The Case for Ada at the USAF Academy

Ricky E. Sward, Martin C. Carlisle, Barry S. Fagin, and David S. Gibson

Department of Computer Science
USAF Academy, CO

{ricky.sward, martin.carlisle, barry.fagin, david.gibson}@usafa.edu
Overview

- Academy Background
- Using Ada at the Academy
- Language Features
- Compiler Features
- Development Experience
- Conclusions
Overview

• Academy Background
• Using Ada at the Academy
• Language Features
• Compiler Features
• Development Experience
• Conclusions
US Air Force Academy

Mission

“Inspire and develop outstanding young men and women to become Air Force officers with knowledge, character and discipline; motivated to lead the world's greatest aerospace-force in service to the nation.”
US Air Force Academy

- Graduates enter Air Force as 2\textsuperscript{nd} Lts
- Pilots, navigators, missile/space ops, computer, communications officers
- Typically five year commitment
- Our majors will lead other programmers
US Air Force Academy

- 500 Faculty – 25% civilian, 75% military
- Military normal tour around 3 years
- Military with PhD’s can stay longer
- Most civilian faculty have PhD
US Air Force Academy

- Every cadet takes 30 core courses in engineering, physics, chemistry, computer science, etc.
- Average course load is 6 courses over 18-week semesters
- Computer Science, Computer Engineering degrees compressed to three years
Overview

- Academy Background
- Using Ada at the Academy
- Language Features
- Compiler Features
- Development Experience
- Conclusions
Using Ada at the Academy

• Ada used in core course
  – 1996 – 2001 programming intensive course
  – 2001 – 2003 Information Technology focus
  – Fall 2003 using visual flow charting tool
• Ada in major’s courses
  – Primary programming language
  – Computer Science and Computer Engineering
  – Used in first three courses of majors
Using Ada at the Academy

• We will continue to use Ada
• Curriculum Committee debate
• Choice was based on the following criteria
  – Language features
  – Compiler features
  – Development experience
• Re-examine this choice every few years
Overview

- Academy Background
- Using Ada at the Academy
- **Language Features**
- Compiler Features
- Development Experience
- Conclusions
Language Features

• Strong typing
  – Errors found at compile time, not run time
• Case insensitivity
• Lack of single character errors
• Subtypes and enumeration types
  – Superior to Java, C++ and C#
• Mixed-language pragmas
• Ada port to .NET framework
Language Features

• Support for Imperative paradigm
  – Essential for teaching introductory programming
  – Learn control flow, assignment and modularization
  – Not possible in Java, since OO required from start

• Support for Object-Oriented paradigm
  – Builds on imperative fundamentals
  – Easy transition using ADT concepts
  – Learn OO gradually when intellectually ready

• Helps our majors succeed in limited time
Overview

- Academy Background
- Using Ada at the Academy
- Language Features
- Compiler Features
- Development Experience
- Conclusions
Compiler Features

- Compile time error messages
  - Useful and accurate
  - Steer students in right direction
  - More helpful than run time errors

- Strong typing and good compiler error messages help our students succeed
with Ada.Text_Io;
use Ada.Text_Io;
procedure Color_Prog is
  type Colors is ( Red, White );
  C : Colors := White;
begin
  case C is
    when Red =>
      Put("Red");
    when White =>
      Put("White");
  end case;
end Color_Prog;

• Compiler error message if “Blue” added
Overview

• Academy Background
• Using Ada at the Academy
• Language Features
• Compiler Features
• Development Experience
• Conclusions
Development Experience

• Industry acceptance
  – Our majors will lead, not program
  – Can focus on education, not training of programmers

• Support for both Imperative and Object-Oriented
  – Decided against Java because OO must be first
  – Possible in C#, so we considered in more detail

• Textbook support
  – Currently no CS-2 textbooks available for C#
  – Major drawback of using C# as primary language
Development Experience

• Development environment considerations
• Visual Studio .NET for C#
  – Very powerful and complicated tool
  – Touch and feel including “intelli-sense”
  – Easy, rapid development of code
  – Graphical User Interface (GUI) builder outstanding
Development Experience

• AdaGIDE for Ada 95
  – Excellent tool easily understood by novice programmer
  – Powerful enough for senior level capstone design courses
  – We control the look and feel of the environment
  – GUI support from Rapid is not as good as Visual Studios
with Ada.Text_IO;
use Ada.Text_IO;
procedure Color_Prog is
  type Colors is
    (Red, White, Blue);
  C : Colors := White;
begin
  case C is
    when Red =>
      Put("Red");
    when White =>
      Put("White");
  end case;
end Color_Prog;
Overview

• Academy Background
• Using Ada at the Academy
• Language Features
• Compiler Features
• Development Experience
• Conclusions
Conclusions

• We will continue to use Ada
• Objections to Ada not technical
  – Appear to be sociological
  – Due to lack of widespread acceptance
• We are educating leaders, not programmers
• Languages are adopting desirable Ada features
  – Strong typing, generics, software engineering support
• We will continue to examine our choice