Ada - The Project: The DoD High Order Language Working Group

A review of Colonel William A. Whitaker's Article

Date: 26 Sept 2012

Presenter: Jack Myers
Course Programming Languages (Graduate)

About Ada

- Ada is an object-oriented, high-level computer programming language which was derived from Pascal.
- Originally created for embedded systems for the United States Department of Defense (DoD).

Expansion

- Application domain:
  - Large-scale info systems
  - Distributed systems
  - Scientific computing
  - Systems programming
- User base:
  - All major defense agencies in West
  - Aerospace community
  - Telecommunications
  - Process control & monitoring systems

Who's Using Ada (besides defense agencies) ?

- **Air Traffic Control**
  - Australia, Belgium, Brazil, Canada, China, Czech Republic, Denmark, Finland, France, Germany, Greece, Hong Kong, Hungary, India, Ireland, Kenya, Netherlands, New Zealand, Pakistan, Scotland, Singapore, South Africa, Spain, Sweden, Taiwan, UK, USA, Vietnam

European Air Traffic Flow Management
The system operates on the basis of defined blocks of airspace known as "domains". Each ATM Surveillance Unit tracks all traffic in its own defined airspace; adjacent units overlap. In the areas of overlap, inter-unit track coordination functions take place, ensuring system tracking continuity. In this manner, adjacent ARTAS Units can co-ordinate their tracks to build a unique, coherent and continuous Air Situation Picture over the complete area.

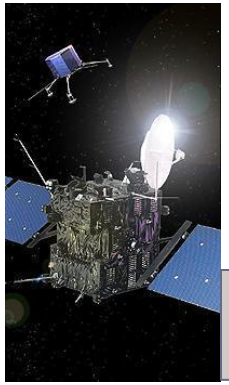
- **Commercial Aviation**
  - Airbus 320,330,340,380
  - Boeing 737, 747, 757,767, 787
  - Lockheed Martin Hercules "hurricane chaser"

- **Railway Transportation**
  - Metrorail in Athens, Budapest, Cairo, Calcutta, Caracas, Hong Kong, London, Madrid, Mexico City, Paris, Prague; NYC subway system; French high speed rail…

Who else?

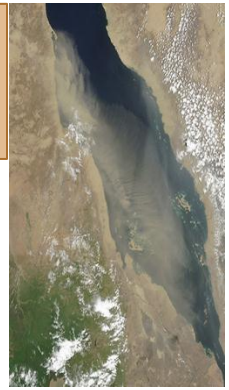
Other Industries
- Commercial Rockets
- Data Communications
- Desktop and Web Applications
- Nuclear Medicine
- Automotive Assembly Robotics
- Nuclear Energy Control Systems
- Steel Mills…

- **Scientific Space Vehicles**
  - ENVISAT-1 (European Environmental Satellite),
  - NASA's Aqua, Aura and Terra (water, atmospheric and land observations)
  - European Space Agency (e.g., Mars Express, Rosetta "Comet Chaser")…



Rosetta will rendezvous with comet 67P in May 2014

Dust over the Red Sea, as imaged from Aqua on 1 Sep 2012



- **Swiss Bank Accounts** PostFinance
- **Semiconductors** PHILIPS sense and simplicity
- **Software Verification** IRS

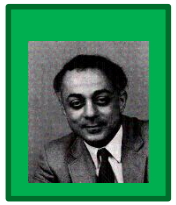
"Twice a year new versions of tax processing programs are tested by Product Assurance and approved for use.

I decided to use Ada because of the language allowed me to quickly generate reliable code. I believed it would be too expensive for me to produce the software in C.

…developmental costs must be kept to a minimum. Ada is a high-level language that is well equipped to produce reliable software both quickly and inexpensively. "

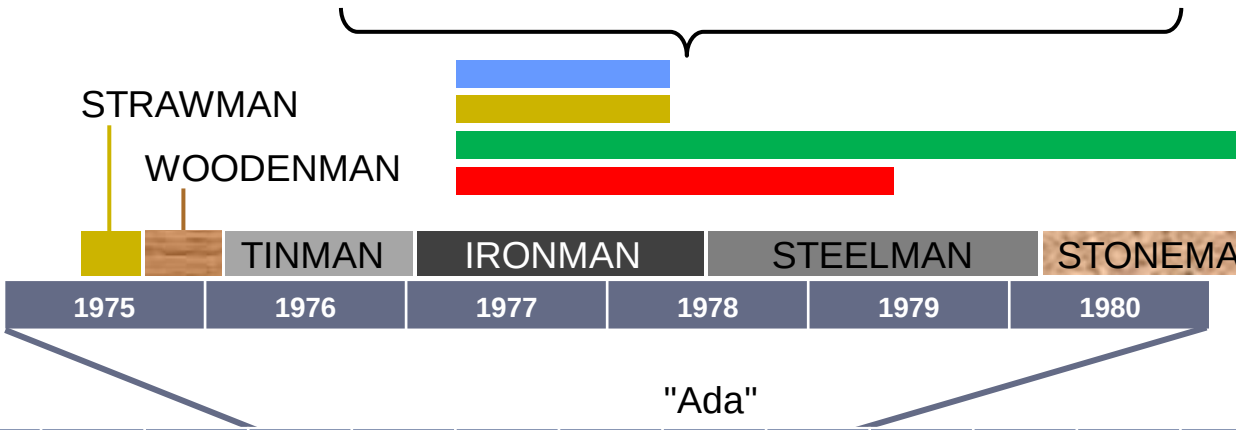
Justice Moore, IRS

History of Ada



Stanford Research Institute

Intermetrics Honeywell SRI Intl SofTech



Ada is entranced
17 year old Ada Lovelace meets Charles Babbage, the inventor of the first automatic computing engine

1833 . . . 1843 . . . 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 198

Ada publishes paper

containing the first published account of a stepwise sequence of operations for solving mathematical problems. Retrospectively, she is labeled the "first programmer."

Whitaker joins OSD

- William A. Whitaker joins the Office of the Secretary of Defense from the Air Force Weapons Lab.
- Commissions "Software Initiative"

HOLWG Formed

- or Higher Order Language Working Group
- Whitaker selected as chair

DoD Ada Mandate

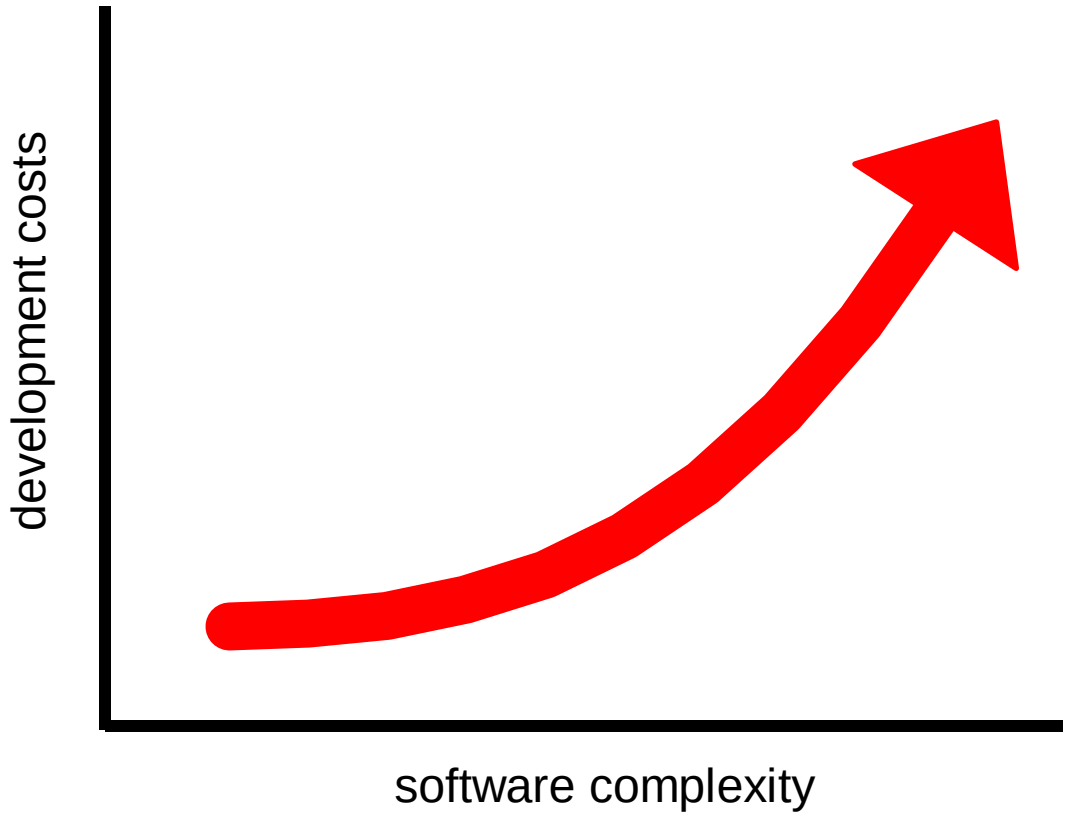
(memo sent in 1983)

"The Ada programming language shall become the single common programming language for Defense mission-critical applications."

BACKGROUND

Why did the DoD begin to look at higher order languages?

The Problem



- Air Force alone spending over $1.5 billion per year on SW.
  - $8B in 2012 $
  - Entire 2012 budget for NSF
- DoD was spending over $3 billion per year on SW
  - Entire 2012 govt budget for Energy
- Each of the services had programs to develop unique languages, architectures and hardware

The Fiscal Pressures

Software

- Most DoD SW costs for embedded systems:
  - Weapons systems
  - Target detection and tracking
  - Weapons direction
  - Communications
  - Command and control
  - Avionics
  - Simulators
  - Test Equipment
- SW component of these systems increasingly larger.

Hardware

- Non-portable SW disincentivized obsolescence of expensive HW.
- Yearly HW maintenance costs beginning to exceed cost of replacement.

The Cultural Factors

- Management may have been short-sighted:
  - Failing to see the long-term implications of the status quo.
- Little attention was paid to developing reusable tools.

Question

Whitaker commented that there was no incentive for computing standards in the DoD, and argues that "short-term management goals (contractor and government) encouraged proliferation."

So approximately how many different languages do you think were in use at DoD circa 1973 to 1975?

A. Fifteen
B. Several dozen
C. One to two hundred
D. Over 450

Language Situation at DoD pre-HOLWG

- Over 450

- http://www.youtube.com/watch?feature=player_embedded&v=0yXwnk8Cr0c#!

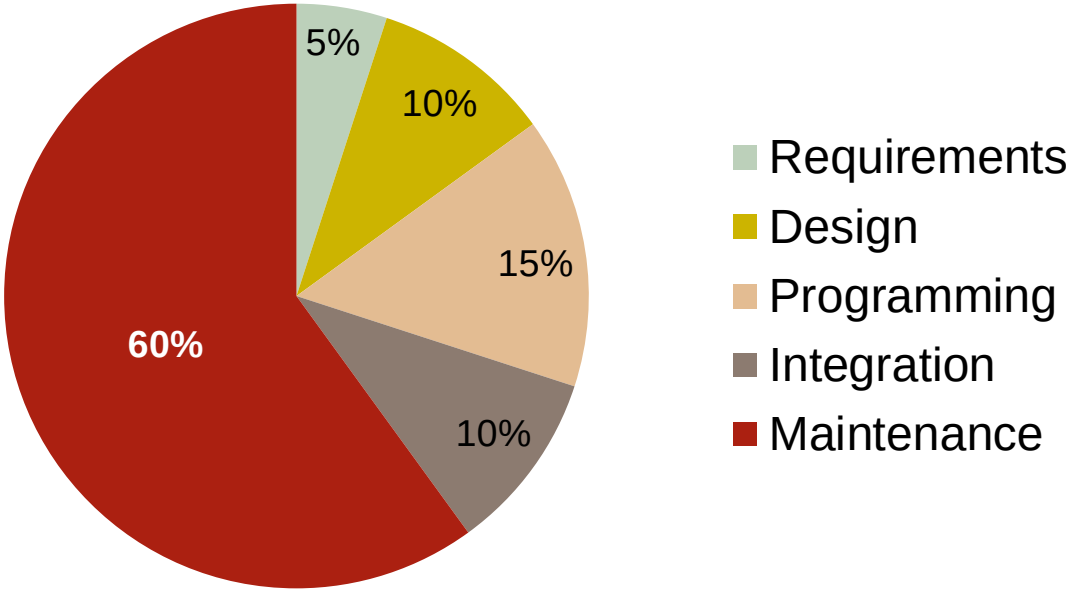
Robert B. K. Dewar, President and CEO of AdaCore. Distinguished Reviewer of Ada Design. Former Chairman of the Algol 68 Committee, which is how he became involved with HOLWG. MIT Lecture, 2006

The Premise

A common language would promote the production of and use of common tools and reusable code.

...and thus, save money!

Estimated Life Cycle Costs for Large SW Systems
(> 10K lines of code)



Source: Dr. John Mylopoulos, Department of Computer Science University of Toronto

Advantages of a Modern, High Order Language

- Reduced programming costs
- Increased readability of programs
- Ease of modification and maintenance
- Allow for automated testing of software
- Required for real-time, parallel processing
- No more assembly!
  - Less readable
  - Less portable
- Error checking facilities
- More reliable programs
- Potential for more efficient compilers

Compilers
Debuggers

Text editors
Interactive help
Automated testing
Large module libraries
Auto-program from specs

Advantages of a Single, High Order Language

- Eliminate the costs of supporting multiple languages
- The burden to create software tools can be distributed across multiple projects – overarching tool development could become a reality.
- Modules and human resources can be shared from project to project
- Lower training costs

"…investigate the requirements and specifications for … commonality, compare with existing DoD efforts and recommend adoption of the necessary common language or languages."

The birth of HOLWG
28 Jan 1975

THE COMMITTEE

A unique approach to solving DoD's high cost of SW problem

HOLWG

Whitaker
(chairman)

Standards Committees
- FORTRAN
- COBOL
- PL/I

US Army

US Navy

US Air Force

Defense Advanced Research Projects Agency

Defense Communications Agency

National Security Agency

France
- Industry Rep
- Government Reps

Germany
- Industry Rep
- Academic Rep
- Government Rep

UK Ministry of Defence

HOLWG Charter

1. Formulate requirements for common DoD high order languages.

2. Compare those requirements with existing languages.

3. Recommend adoption or implementation of the necessary languages.

Objectives

- The language should facilitate reduction of all life cycle SW costs.
- Portability of software across projects and across HW.
- Flexibility needed for facile maintenance.
- Reliability is key.
- Readability more important than writability.
- Must not be less efficient than writing machine code.

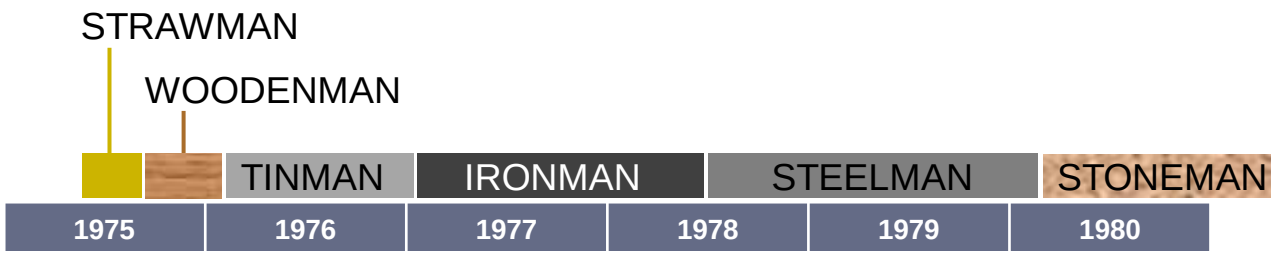
THE REQUIREMENTS

Treating language development like any other application

High Level Requirements / Constraints

- Machine independence
- "Applications user language" -- written for users, not minimalistic
- No need for assembly
- File input/output system
- "go to" (thanks to the Navy)
- Support for 6 bit characters
  - In 1975, still using IBM keypunches that used 6 bits.
  - 6 bits can only encode 64 distinct characters; only upper-case letters, numerals, and some European characters, punctuation characters and control characters would fit.
- Fixed-point arithmetic
  - In 1975, DoD HW still limited to fixed-point representations
  - With less bits available, precision ↓, however calculation speed ↑

Iterations of Requirements



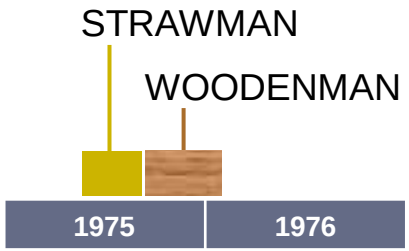
Desire to provoke discussion

- Strawman
  - Laid out the basic technical requirements
  - Considered both "too specific" and "too vague"!

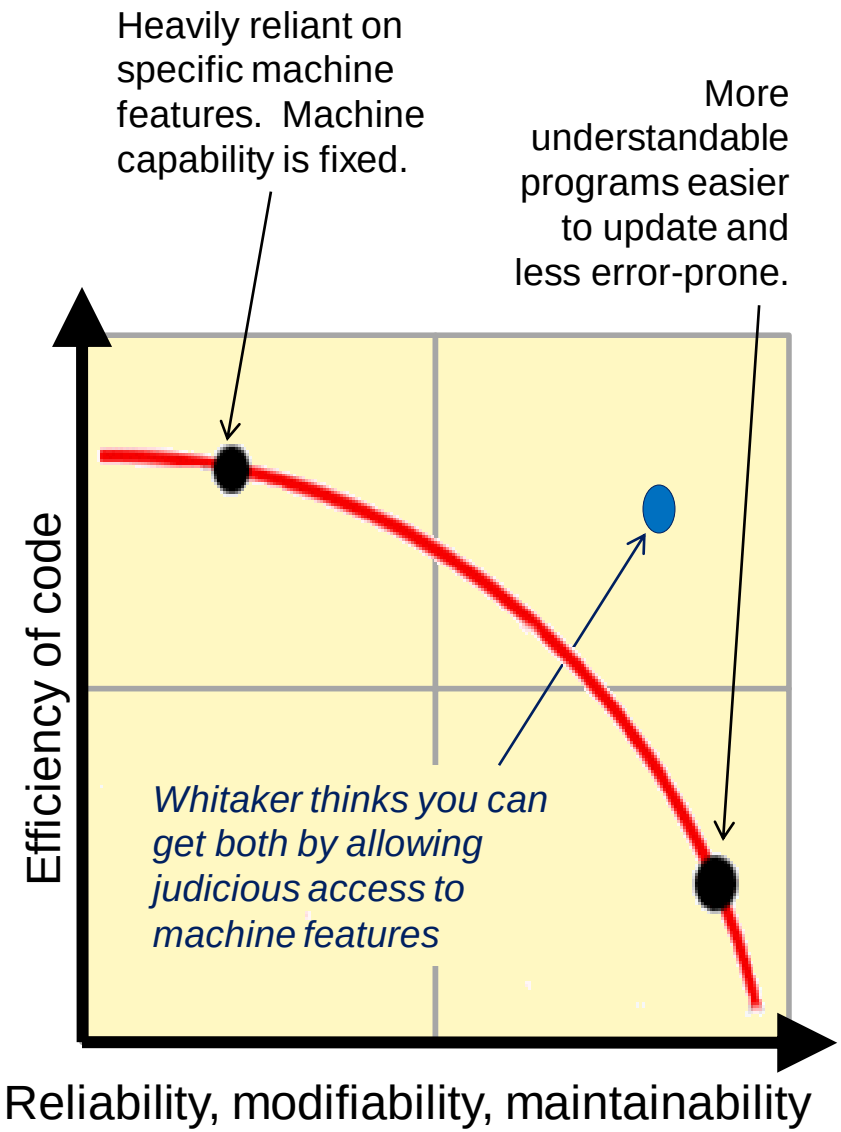
Ada Language Principles

1. Every user level aspect will be **formally specified**. None will be left to the translator implementer, operating system or object machine.
2. There will be a **traceable** operational and/or technological **requirement** for each primitive data, operation and control structure.
3. The language will not reveal minor differences in computer architecture.
4. Machine dependent code cannot appear interspersed in-line in source language programs.
5. The **semantics** will be **determinable** from the description.
6. Constructs will have only one reasonable interpretation
7. An **axiomatic definition** - mathematically complete in the Turing sense.
8. No sacrifice of clarity for efficiency; no special features for rare cases.
9. All defaults in a program will be specifically provided in the language specification or in the program. Defaults will be minimal.

Iterations of Requirements



- Woodenman
  - Discussed needed characteristics of the language
  - Object Efficiency vs. Program Clarity and Correctness
  - Programming Ease vs. Safety from Programming Errors



Question

Whitaker discusses the tradeoff between the stringency of a language and its ease to write. "There is a clear tradeoff between programming ease and safety. The more tolerant the programming language and the less it requires in specifications of the intent and assumptions of the programmer, the easier the coding task."

One such example would be the need to declare variables.

Where does Ada fall on this issue?

Programming Ease vs. Safety from Programming Errors

- On the side of safety

- http://www.youtube.com/watch?feature=player_embedded&v=0yXwnk8Cr0c#!

Robert B. K. Dewar, President and CEO of AdaCore. Distinguished Reviewer of Ada Design. Former Chairman of the Algol 68 Committee, which is how he became involved with HOLWG. MIT Lecture, 2006

Requirements for Variables

The **type** of each variable will be **explicitly individually specified** in the source program.

Scalar variables, arrays and **records** will be the only primitive variable structures in the language.

Variables and arrays will be of **fixed type**.

All **variables** will be **initialized** at the time of their **allocation**. All initial values will be explicitly specified in the program (not necessarily within this program module).

The language will provide for **initialization** of variables **at their definition**.

Arrays will be of **homogeneous** type.

The user will be able to specify array origin.

The number of dimensions for an array will be fixed but not limited by the translator.

The language will support **dynamic array size**. The default for arrays will be static size.

The language will support **records of a heterogeneous type** with fixed conditional structure.

More from Woodenman

Syntax

- Clarity and readability of programs should be primary criteria for selecting a syntax.
  - Simple syntax
  - Minimal abbreviations
- User should not be able to override syntax
  - e.g., change precedence rules or add operators

I. SOURCE LANGUAGE

- I.A Character Set
  - I.A.1 Types
  - I.A.2 Variables
  - I.A.3 Parameters
- I.B Operations
- I.C Control Structures
- I.D Syntax
  - I.D.1 Parsing Rules
  - I.D.2 Lexical Structure
  - I.D.3 Character Set

Flexibility

A. MAJOR CHARACTERISTICS

B. SOURCE LANGUAGE CHARACTERISTICS

- B.1 Data Types
- B.2 Operations
- B.3 Variables, Literals and Constants
- B.4 Extension Facilities
- B.5 Scopes
- B.6 Expressions
- B.7 Control Structures
- B.8 Parameters
- B.9 Standard Extensions
- B.10 Syntax

"The user should be able, within the source language, to extend existing operations to new data types.

When an operation is an abstraction of an existing operation for a new type or is a generalization of an existing operation, it is inconvenient, confusing and misleading to use any but the existing operator symbol or name."

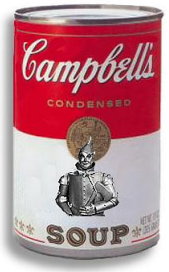
Woodenman

- Semantic extensions versus syntactic ones

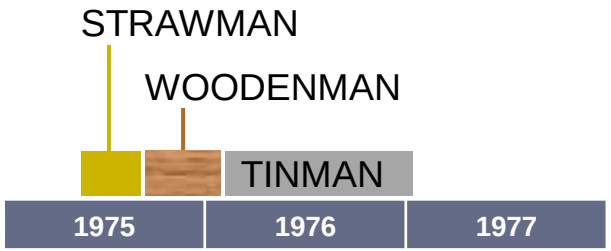
Analogy:

Coining new words is okay.
Coining new rules of grammar is not.

V.E.4 [Can Extend Existing Operators]
V.H.2 [No Syntax Extensions]



One Language?



- **Different branches claim uniqueness**
  - Each branch has different requirements.
- …but…
  - Individual requirements were identical
- …yet there was still hedging!

DoD Directive 5000.29:

"DoD approved high order programming languages will be used to develop Defense systems unless…"

26 April 1976

Others had common goals:

- British Ministry of Defence standardized on CORAL 66.
- Commission of the European Communities adopted concept of common language for EEC.
- Other Governmental Initiatives
  Germany → PEARL
  France → LTR
  Japan → CTL-B

EXISTING LANGUAGES

Could an existing language meet the requirements?

Report to the ~~HOLWG~~ High Order Language Working Group (HOLWG)

14 Jan[redacted]77

EXECUTIVE SUMMARY

87 p.

LANGUAGE EVALUATION COORDINATING COMMITTEE

Prepared by:

Serafino Amoroso, ~~(Chairman)~~ — CENTACS, Fort Monmouth, NJ

~~and~~

Peter Wegner, (Technical Advisor) — Brown University, Providence, RI

~~and with assistance~~ from:

Derek Morris	CENTACS, Fort Monmouth, NJ
Douglas White	RADC, Rome, NY
Warren Loper	NELC, San Diego, CA
Lloyd Campbell	BRL, Aberdeen, MD
Calvin Showalter	NAVAIR, Washington, DC

410 122

23 Languages to Consider

~~ALGOL 60~~	ECL	LIS	PEARL
ALGOL 68	EUCLID	LTR	PL/I
~~CMS-2~~	~~FORTRAN~~	MORAL	~~SIMULA 67~~
~~COBOL~~	HAL/S	RTL/2	SPL/1
~~CORAL 66~~	~~JOVIAL J 3B~~	Pascal	~~TACPOL~~
CS-4	~~JOVIAL J-73~~	PDL/2	

Widely Used:	COBOL, FORTRAN, PL/I
Dismissed!	COBOL (business-oriented language), FORTRAN (not modern enough)
Intl Choices:	PEARL (Germany), LTR & LIS (France), CORAL 66 & MORAL (UK)
PL/I - based	HAL-S (NASA), PEARL
PASCAL - based	SPL/I, PDL/2 (Texas Instruments), LTR, CS-4 (Intermetrics), LIS, EUCLID
DoD Languages:	CMS-2 (Navy), TACPOL (Army), JOVIAL (Air Force)

Without exception, the following languages were found by the evaluators to be inappropriate to serve as base languages for a development of the Common Language: FORTRAN, COBOL, TACPOL, CMS-II, JOVIAL J73, JOVIAL J3B, SIMULA 67, ALGOL 60, and CORAL 66.

Modifications of PASCAL, PL/I or ALGOL 68 remained.

Why not PL/I, Pascal or ALGOL 68?

- **PL/I**
  - Primitive data types, control structures, extensibility, modularity…
  - Too much freedom of expression for the user
  - Complex language
- **Pascal**
  - Lacking parallel processing, real-time error handling, precision facilities
  - Does not permit independent program and data modules
- **ALGOL 68**
  - "Difficult to implement"

Contractor Evaluations

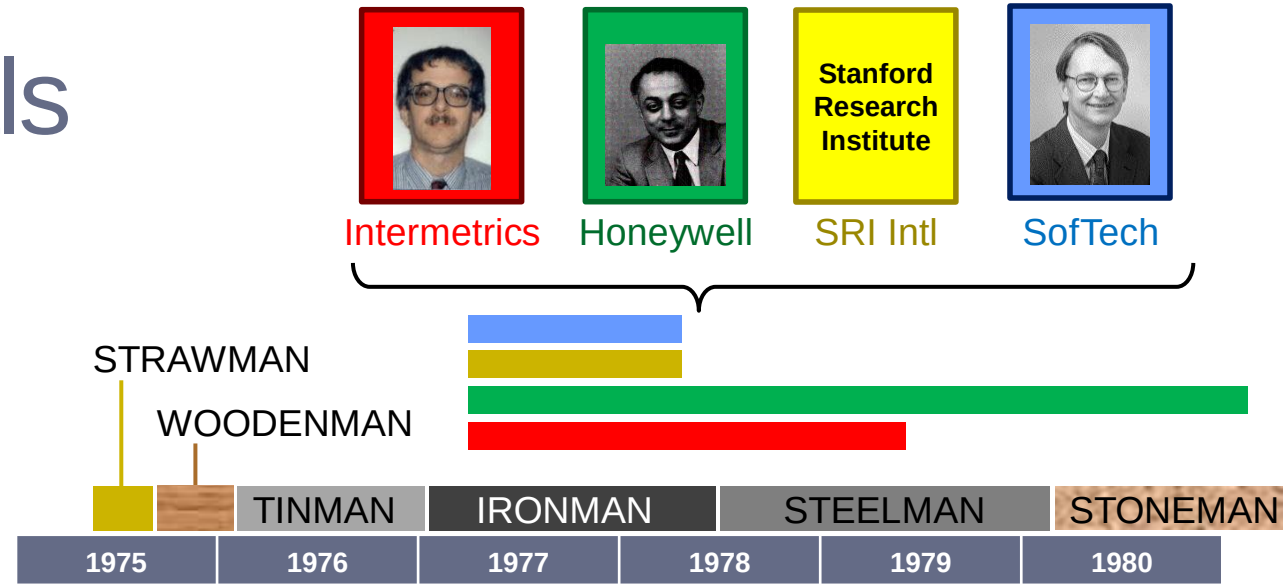
1. Determine degree of compliance of language to requirements
2. Feasibility of modifying the language to bring it into compliance
3. Identify features in excess of the requirements

	SOFTECH	INTERMETRICS	RLG	CSC	SAI	IBM	OTHER
FORTRAN		X				X	
COBOL		X				X	
PL/I		X				X*	
TACPOL	X	X	X				
HAL/S						X	
CMS-2			X	X			
CS-4		X	X	X	X		
J-3B	X			X			
J-73		X		X			
ALGOL 60					X		
CORAL 66			X	X			X
ALGOL 68	X				X		
SIMULA 67	X						
PASCAL	X		X				
LIS					X		X
LTR							X
RTL/2							X
PEARL				X			X
SPL/I				X			
EUCLID				X			
MORAL							X
ECL							X
PDL/2							X

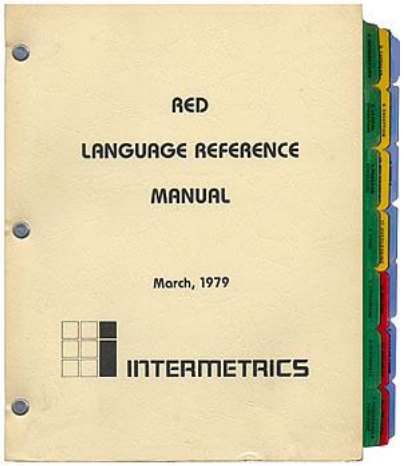
DEVELOPING ADA

How to build a new language

Proposals



- Eighty volunteer analysis teams were formed to analyze the designs.
  - Language Experts
  - Users ← smart !



Red *nearly* made it.

Ada Mandate

Whitaker noted that Richard DeLauer, the Under Secretary of Defense for Research and Engineering, sent out a memo mandating Ada for future DoD mission critical systems.

How effective do you think this was?

"The Ada programming language shall become the single common programming language for Defense mission-critical applications."

From 1984 onward…

The mandate for DoD to use Ada only

- Sometimes disregarded even for critical applications!

- http://www.youtube.com/watch?feature=player_embedded&v=0yXwnk8Cr0c#!

Robert B. K. Dewar, President and CEO of AdaCore. Distinguished Reviewer of Ada Design. Former Chairman of the Algol 68 Committee, which is how he became involved with HOLWG. MIT Lecture, 2006

CONTROLLING ADA

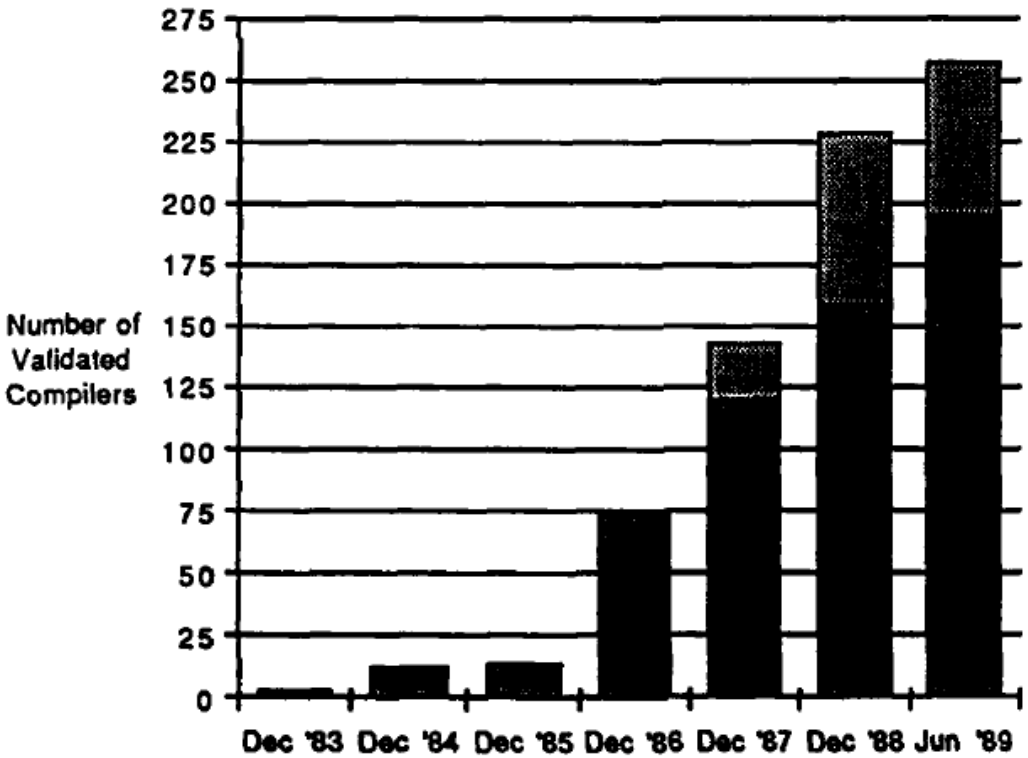
How to ensure the language is maintained well.

Compilers

- HOLWG was worried that compilers would exceed the scope of the language and add their own proprietary standards.
- Difference between a "standard" language and a "common" one.
- Single compiler versus multiple compilers
  - Control versus Features

Compilers

- Ada compilers can range from under $1000 to over $100K.
- Compilers can be tailored to the application needs without distorting the language.
- Competition breeds improved compiler features:
  - Program libraries
  - Generation of machine code (especially for embedded systems)
  - Level of error messaging and warnings
  - Interactive help
  - Interface (e.g., ability to browse modules)
  - Vendor support



* Source : AdaIC Vol VII, No. 1

Figure 1 Validated Ada Compilers (1983-1989).

Ada Language Control

- Compiler verification
  - Ada Compiler Validation Capability suite of test programs
- Language Standardization
  - Ada became ANSI standard, then ISO standard
  - Ada Language Reference Manual

ADA OVER TIME

How the language has evolved.

Object Orientation and Ada

Simula 67, an object oriented language, was one of the languages reviewed by HOLWG. Did the developers of Ada build object orientation into the language?

Object Orientation and Ada

- No, not at first

- http://www.youtube.com/watch?feature=player_embedded&v=0yXwnk8Cr0c#!

Robert B. K. Dewar, President and CEO of AdaCore. Distinguished Reviewer of Ada Design. Former Chairman of the Algol 68 Committee, which is how he became involved with HOLWG. MIT Lecture, 2006

Version	RFC #	Date
Ada 83	ANSI-MIL-STD-1815	22 Jan 83
	ISO-8652	1987
Ada 95	ISO-8652	1995
Ada 2005	Corrigendum Amendment	2001 2007
Ada 2012?	ISO-8652	submitted

Versions of Ada

- Ada 83
  - Original version used in DoD
  - Object-based, but not object-oriented
    - Generic Units – "templates which are parameterized or not, and from which corresponding (nongeneric) subprograms or packages can be obtained." ANSI-MIL-STD-1815A
    - No full support for inheritance or run-time polymorphism

Ada Version Comparison

Object-Oriented Programming	Ada 83	Ada 95	Ada 2005	Ada 2012
Derived types	√	√	√	√
Tagged types		√	√	√
Multiple inheritance of interfaces			√	√
Named access types	√	√	√	√
Access parameters, Access to subprograms		√	√	√
Enhanced anonymous access types			√	√
Aggregates	√	√	√	√
Extension aggregates		√	√	√
Aggregates of limited type			√	√
Unchecked deallocation	√	√	√	√
Controlled types, Accessibility rules		√	√	√
Accessibility rules for anonymous types			√	√
Preconditions and postconditions				√
Type invariants				√
Subtype predicates				√

Ada 83 Grammar

(from ANSI-MIL-STD-1815A)

generic_declaration → *generic_specification*

generic_specification → *generic_formal_part subprogram_specification*

→ *generic_format_part package_specification*

generic_formal_part → **generic** {*generic_parameter_declaration*}

generic_parameter_declaration → *identifier_list* : [**in** [**out**]] *type_mark* [:= *expression*];

→ **type** *identifier* **is** *generic_type_definition*;

→ *private_type_declaration*

→ **with** *subprogram_specification* [**is** *name];*

→ **with** *subprogram_specification* [**is** <>];

generic_type_definition → (<>)

→ **range** <>

→ **digits** <>

→ **delta** <>

→ *array_type_definition*

→ *access_type_definition*

identifier → *letter* [[underline] *letter_or_digit*]

letter_or_digit → letter

→ digit

letter → upper_case_letter

→ lower_case_letter

upper case letters	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
lower case letters	a b c d e f g h I j k l m n o p q r s t u v w x y z
digits	0 1 2 3 4 5 6 7 8 9

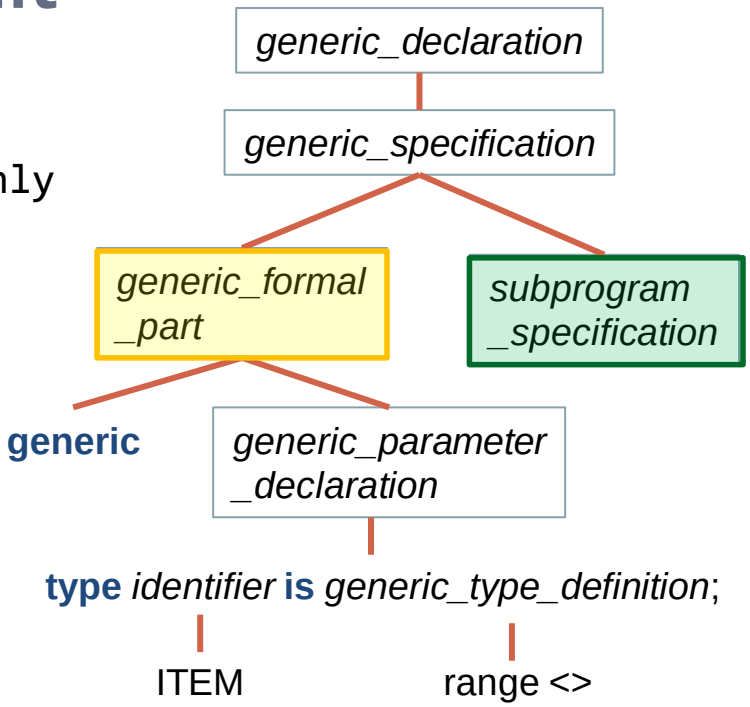
Example of Generic Unit

```
generic
   type ITEM is range <>;   -- integer class only
```

```
procedure Exchange_Data(X,Y : in out ITEM);
```

```
procedure Exchange_Data(X,Y : in out ITEM) is
Temp : ITEM;
begin
   Temp := X;
   X := Y;
   Y := Temp;
end Exchange_Data;
```

generic_procedure_body



- Generic declaration declares a generic unit
- Specification of a procedure specifies:
  - Identifier (Exchange_Data)
  - Formal parameters (X and Y)
  - …their mode (in out) and their type (ITEM; i.e. INTEGER)

```
generic
   type ITEM is range <>;   -- integer class only

procedure Exchange_Data(X,Y : in out ITEM);

procedure Exchange_Data(X,Y : in out ITEM) is
Temp : ITEM;
begin
   Temp := X;
   X := Y;
   Y := Temp;
end Exchange_Data;

with Exchange_Data;
with Ada.Text_IO, Ada.Integer_Text_IO;
use Ada.Text_IO, Ada.Integer_Text_IO;

procedure SwapSome is
   type MY_INT is new INTEGER range 1..128;

   procedure SwapInt is new Exchange_Data(INTEGER);
   procedure SwapNew is new Exchange_Data(MY_INT);

   Index1 : INTEGER := 17;
   Index2 : INTEGER := 33;
   Limit1 : MY_INT  := 3;
   Limit2 : MY_INT  := 7;
```

Object features of Ada 83.

- `MY_INT` is a derived type, permissable in Ada 83
- `SwapInt` and `SwapNew` utilize the generic unit to instantiate new objects.

```
generic
   type ITEM is range <>;   -- integer class only

procedure Exchange_Data(X,Y : in out ITEM);

procedure Exchange_Data(X,Y : in out ITEM) is
Temp : ITEM;
begin
   Temp := X;
   X := Y;
   Y := Temp;
end Exchange_Data;
```

formal parameters

```
procedure SwapSome is
   type MY_INT is new INTEGER range 1..128;

   procedure SwapInt is new Exchange_Data(INTEGER);
   procedure SwapNew is new Exchange_Data(MY_INT);

begin
   SwapInt(Index1, Index2);
   SwapNew(Limit1, Limit2);
end SwapSome;
```

actual parameters

Mode in out

The formal parameters are variables and the procedure allows both reading and updating of the values of the associated parameters.

Value of the **actual parameter** is copied into value of the associated **formal parameter**.

After completion of subprogram, the value of the **formal parameter** is copied back into the associated **actual parameter.**

Pass by value-result, aka "copy-in, copy-out"

Code written by Professor Laurent Pautet
Network and Computer Science Department
Telecom ParisTech

Ada 95 should be Object Oriented and not just Object Based

"Ada 9X should provide:

1. an easily implemented and efficient mechanism for dynamically selecting a subprogram that is to be called with a particular argument list;

2. a means of separating the set of subprograms that can be selected dynamically from the code that makes the selection"

Specification 4.1-A(1)

- Seeks the ability to associate <u>operations</u> (subprograms) with <u>objects</u>, and to <u>dynamically</u> select and execute those operations.

- **Run-time polymorphism**

"Ada 9X should make it possible to define new declared entities whose properties are adapted from those of existing entities by the addition or modification of properties or operations in such a way that:

- the original entity's definition and implementation are not modified;

- the new entity (or instances thereof) can be used anywhere the original one could be, in exactly the same way."

Specification 4.3-B(1)

- In Ada 83, recompilation was often needed when abstractions were broken to extend to a new abstraction.

- **Inheritance**

Ada Version Comparison

Programming Structure, Modularity	Ada 83	Ada 95	Ada 2005	Ada 2012
Packages	√	√	√	√
Child units		√	√	√
Limited with clauses and mutually dependent specs			√	√
Generic units	√	√	√	√
Formal packages		√	√	√
Partial parameterization			√	√
Conditional expressions, Case expressions				√
Quantified expressions				√
In-out parameters for functions				√
Iterators				√
Expression functions				√

Ada Mandate revoked in 1997

- With the increased availability of COTS software
- Also it was unlikely the DoD would ever return to the chaos of the 1970s.

References

For More Information on…	Source
Ada Lovelace	"Ada Lovelace - The Babbage Engine | Computer History Museum." *Computer History Museum*. 2008. http://www.computerhistory.org/babbage/adalovelace
Details on Ada 95	Barnes, John. *Ada 95 Rationale: The Language, the Standard lLbraries*. Berlin: Springer-Verlag, 1995.
Who's Using Ada	Feldman, Michael B.. "Who's Using Ada?." *School of Engineering & Applied Science at The George Washington University*. George Washington University, Feb. 2012. http://www.seas.gwu.edu/~mfeldman/ada-project-summary.html
Robert Dewar's Full Lecture	"Ada Past, Present and Future | Lectures | Ada Answers | AdaCore." AdaCore : The GNAT Pro Company. http://www.adacore.com/adaanswers/lectures/ada-past-present-and-future/
Transcribed Strawman, Woodenman, Tinman, Ironman	Transcribed by Mary S. Van Deusen. http://www.iment.com/maida/computer/requirements/strawman.htm http://www.iment.com/maida/computer/requirements/woodenman.htm http://www.iment.com/maida/computer/requirements/tinman.htm http://www.iment.com/maida/computer/requirements/ironman.htm
The Evils of Goto	Dijkstra, Edsger. "Goto Statement Considered Harmful." *Communications of the ACM* 11.3 (1968): 147-48. https://files.ifi.uzh.ch/rerg/arvo/courses/kvse/uebungen/Dijkstra_Goto.pdf
HOLWG Report from Language Evaluation Coordinating Committee	http://www.dtic.mil/dtic/tr/fulltext/u2/a037635.pdf
The state of Ada Compilers in 1990.	Commonwealth of Austraila. Information Technology Division. *ADA Compiler Benchmarking: Initial Investigations.* By R. Vernik, I. Turner, and S. F. Landherr. 1990. US Technical Information Service. Web. http://www.dtic.mil/dtic/tr/fulltext/u2/a224789.pdf
Ada Version Comparisons	"Ada Comparison Chart." AdaCore.com. http://www.adacore.com/adaanswers/about/ada-comparison-chart
Ada 83 ANSI Standard	American National Standards Institute, Inc. *ANSI/MIL-STD-1815A: Ada Programming Language*. 17 Feb. 1983. http://www.everyspec.com/MIL-STD/MIL-STD-1800-1999/ANSI-MIL-STD-1815A_8646/