Safe and Secure Software

An Invitation to Ada 2005

Foreword / Contents / Introduction / Bibliography

Courtesy of AdaCore
The GNAT Pro Company

John Barnes
Foreword

The aim of this booklet is to show how the study of Ada in general and Ada 2005 in particular, is helpful to everyone designing safe and secure software regardless of the programming language in which the software is eventually written. After all, successful implementers of safe and secure software write in the spirit of Ada in any language!

Thank you John for showing this throughout your papers, rationales, books, and this booklet.

AdaCore dedicates this booklet to all the designers and implementers of safe and secure software.
Contents

Introduction 1

1 Safe Syntax 3
   Equality and assignment 3
   Statement groups 5
   Named notation 6

2 Safe Typing 9
   Using distinct types 9
   Enumerations and integers 11
   Constraints and subtypes 13
   Arrays and constraints 14
   Real errors 17

3 Safe Pointers 19
   References, pointers and addresses 19
   Access types and strong typing 21
   Access types and accessibility 23
   References to subprograms 24
   Nested subprograms as parameters 26

4 Safe Architecture 31
   Package specifications and bodies 31
   Private types 35
   Generic contract model 37
   Child units 38
   Unit testing 39
   Mutually dependent types 40

5 Safe Object-Oriented Programming 43
   Object-Orientation versus Function-Orientation 43
   Overriding indicators 47
6 Safe Object Construction
   Variables and constants 55
   Constant and variable views 57
   Limited types 58
   Controlled types 61
7 Safe Memory Management 65
   Buffer overflow 65
   Heap control 66
   Storage pools 69
   Restrictions 73
8 Safe Startup 75
   Elaboration 75
   Elaboration pragmas 77
   Dynamic loading 78
9 Safe Communication 81
   Representation of data 81
   Validity of data 83
   Communication with other languages 84
   Streams 85
   Object factories 87
10 Safe Concurrency 91
   Operating systems and tasks 91
   Protected objects 93
   The rendezvous 98
   Restrictions 101
   Ravenscar 102
   Timing and scheduling 102
## Contents

11 Certified Safe with SPARK 105

- Contracts 105
- Correctness by construction 106
- The kernel language 109
- Tool support 110
- Examples 112
- Certification 113

Conclusion 115

Bibliography 119
Introduction

The aim of this booklet is to show how Ada 2005 addresses the needs of designers and implementers of safe and secure software. The discussion will also show that those aspects of Ada that make it ideal for safety-critical and security-critical application areas will also simplify the development of robust and reliable software in many other areas.

The world is becoming more and more concerned about both safety and security. Moreover, software now pervades all aspects of the workings of society. Accordingly, it is important that software which is concerned with systems for which safety or security are a major concern should be safe and secure.

There has been a long tradition of concern for safety going back to the development of railroad signaling and more recently with aviation. Vital software systems such as those that control aircraft navigation and landing have to meet well established certification and validation criteria.

More recently there has been growing concern with security in systems such as banking and communications generally. This has been heightened with concern for the activities of terrorists.

Safety and security are intertwined through communication. An interesting characterization of the difference is

- safety – the software must not harm the world,
- security – the world must not harm the software.

So a safety-critical system is one in which the program must be correct, otherwise it might wrongly change some external device such as an aircraft flap or a railroad signal, with serious real-world consequences.

And a security-critical system is one in which it must not be possible for some incorrect or malicious input from the outside to violate the integrity of the system, for example by corrupting a password checking mechanism and stealing social security information.

The key to guarding against both problems is that the software must be correct in the aspects affecting the system's integrity. And by correct we mean that it meets its specification. Of course if the specification is incomplete or itself incorrect then the system will be vulnerable. Capturing requirements correctly is a hard problem and is the focus of much attention from the lean software development community.

One of the trends of the second half of the twentieth century was a universal concern with freedom. But there are two aspects of freedom. The ability of the
individual to do whatever they want conflicts with the right to be protected from the actions of others. Maybe A would like the freedom to smoke in a pub whereas B wants freedom from smoke in a pub. Concern with health in this example is changing the balance between these freedoms. Maybe the twenty-first century will see further shifts from "freedom to" to "freedom from".

In terms of software, the languages Ada and C have very different attitudes to freedom. Ada introduces restrictions and checks, with the goal of providing freedom from errors. On the other hand C gives the programmer more freedom, making it easier to make errors.

One of the historical guidelines in C was "trust the programmer". This would be fine were it not for the fact that programmers, like all humans, are frail and fallible beings. Experience shows that whatever techniques are used it is hard to write "correct" software. It is good advice therefore to use tools that can help by finding bugs and preventing bugs. Ada was specifically designed to help in this respect. There have been three versions of Ada – Ada 83, Ada 95 and now Ada 2005.

The purpose of this booklet is to illustrate the ways in which Ada 2005 can help in the construction of reliable software, by illustrating some aspects of its features. It is hoped that it will be of interest to programmers and managers at all levels.

It must be stressed that the discussion is not complete. Each chapter selects a particular topic under the banner of Safe X where Safe is just a brief token to designate both safety and security. For the most critical software, use of the related SPARK language appears to be very beneficial, and this is outlined in Chapter 11.

A topic with which Ada has much synergy is lean software development – there is not enough space in this booklet to expand on this concept but the reader is encouraged to explore its good ideas elsewhere.

As the twenty-first century progresses we will see software becoming even more pervasive. It would be nice to think that software in automobiles for example was developed with the same care as that in airplanes. But that is not so. My wife recently had an experience where her car displayed two warning icons. One said "stop at once", the other said "drive immediately to your dealer". Another anecdotal motor story is that of a driver attempting to select channel 5 on the radio, only to see the car change into 5th gear! Luckily he did not try Replay.

For a fuller description of Ada 2005, SPARK, and lean software development and papers on related topics please consult the bibliography.
The following two books are comprehensive descriptions of Ada 2005 and SPARK respectively. Both contain CDs with appropriate supporting material.


The following award-winning book is a good introduction to lean software development.


The following websites provide access to much useful information.

- www.ada-europe.org – for Ada-Europe, conferences and journal.
- www.sparkada.com – for SPARK.

The following further documents and books are referenced in the text.


