Rigorous Methods for Software Engineering (F21RS-F20RS)
Getting Started with SPARK 2014

Andrew Ireland
Department of Computer Science
School of Mathematical and Computer Sciences
Heriot-Watt University
Edinburgh

Overview

- Context and a little history.
- Accessing SPARK toolkit and an Ada compiler – Linux and Windows.
- How to statically analyze, compile and execute SPARK code, e.g. *Hello World!*

A Brief History of Ada

- In response to increased software development and maintenance costs the US Defence Department in 1975 published a set of strict safety criteria for programming languages that were to be used in their systems.
- None of the available languages met the criteria so a competition was set-up to design a new language.
- The winner was Ada (1983) and became a ISO Standard.
- A major revision of the language was completed in 1995, giving rise to Ada 95: revision included object-oriented constructs, optional annexes, *e.g.* systems programming, real-time systems, distributed systems, security, ...
- The last major revision came with Ada 2012, where the language extensions included *aspects* and *pragmas*. These allow a programmer to write both executable statements and assertions, i.e. properties about the intended behaviour of your code that can be verified statically.

Ada – Some Applications

- Ada is the most commonly used language for Air Traffic Control Systems worldwide.
- Ada is used increasingly in commercial "fly-by-wire" aircraft such as the Boeing 777 and Airbus.
- Ada is used within advanced avionics, *i.e.* fighter jets, and many military command and control applications.
- Ada is used for embedded systems in nuclear power plants, rail transportation systems, industrial process control, ...

An Advert for SPARK like Programming Language

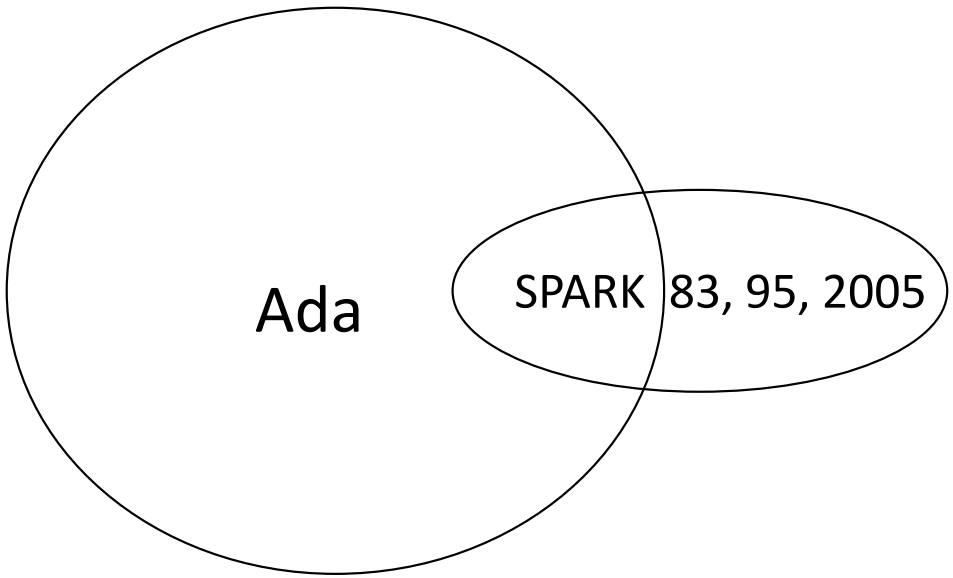
"It is not too late! I believe that by careful pruning of the Ada language, it is still possible to select a very powerful subset that would be reliable and efficient in implementation and safe and economic to use."

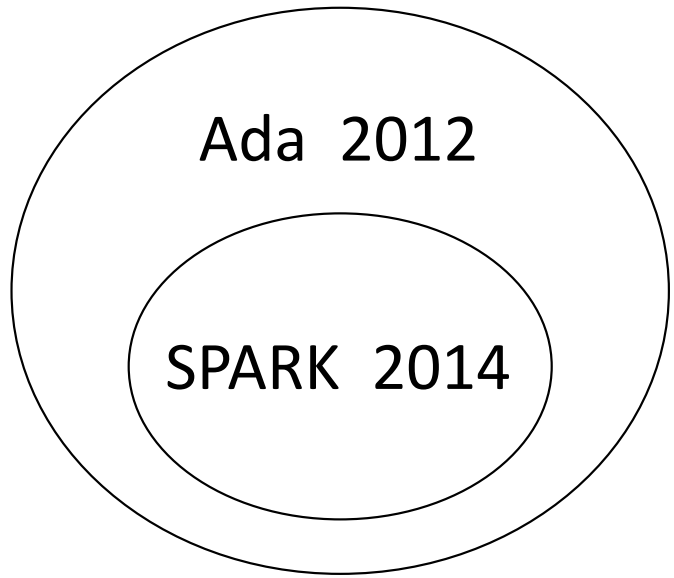
Professor Tony Hoare,
1980 ACM Turing Award Lecture

What is SPARK?

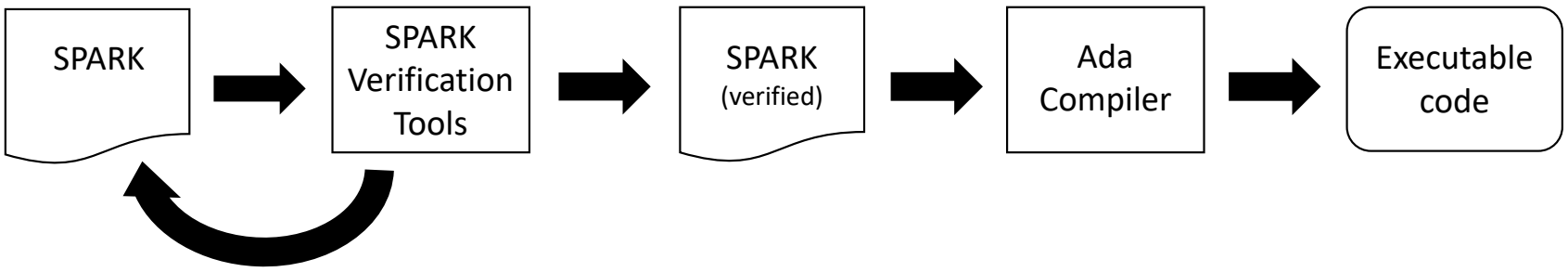
- SPARK is a high level programming language aimed at high integrity applications.
- SPARK was designed to exploit the strengths of Ada while eliminating the potential for ambiguities and insecurities, *e.g.*
  - functions in SPARK are true mathematical functions, *i.e.* can not have side-effects, so any ambiguity in terms of order of evaluation is eliminated.
  - Pointers are prohibited.
  - Aliasing is prohibited.
- SPARK was designed to support verification, both in terms of mainstream static analysis, *i.e.* flow analysis, and formal proof.
- The amount of space a SPARK program requires at run-time can be predicted via static analysis, *e.g.* guaranteed to have no memory leakage.

Ada-SPARK Relationships





The SPARK Approach



- The SPARK verification tools are applicable before coding is complete, *i.e.* the contracts should proceed the coding.
- SPARK approach advocates "correctness-by-construction".
- Note that adding contracts to existing Ada code, i.e. "Sparking the Ada" – is **not recommended!**

SPARK – Some Applications

SHOLIS: a sophisticated decision support system for helicopters landing on a Type 23 Frigate HMS Sutherland and Merlin HM2.

Eurofighter Typhoon: one of the most advanced multi-role combat aircraft

iFACTS: provides a range of decision support and management tools for air traffic controllers - including electronic flight strips and aircraft trajectory prediction

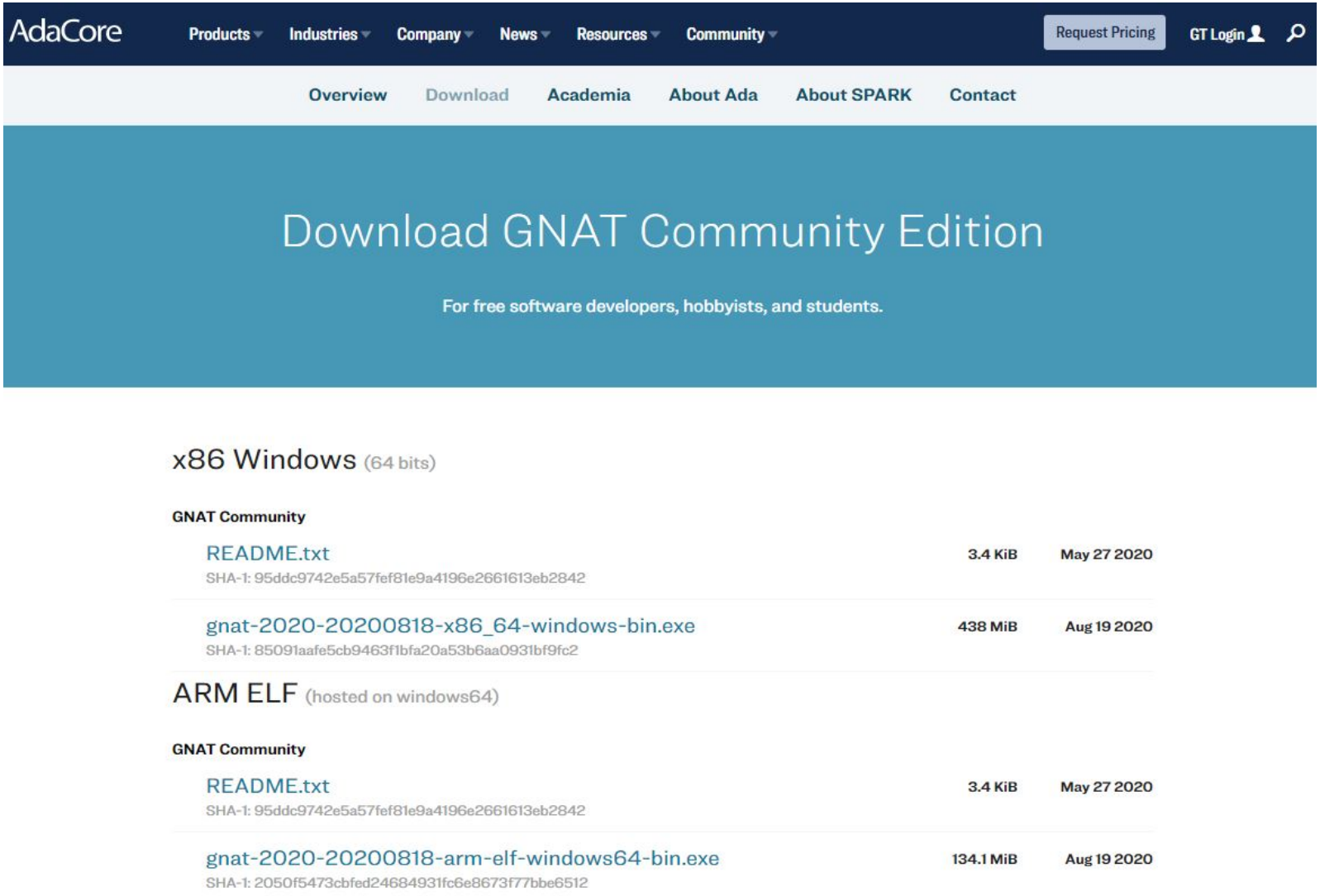
Accessing Ada and the SPARK Toolkit (Edinburgh)

- Linux:
  - Available in the School Linux Lab (EMB 2.50).
  - Remote access via **X2GO** for details see:

    `https://www.macs.hw.ac.uk/cs/faq.html#Qnx`
- Windows:
  - Available in the University Windows Lab (EMB 2.52).
  - Remote access via **KeyServer** for details see:

    `https://www.hw.ac.uk/uk/services/is/it-essentials/keyserver.htm`

Down loading Ada and the SPARK Toolkit



Use the **Community 2020** version, which you can download from `https://www.adacore.com/download`

Hello World!

```
pragma SPARK_Mode (On);
with Text_IO;
-- My first program (this is a comment)!
procedure Hello is
begin
   Text_IO.Put_Line("Hello WORLD!");
end Hello;
```

A closer look at the language later, for now let's see how to analyze, compile and execute this code.

SPARK: Static Analysis and Compilation

- The SPARK tools can be run either via the:
  - Linux **command line** OR
  - Windows **GNAT Programming Studio**
- Here I will focus on the Linux **command line** version.

Linux Command Line: Basics

- **Note that before you can use Ada or the SPARK toolkit you need to type** `usegnat` **at the command line in order to configure GNAT for Ada 2012.** This only needs to be done once per Linux shell window.
- Good practice to create a separate directory for each SPARK program or application, and separate source and object files. I recommend:

```
<<root-name>>
    src  -- source, e.g. ads/adb files
    obj  -- compilation and executable files
    <<project-file-name>>.gpr -- project file
```

Linux Command Line: Basics

- A **project file** tells the SPARK tools where files are located and provides configuration settings, e.g. compilation options.
- Here is a project file (`myproject.gpr`) for my Hello World program:

```
project MyProject is
   for Source_Dirs use ("src");
   for Object_Dir use "obj";
   for Main use ("hello.adb");
   package Compiler is
      for Default_Switches ("Ada") use ("-gnatwa");
   end Compiler;
end MyProject;
```

- `Main` is the only program specific part of this project file.
- A copy of the above project file is located in: `http://www.macs.hw.ac.uk/~air/rmse/SPARK/code/`

Linux Command Line: Static Analysis

Static analysis is performed using the `gnatprove` command, e.g.

```
gnatprove -P myproject.gpr hello.adb

Phase 1 of 2: generation of Global contracts ...
Phase 2 of 2: flow analysis and proof ...
Summary logged in .../Hello/obj/gnatprove/gnatprove.out
```

Linux Command Line: Compilation and Execution

- The GNAT Ada `gnatmake` command automates the compilation process, e.g.

```
gnatmake -P myproject.gpr
```

```
Compile
   [Ada]          hello.adb
Bind
   [gprbind]      hello.bexch
   [Ada]          hello.ali
Link
   [link]         hello.adb
```

The resulting executable has no file extension, e.g. `hello` and can be found within the `obj` directory. To execute it, simply type its name via the command line!

```
hello
Hello WORLD!
```

Additional Guidance

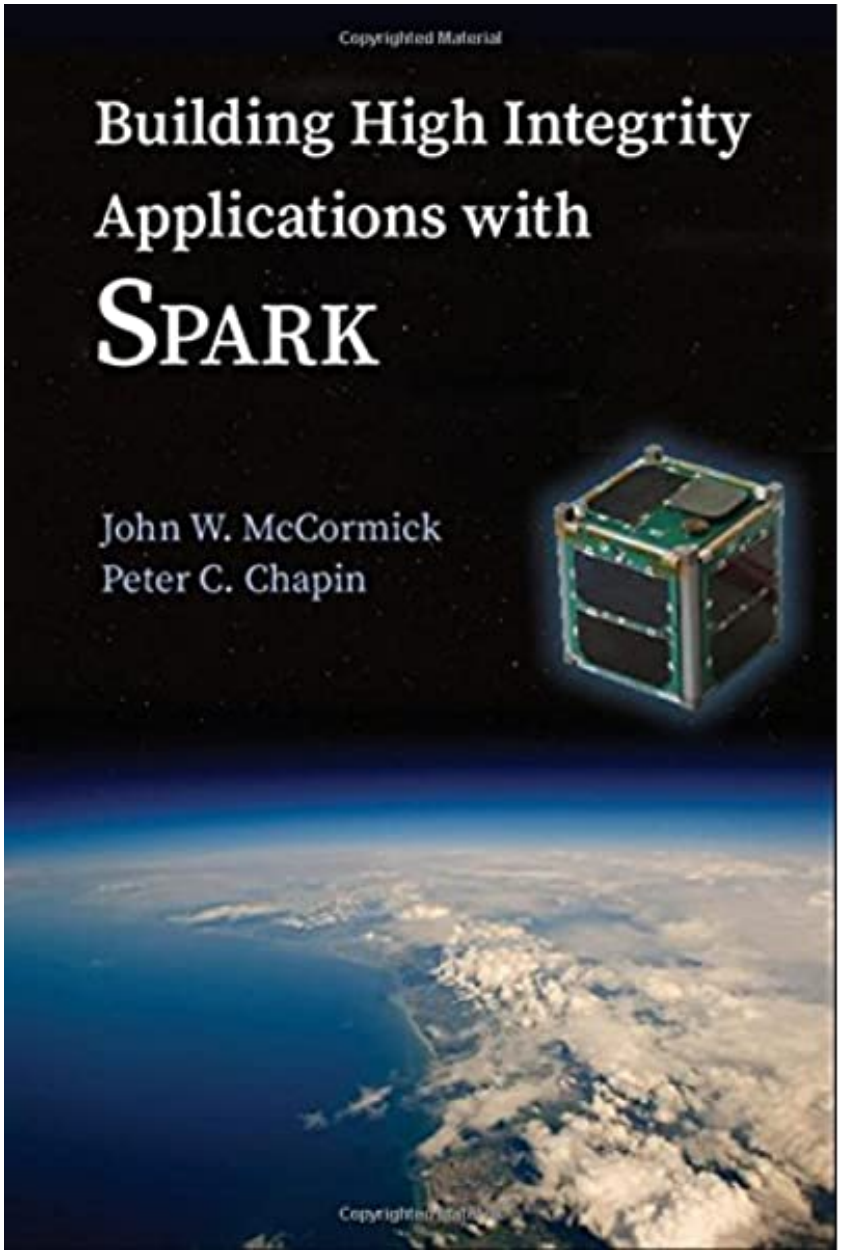
- Videos showing how to analyse and run the "Hello World" program are available via the Week 1 module on Canvas, i.e.
  - Video of "Hello World" in SPARK (Linux version)
  - Video of "Hello World" in SPARK (Windows 10 version)
- Note that in **GNAT Programming Studio**:
  - The same directory (i.e. folder) structure is used, i.e. `src` and `obj`.
  - However, the project file is automatically generated.

Summary

Learning outcomes:

- A brief history of Ada and its relationship with SPARK.
- How to access the SPARK toolkit and an Ada compiler.
- How to analyze, compile and execute SPARK code.

Summary



Recommended reading:

- "Building High Integrity Applications with SPARK" Mccormick, J.W. and Chapin, P.C. Cambridge University Press, 2015.
- SPARK 2014 User's Guide: `https://docs.adacore.com/spark2014-docs/html/ug/`
- AdaCore: SPARK Pro: `https://www.adacore.com/sparkpro`