

AdaCore
The GNAT Pro Company



Tokeneer: Beyond Formal Program Verification

Yannick Moy and Angela Wallenburg

June 21, 2010

The Tokeneer Project

The Tokeneer Challenge

The Tokeneer Lessons

The Tokeneer Project

The Tokeneer Challenge

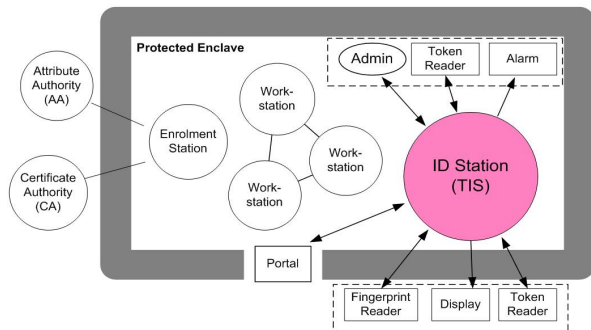
The Tokeneer Lessons

Project goals to demonstrate:

- ▶ Common Criteria **EAL5** achievable in a **cost effective** manner
- ▶ Praxis “**Correctness by Construction**” approach
 - ▶ sound, formal notations
 - ▶ keep it simple
 - ▶ remove errors as soon as possible
 - ▶ generate evidence as you go

Tokeneer originally developed by NSA

What is Tokeneer?



- ▶ Protect secure information in physically secure enclave
- ▶ Demonstrate use of Smart Cards and Biometrics

Metrics – Development Statistics

	Ada LOC	SPARK LOC	LOC/day coding	LOC/day overall
core	9,939	16,564	203	38
support	3,697	2,240	182	88

- ▶ Total effort: 260 man days
- ▶ Team: 3 people part-time
- ▶ Total schedule: 9 months elapsed

Metrics – Post Delivery Defects

- ▶ software defects found in independent test: **zero**
- ▶ by NSA prior to public release (2004-2008): **zero**

Tokeneer Public Release (2008)

- ▶ NSA “Technology Transfer Agreement”
- ▶ essentially open-source, entire project archive
- ▶ Google for “Tokeneer” or
- ▶ <http://www.adacore.com/tokeneer>

The Tokeneer Project

The Tokeneer Challenge

The Tokeneer Lessons

Finding Bugs in Tokeneer!

Who	How	Where	#
Rod Chapman	SPARK tools	code	1
Diomidis Spinellis	code review	code	1
Woodcock & Aydal	Alloy analyzer	specification	12
Moy & Wallenburg	code review	code	11
Moy & Wallenburg	static analysis	code	9

Absence of run-time errors, by applying the SPARK tools

```
| 4 | start | rtc check @ 234 | Undischarged
```

```
233 if Success and then  
234   (RawDuration * 10 <= Integer(DurationT ' Last) and  
235    RawDuration * 10 >= Integer(DurationT ' First))  
236 then  
237   Value := DurationT(RawDuration * 10);  
238 else
```

- ▶ RawDuration is very large \Rightarrow overflow during multiplication
- ▶ Solution is to divide instead

Security code review, focusing on error handling

```
534 File.Delete(TheFile => TheFile ,  
535           Success => OK);  
536 AuditSystemFault := AuditSystemFault and not OK;
```

- ▶ File successfully deleted \Rightarrow AuditSystemFault is cleared
- ▶ Deletion fails \Rightarrow AuditSystemFault not always set

Static Analysis Setup

Tool	Category	Who	Run-Time	Review Time
GNAT	compiler	AdaCore	3s	10mn
CodePeer	analyzer	AdaCore & SofCheck	> 1mn < 18mn	1 day
Examiner	analyzer	Praxis	10s	5s
Simplifier	prover	Praxis	30s	5s
Dead Path Analyzer	prover	Praxis	4mn	1 day

Code Review Setup

Blind Spot	# in SPARK	# in Ada
hide annotations	29	-
loop termination	6	17
exception handling	-	101

Review time: 1 day

Categories of Problems

Defect:

may lead to an observable failure of the system

Example: sorting procedure does not sort

Code quality issue:

cannot lead to an observable failure of the system

could lead to an observable system failure during maintenance

Example: sorting procedure is called `Remove_Duplicates`

Disclaimer: Experimental Bias

Not a quantitative comparison of tools and methods

- ▶ During development, code was cleared by GNAT, Examiner and Simplifier
- ▶ During development, various reviews were performed
- ▶ First application of CodePeer, Dead Path Analyzer and focused reviews

Not a quantitative comparison between SPARK and Ada code

- ▶ SPARK tools only applicable to SPARK code
- ▶ Review of static analysis results focused on SPARK code
- ▶ Code review focused on SPARK code or its interface to Ada code

Defects and Issues Found in Tokeneer

	# SPARK defects	# SPARK issues	# Ada defects	# Ada issues	# total (# exclusive)
GNAT	1	1		1	3 (0)
CodePeer	1	4	3	1	9 (6)
SPARK tools	2		-	-	2 (1)
static analysis	2	4	3	1	10
hide annotations			-	-	0
loops	1	1	1	1	4
exceptions	-	-	3	1	4
other	1	2		1	4
code review	2	3	4	3	12
total	4	7	7	4	22

The Tokeneer Project

The Tokeneer Challenge

The Tokeneer Lessons

Defects Found in SPARK Code

Only one run-time error

- ▶ Found by a new version of the SPARK toolset

One possible cause of non-termination

- ▶ Blind spot of the SPARK verifications
- ▶ Best handled by a methodology for code review for SPARK

Two defects related to error handling

- ▶ Exceptions are a better way to signal errors
- ▶ Best handled by a methodology for code review for SPARK

One functional error

- ▶ Side-effect of detecting logic errors in code

Defects Found in Ada Code

Many serious errors

- ▶ Three serious run-time errors (by static analysis)
- ▶ Four serious functional errors (by code review)

Most errors cannot occur in SPARK

- ▶ Prevented by the language: ignoring exceptions
- ▶ Examiner: uninitialized variable, ignoring error status
- ▶ Simplifier: access out of array bounds, accessing the null string

Remaining errors

- ▶ Low-level manipulations (string) are best verified
- ▶ Exceptions are a better way to signal errors

High-Level Lessons Learned

Need for dynamic code coverage

- ▶ Most issues in SPARK are due to lack of coverage
- ▶ Static detection of dead code is typically incomplete
- ▶ Need for executable annotations

Tool setup and configuration

- ▶ One issue in SPARK was missed by not using appropriate compiler switch
- ▶ Need to be reviewed even more than code

Tools should facilitate review

- ▶ Most time spent in static analysis: review of results

One language of executable annotations

Combine tests, static analysis and proofs

Sound static analysis applicable to DO-178C

Separate verification for scaling and early adoption

Free software structured as toolchains for Ada and C

HI-LITE

<http://www.open-do.org/projects/hi-lite/>

Defect in SPARK: Wrong Variable Used

GNAT warns that a condition in a test is known to be true

```
keystore.adb:349:23: warning:  
condition is always True (see test at line 344)
```

```
344 if RetVallni = Interface.Ok then  
345   Interface.FindObjects  
346     (HandleCount => HandleCount ,  
347     ObjectHandles => Handles ,  
348     ReturnValue => RetValDo);  
349   if RetVallni = Interface.Ok then
```

- ▶ Functional error detected as a side-effect of logic error in code
- ▶ Could be detected by dynamic branch coverage
- ▶ Detected by three tools: GNAT compiler, CodePeer analyzer, SPARK Dead Path Analyzer

Issue in SPARK: Redundant Conjunct in Precondition

CodePeer warns that a condition in an annotation is always true

```
enclave.adb:1107:13: warning: condition is always true
```

```
1107 —# (Admin.prf_rolePresent(TheAdmin) = Typ.Guard
1108 —#  ->
1109 —# ((Admin.IsDoingOp(TheAdmin) and
1110 —#   Admin.TheCurrentOp(TheAdmin) = OverrideLock)
1111 —#   or not Admin.IsDoingOp(TheAdmin)));
```

- ▶ Benefit of sharing annotations between tools
- ▶ Could be detected by dynamic condition coverage on executable annotations
- ▶ Need for abstraction in contracts

Defect in Ada: Access Out of Array Bounds

CodePeer warns that an array access might be out of bounds

```
tcpip.adb:205:16: medium:  
array index check might fail: requires i >= 2
```

```
204 if Msg.Data(i) = ASCII.Lf and then  
205   Msg.Data(i - 1) = ASCII.Cr then
```

- ▶ Class of errors that are automatically detected in SPARK code
- ▶ Hard to detect by testing (except concolic testing or fuzzing)

GNAT warns that a local variable is assigned but never used

```
tcpip.adb:446:07: warning:  
variable "Address" is assigned but never read
```

- ▶ Class of errors that are automatically detected in SPARK code
- ▶ Could not be detected by dynamic coverage

Defect in SPARK: Possible Non-termination

A loop may not terminate if a function called terminates in error

```
149 while Stop=0 and not File.EndOfFile(TheFile) loop  
150   — Read the next (non-empty) line of the file  
151   —# assert PrivateKeyPresent(KeyStore.State) =  
152   —#       PrivateKeyPresent(KeyStore.State ~);  
153   File.GetLine(TheFile, TheCert, Stop);  
154 end loop;
```

- ▶ Exceptions are a better way to signal errors
- ▶ Need adequate methodology for integrating proof and code review for exception handling

A postcondition is trivially true

```
174 —# post ((Op = OverrideLock <->  
175 —#         prf_rolePresent(TheAdmin~) = Typ.Guard)  
176 —#         ->  
177 —#         (Op = OverrideLock <->  
178 —#         prf_rolePresent(TheAdmin) = Typ.Guard))
```

- ▶ Logic errors best detected by static analysis tools (GNAT, CodePeer)
- ▶ Benefit of sharing annotations between tools
- ▶ Need for executable annotations

Defect in Ada: Error Code not Set

A procedure can return in error while setting a Success flag to True

```
283     Success := True;  
284     ...  
285     — some code that can raise an exception  
286     ...  
287 exception  
288     when E : others =>  
289         DebugOutput("Send Error.");
```

- ▶ Exceptions are a better way to signal errors

An access can raise an exception which is caught by a local handler

```
161 — Found the Key string— is it enclosed by quotes?  
162 if LocalMsg(KeyStart - 1) = '' and  
163     LocalMsg(KeyStart + Key'Length) = '' then
```

- ▶ Class of errors that are automatically detected in SPARK code
- ▶ Hard to detect by testing (except concolic testing or fuzzing)