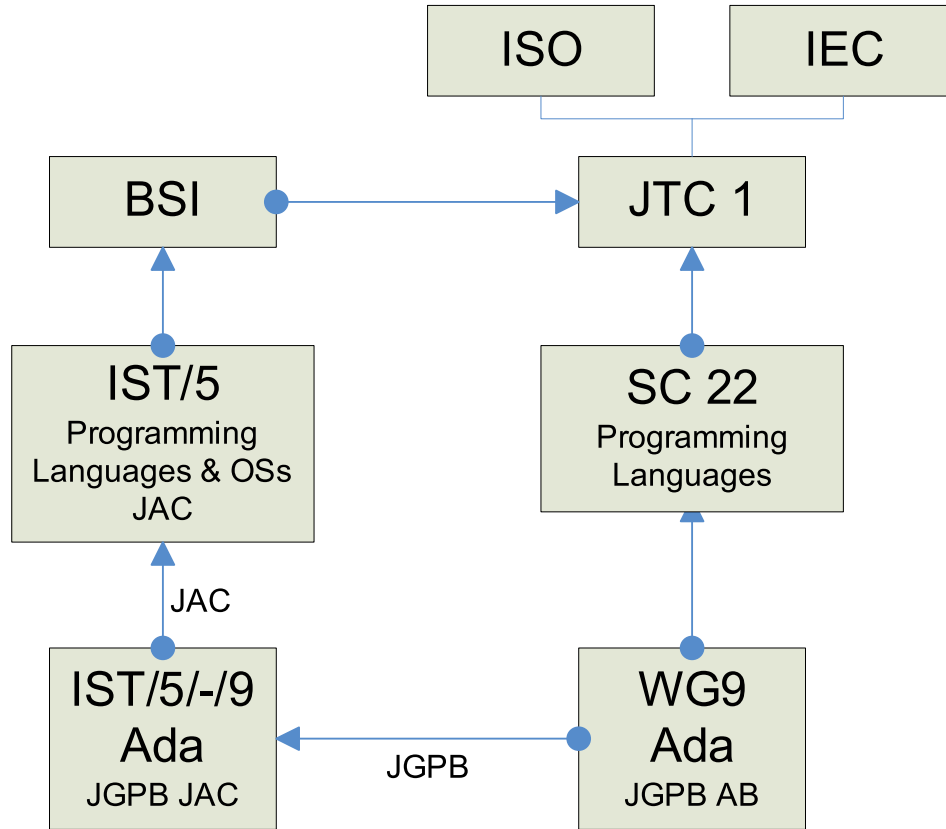
- Argus



- Daring



Standardisation

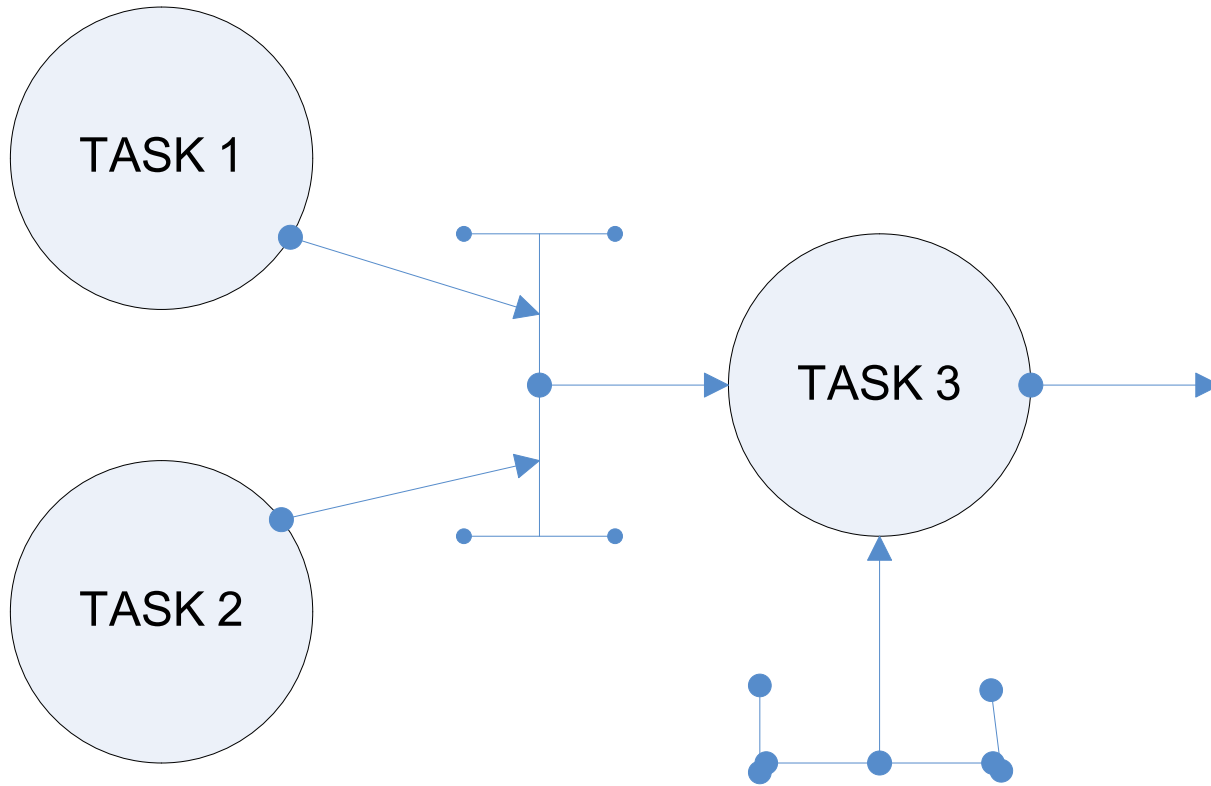


History of porting: Coral 66 to Ada 83

- 'FOR' I := 1 'STEP' 1 'UNTIL' 10 'DO'
- some_statement

- for I in 1 .. 10 loop
- some_statement;
- end loop

History of porting: Coral 66 to Ada 83



History of porting: Ada 83 to Ada 95

- Single exception handler for Constraint and Numeric Errors
- **when CONSTRAINT_ERROR =>**
- to:
- **when CONSTRAINT_ERROR | NUMERIC_ERROR =>**

Porting: Ada 95 to Ada 2005

Ada 2005

New reserved words

- In Ada 2005 “interface” is now a reserved word
- This has mostly hit areas that had a convention of publishing the user interface to a package X in a child package called X.Interface
- Occasionally variables were called interface
- “English” synchronised not affected by the “American” synchronized becoming reserved
- Developing convention to add a prefix, e.g. GC_Interface for GNATCOM
- 149 files affected (out of 19K, 85K after code generation)

No 'access in generic bodies (1)

- 'access no longer allowed in generic body (AI-229)
- Use constant set to x'access in private part of spec
- A retrospective change to Ada 95, besides being required for Ada 2005
- 16 files affected

No 'access in generic bodies (2) – original Ada 95

- --context clauses including
 - generic
 - --parameters
- package P is
 - --types
 - --subprogram_specs including
 - procedure Callback;
- private
 - --types
 - --subprogram_specs
- end P;

No 'access in generic bodies (3) – original Ada 95

- --context clauses including
- with Y;
- generic
- package body P is
- --subprogram_bodies including
- procedure Callback is
- begin
- --;
- end Callback;

- procedure Proc is
- begin
- --
- Y.Z (Callback_Ptr => Callback'Access);
- --
- end Proc;
- end P;

No 'access in generic bodies (4) – new Ada 95 & Ada 2005

- --context clauses including
- with X;
- generic
- --parameters
- package P is
- --types
- --subprogram_specs including
- procedure Callback;
- private
- --types
- --subprograms
- Callback_Access : constant X.Callback_Ptr_Type := Callback'Access;
- end P;

No 'access in generic bodies (5) – new Ada 95 & Ada 2005

- --context clauses including
- with Y;
- generic
- package body P is
- --subprogram_bodies including
- procedure Callback is
- begin
- --;
- end Callback;

- procedure Proc is
- begin
- --
- Y.Z (Callback_Ptr => Callback_Access);
- --
- end Proc;
- end P;

Limited function results (1)

- In Ada 2005 a function return cannot be of limited type
- Make the objects to be returned aliased, return an access type pointing to them, and let caller de-reference using `.all`
- Using an Ada 2005 anonymous access type would affect the fewest files but then the code wouldn't be Ada 95/2005 portable
- Having language specific variants would be a pain for configuration control, so a named access type has been used
- 5 files affected

Limited function results (2) – original Ada 95

- package T.GF is
 - type Gsi_Handle_Type is limited private;
 - -- other types
 - -- subprogram_specs
- private
- type GSI_Handle_Type is ...;
- end T.GF;
- with T.GF;
- with T.S;
- package R.MC is
 - -- types
 - -- subprogram_specs including
 - function Get_GSI_Handle (Source_Name : T.S.Source_Name_T)
 - return T.GF.GSI_Handle_Type;
- end R.MC;

Limited function results (3) – original Ada 95

- package R.MGI is
- -- subprogram_specs
- GSI_Handle_A : T.GF.GSI_Handle_Type;
- GSI_Handle_B : T.GF.GSI_Handle_Type;
- end R.MGI;
- with R.MGI;
- package body R.MC is
- -- subprogram_bodies including
- function Get_GSI_Handle (Source_Name : T.S.Source_Name_T)
- return T.GF.GSI_Handle_Type is
- begin
- case Source_Name is
- when A =>
- return R.MGI.GSI_Handle_A;
- when B =>
- return R.MGI.GSI_Handle_B;
- end case;
- end Get_GSI_Handle;
- end R.MC;
- ...
- GSI_Handle := R.MC.Get_GSI_Handle (My_Source_Name);

Limited function results (4) – Ada 2005 only

- -- Avoids changing T.GF:
- with T.GF;
- with T.S;
- package R.MC is
- -- types
- -- subprogram_specs including
- function Get_GSI_Handle (Source_Name : in T.S.Source_Name_T)
- return access T.GF.GSI_Handle_Type;
- end R.MC;

Limited function results (5) – Ada 2005 only

- package R.MGI is
- -- subprogram_specs
- GSI_Handle_A : aliased T.GF.GSI_Handle_Type;
- GSI_Handle_B : aliased T.GF.GSI_Handle_Type;
- end R.MGI;
- with R.MGI;
- package body R.MC is
- -- subprogram_bodies including
- function Get_GSI_Handle (Source_Name : T.S.Source_Name_T)
- return access T.GF.GSI_Handle_Type is
- begin
- case Source_Name is
- when A =>
- return R.MGI.GSI_Handle_A'Access;
- when B =>
- return R.MGI.GSI_Handle_B'Access;
- end case;
- end Get_GSI_Handle;
- end R.MC;
- ...
- GSI_Handle := R.MC.Get_GSI_Handle (My_Source_Name).all;

Limited function results (6) – Ada 95/2005 portable

- package T.GF is
 - type Gsi_Handle_Type is limited private;
 - type Gsi_Handle_Type_Ptr is access all Gsi_Handle_Type;
 - -- other types
 - -- subprogram_specs
- private
- type GSI_Handle_Type is ...;
- end T.GF;
- with T.GF;
- with T.S;
- package R.MC is
 - -- types
 - -- subprogram_specs including
 - function Get_GSI_Handle (Source_Name : in T.S.Source_Name_T)
 - return T.GF.GSI_Handle_Type_Ptr;
- end R.MC;

Limited function results (7) – Ada 95/2005 portable

- package R.MGI is
- -- subprogram_specs
- GSI_Handle_A : aliased T.GF.GSI_Handle_Type;
- GSI_Handle_B : aliased T.GF.GSI_Handle_Type;
- end R.MGI;
- with R.MGI;
- package body R.MC is
- -- subprogram_bodies including
- function Get_GSI_Handle (Source_Name : T.S.Source_Name_T)
- return T.GF.GSI_Handle_Type_Ptr is
- begin
- case Source_Name is
- when A =>
- return R.MGI.GSI_Handle_A'Access;
- when B =>
- return R.MGI.GSI_Handle_B'Access;
- end case;
- end Get_GSI_Handle;
- end R.MC;
- ...
- GSI_Handle := R.MC.Get_GSI_Handle (My_Source_Name).all;

Limited function results (8)

- Note that GNAT also gives this error when a limited constructor function composes by calling other limited constructor functions, and user defined operators (e.g. "+" and "and") are used to combine the components
- We and AdaCore regard this as a GNAT bug, though some people think that, as for normal functions, operators shouldn't be allowed to return limited results either
- The work-around is simply to use the ugly "and" (X, Y) instead of X and Y
- 2 files affected

Limited function results (9) - original

- package FT is
 - type F is limited private;
 - -- other types
 - function "and" (L : F, R : Integer) return F;
- private
 - type Buffer;
 - type F is access Buffer;
- end FT;
- with FT; use FT;
- package body O is
 - function FOT (Into : F, Component_B : Integer) return F is
 - begin
 - return Into and Component_B;
 - end FOT;
- end O;

Limited function results (10) – work-around

- package FT is
 - type F is limited private;
 - -- other types
 - function "and" (L : F, R : Integer) return F;
- private
 - type Buffer;
 - type F is access Buffer;
- end FT;
- with FT; use FT;
- package body O is
 - function FOT (Into : F, Component_B : Integer) return F is
 - begin
 - return "and" (Into, Component_B);
 - end FOT;
- end O;

Explicit null exclusion for renamed subprograms (1)

- When a subprogram renames another and they dispatch upon an access parameter, then the access parameter must be explicitly null excluding (RM-3.9.2, changed as a result of AI95-00404-01)
- It would be easy enough to add “not null” but then the code wouldn’t be Ada 95/2005 portable
- Having language specific variants would be a pain for configuration control, so the renames has been changed to a body calling a named routine – a stepping stone procedure
- Note that AI447 will retrospectively change Ada 95 to allow explicit null exclusion, to ease transition
- 2 files affected

Explicit null exclusion for renamed subprograms (2) – original Ada 95

- procedure Process_Table_Has_Been_Cleared
- (Subscriber : access Subscriber_Type;
- First_Instance_Id : in Instance_Id_Type;
- Last_Instance_Id : in Instance_Id_Type) is

- begin

- --;

- end Process_Table_Has_Been_Cleared;

- procedure Table_Has_Been_Partially_Cleared
- (Subscriber : access Subscriber_Type;
- First_Instance_Id : in Instance_Id_Type;
- Last_Instance_Id : in Instance_Id_Type) renames
- Process_Table_Has_Been_Cleared;

Explicit null exclusion for renamed subprograms (3) – Ada 2005 only

- procedure Process_Table_Has_Been_Cleared
- (Subscriber : not null access Subscriber_Type;
- First_Instance_Id : in Instance_Id_Type;
- Last_Instance_Id : in Instance_Id_Type) is

- begin

- --;

- end Process_Table_Has_Been_Cleared;

- procedure Table_Has_Been_Partially_Cleared
- (Subscriber : not null access Subscriber_Type;
- First_Instance_Id : in Instance_Id_Type;
- Last_Instance_Id : in Instance_Id_Type) renames
- Process_Table_Has_Been_Cleared;

Explicit null exclusion for renamed subprograms (4) – Ada 95/2005 portable

- procedure Process_Table_Has_Been_Cleared
- (Subscriber : access Subscriber_Type;
- First_Instance_Id : in Instance_Id_Type;
- Last_Instance_Id : in Instance_Id_Type) is

- begin

- --;

- end Process_Table_Has_Been_Cleared;

- procedure Table_Has_Been_Partially_Cleared
- (Subscriber : access Subscriber_Type;
- First_Instance_Id : in Instance_Id_Type;
- Last_Instance_Id : in Instance_Id_Type)
- is
- begin
- Process_Table_Has_Been_Cleared (Subscriber => Subscriber,
- First_Instance_Id => First_Instance_Id,
- Last_Instance_Id => Last_Instance_Id);
- end Table_Has_Been_Partially_Cleared;

Inconsistency in Interfaces.C.Strings

- The procedure Update in Interfaces.C.Strings no longer adds a nul character (AI-242)
- A retrospective change to Ada 95, besides being required for Ada 2005
- Only 1 file affected
- But not nice since not an illegality detectable by the compiler, may not be found until run time

Containers (1)

- Vectors as an extensible array to store "lists" of record component names and enumeration literals
- No problems using them
- It's probably an implementation issue, but the impression that we had from looking at the GNAT implementation was that Ada Vectors after most operations are exactly the capacity they need to be. They do not have an extra bit of capacity for expansion in the same way as GNAT's implementation of Ada Unbounded Strings. It's easy to work around this by making calls to `Reserve_Capacity` and overestimating the capacity required but it's not as convenient

Containers (2)

- We still use Booch Unbounded Maps for mapping things like enumeration literals to their represented values and vice versa. Ada 2005 Hashed_Maps do not appear to provide bucket statistics that Booch Maps do. Bucket statistics are useful in determining the effectiveness of the hashing function. We can test the hashing function in isolation, but it would be more convenient being able to get the statistics from the container
- Overall, it's nice to finally have containers as part of the predefined Ada library

“Would be nice” for Ada 2010 (1)

- One of the goals for Ada is supposed to be “no surprises”
- But...
- `procedure Long_Name (X: Positive);`
- ...
- `-- Conformance of constraints not enforced`
- `procedure Short_Name (Y : Integer) renames Long_Name;`
- ...
- `Short_Name (-1); -- Compiles but raises Constraint_Error when run`
- Similarly for an in out generic parameter, which is a special case of renaming

“Would be nice” for Ada 2010 (2)

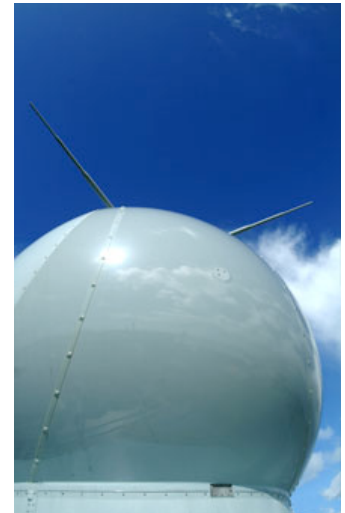
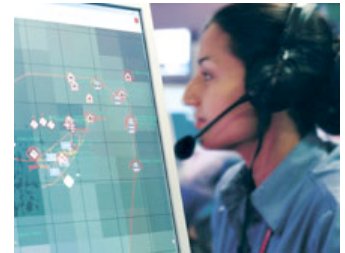
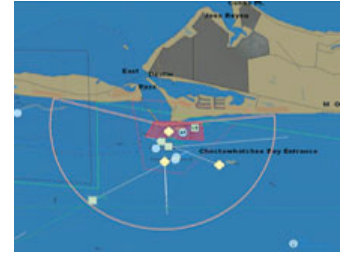
- type My_Float is new Float;
- function "=" (L, R : My_Float) return Boolean is
- begin
- if L > 0.99 * R and then L < 1.01 * R then
- return True; -- Near enough
- else
- return False;
- end if;
- end "=";
- type Position is record
- Latitude : My_Float;
- Longitude : My_Float;
- end record;
- Position1 : Position := (90.00, 90.00);
- Position2 : Position := (90.01, 89.99);
- begin
- if Position1 = Position2 then – Uses original predefined equals for components

“Would be nice” for Ada 2010 (3)

- How about
- `pragma Composable_Equality;`
- or even
- `pragma Work_Right_Like_Wot_Tagged_Types_Do;`
- ?

Conclusions

- Quite straightforward to port to Ada 2005
- Slightly harder to use call-backs and limited types
- Anonymous access types and containers useful



BAE Systems Integrated System Technologies (Insyte) Limited
Victory Point
Lyon Way, Frimley, Camberley
Surrey, GU16 7EX
United Kingdom
Telephone +44 (0) 1276 603000

