### 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

▲□▶ ▲□▶ ▲□▶ ▲□▶ ■ ●の00

#### 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!

▲□▶ ▲□▶ ▲ □▶ ▲ □▶ □ のへぐ

#### 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

> > ▲□▶ ▲□▶ ▲□▶ ▲□▶ ■ ●の00

#### 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. Crown Copyright © 2016. Reused under the Open Government Licence **Eurofighter Typhoon:** one of the most advanced multirole combat aircraft

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

イロト イボト イヨト イヨト 三日

## Accessing Ada and the SPARK Analysis Tools (Edinburgh)



#### Linux:

- GNAT Studio is available in the School Linux Lab (EMB 2.50), i.e., open a Linux terminal window and type gnatstudio at the command line.
- Remote access via X2GO for details see:

https://www.macs.hw.ac.uk/cs/faq.html#Qnx

#### Windows:

- GNAT Studio is available in the University Windows Lab (EMB 2.52), i.e., available as a Windows App.
- Remote access via KeyServer for details see:

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

▲□▶ ▲□▶ ▲□▶ ▲□▶ ■ ●の00

### Down loading Ada and the SPARK Toolkit



#### x86 Windows (64 bits)

GNAT Community		
README.txt SHA-1:95ddc9742e5a57fef81e9a4196e2861613eb2842	3.4 KiB	May 27 2020
gnat-2020-20200818-x86_64-windows-bin.exe SHA-1: 85091aafe5cb946311bfa20a53b6aa0931bf9fc2	438 MiB	Aug 19 2020
ARM ELF (hosted on windows64)		
GNAT Community		
README.txt SHA-1: 95ddc9742e5a57fef81e9a4196e2661613eb2842	3.4 KiB	May 27 2020
gnat-2020-20200818-arm-elf-windows64-bin.exe SHA-1: 205015473cbfedz46848311c6e8673177bbe6512	134.1 MiB	Aug 19 2020

Use the **Community 2021** version, which you can download from https://www.adacore.com/download

▲□▶ ▲□▶ ▲□▶ ▲□▶ ■ ●の00

```
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;
```

▲□▶ ▲□▶ ▲□▶ ▲□▶ ■ ●の00

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

Welcome to GNAT Studio
GNAT STUDIO
Version Community 2021
+ Create new project
🚞 Open project
Start with default project

◆□ > ◆□ > ◆豆 > ◆豆 > ̄豆 = のへで

Creating a new project.



▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @

#### Select Create new project.



Location Deploy project in /home/air/doc-hw-h-drive/teach/rmse/spark2014/ Browse The location of the project to create. Settings Project Name default The name of the project. Main Name main The name of the main unit (no extension).				$\odot \odot \otimes$
Deploy project in /home/air/doc-hw-h-drive/teach/rmse/spark2014/ Browse The location of the project to create. Settings Project Name default The name of the project. Main Name main The name of the main unit (no extension).	Location			
The location of the project to create.  Settings  Project Name default  The name of the project.  Main Name main  The name of the main unit (no extension).	Deploy project ir	/home/air/doc-hw-h-drive	e/teach/rmse/spark2014/	Browse
SettIngs Project Name default The name of the project. Main Name main The name of the main unit (no extension).	The location of the p	roject to create.		
Project Name default The name of the project. Main Name main The name of the main unit (no extension).	Settings			
The name of the project.  Main Name The name of the main unit (no extension).	Project Name	default		
Main Name main The name of the main unit (no extension).	The name of the pro	iect.		
The name of the main unit (no extension).	Main Name	main		
	The name of the mai	n unit (no extension).		
			Cancel Back	Apply
Cancel Back Apply				

•	$\odot$ $\odot$ $\otimes$
Location	
Deploy project in	/home/air/doc-hw-h-drive/teach/rmse/spark201 //Hello Browse
The location of the pro	ject to create.
Settings	
Project Name	hello
The name of the proje	
Main Name	hello
The name of the main	unit (no extension).
	Main program name
	Viviant program name
	(no <b>.adb</b> extension)
	Cancel Back Apply



Note separate folders for source (src) and executable (obj) files.

▲□▶ ▲圖▶ ▲匡▶ ▲匡▶ ― 匡 … のへで



Note a skeleton main file is generated inside the src folder.



You can create your files within GNAT Studio or via your preferred editor and copy them to the **src** folder.



▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @



▲□▶ ▲□▶ ▲□▶ ▲□▶ □ のQ@



▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @



▲□▶ ▲□▶ ▲□▶ ▲□▶ □ のQ@

File	Edit Navigate Find Code VCS Build S         ■       ●	SPARK Analyze Debug Vlew Windo Examine All Examine All Sources Examine File	ello.adb
Project	✓ Hello ✓ src	Examine Subprogram	<pre>pragma SPARK_Mode (On); with Text_IO; use Text_IO;</pre>
scenario	nello.adb	Prove All           Prove All Sources           Prove File         Ctrl+Alt+F           Prove Subprogram         Ctrl+Alt+S           Prove Selected Region         Ctrl+Alt+R	<pre>procedure Hello is begin Put_Line("Hello WORLD!"); end Hello;</pre>
Switches		Prove Line Ctri+Alt+L Exit Manual Proof Show Report Show Log Clean Proofs	

Select **Show Log** option from the **SPARK** menu to see a summary of the static analysis.

			areato - Americanova - Thometeritano - manana manana ana waxa thenatan Manahari a terio ha
File	Edit Navigate Find Code VCS Build SPARK Analyze Debug	View W	Window Help
9	≌ 😫   ⊖ ⊝ Aa   🕈 ⇒   ♥ 🕹 🖾 🖄 🛓 ▶ 🛞		
~	0 🖌 🗟 🗮 🔍 filter		🕐 hello.adb 📄 gnatprove.out
rojec	- Hello		1 Summary of SPARK analysis
۵.	hello.adb		3
arlo	le obj		4 5 SPARK Analysis results Total Flow CodePeer Provers Justified Unproved
Scen			6 7 Data Dependencies
			8 Flow Dependencies
hes			9 Initialization
witc		1	11 Run-time Checks
ŝ		1	12 Assertions
		1	14 LSP Verification
			15 Termination
		1	17
			18 Total
		2	20
		2	22 max steps used for successful proof: 0
		2	23 Analyzed 1 unit
			24 in unit netto, i supprograms and packages out of 1 analyzed 25 Hello at hello.adb:4 flow analyzed (0 errors, 0 checks and 0 warnings)
-		=	
Learr	Actions		

◆□▶ ◆□▶ ◆ 臣▶ ◆ 臣▶ ○ 臣 ○ の Q @

The Summary of SPARK analysis is shown above.

File	Edit Navigate Find Code VCS	Build SPARK Analyze De	ebua Vie	w Window Help		
a		Check Syntax	5		hello.	adb
ť	0 / 🔄 📕 Q-filter	Compile File S	hift+F4	hello.adb	gnatprove.out	
roje	- Hello	Project		Build & Run	hello.adb	
٩	bello.adb	Clean	÷	Build & Debug		
0	o obj	Run	•	hello.adb	F4	
Scenario		Settings	•	Build All Compile All Sources	Build Main hello.adb Action: Build Main Number 1	F
Switches				Build <current file=""> Custom Build 10 muintailas 11 Ruintime 12 Assertion 13 Functiona 14 LSP Verif 15 Terminati 16 Concurren 17 18 Total 19 20 21 max steps 22 22 Applied</current>	Action: Build Main Mumber 1 Category: Build Shortcut: F4 Menu: Build/Project/hello.adb Checks is 1 Contracts ication on cy used for successful proof	F: 0

The **Project**  $\Rightarrow$  **Build & Run** option from the **Build** menu complies and runs your designated main file.



▲□▶ ▲□▶ ▲□▶ ▲□▶ □ のQ@

#### Hello WORLD!

Welcome to GNAT Studio
GNAT STUDIO
Version Community 2021
+ Create new project
Dpen project
Start with default project

◆□ > ◆□ > ◆豆 > ◆豆 > ̄豆 = のへで

Openning an existing project.



▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @

Select Open project.



Version Community 2021





#### Additional Guidance

- A video showing how to create, analyse, compile and run the "Hello World" program using GNAT Studio on Windows is available via the Week 1 module on the course's Canvas pages.
- An online tutorial for GNAT Studio is available via: https://docs.adacore.com/live/wave/gps/html/gps\_ tutorial/

### Additional Guidance

# Building High Integrity Applications with **SPARK**

Copyrighted Material

John W. McCormick Peter C. Chapin



- "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

#### 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.