Recent results with Correctness by Construction and SPARK

Rod Chapman

Praxis High Integrity Systems

Contents

- What are C-by-C and SPARK?
- Projects and Results
- SPARK technical update
- C-by-C and the SEI PSP/TSP

Contents

- What are C-by-C and SPARK?
- Projects and Results
- SPARK technical update…
- C-by-C and the SEI PSP/TSP

What is Correctness by Construction

- A systems and software development approach.
- Key principles:
  - Make it hard to introduce defects in the first place.
  - Detect and correct defects as soon as possible after their introduction

- Easy huh? Easier said than done…

Correctness by Construction(2)

- Observation
  - We can't rely on testing alone as the primary verification activity - much too expensive and risk prone.
  - Also, for the most critical systems, testing can **never** generate sufficient evidence.
- So what else can we do?

Correctness by Construction(3)

- Therefore, C by C is a design approach characterized by:
  - Use of static verification to **prevent** defects at all stages.
  - Small, verifiable design steps.
  - Appropriate use of formality (aka “Maths”).
  - “Right tools and notations for the job” approach.
  - Generation of certification/evaluation evidence as a side-effect of the development process. E.g. for a safety-case.

So what's SPARK?

- SPARK embodies the principles of C by C in a programming language and verification system.

- Languages *really do* matter.
  - They affect the way we think about the world, the problem we're solving etc. etc.

The Catch...

- Our ability to perform static verification critically depends on the language or notation under analysis.
- In particular, **ambiguity** in the definition of the language severely limits what is achievable.
- Ideally, languages and notations should be as unambiguous as possible.

Ambiguity in Computing Languages

- This idea is not new…

*"… one could communicate with these machines in any language provided it was an **exact** language …"*

*"… the system should resemble normal mathematical procedure closely, but at the same time should be as **unambiguous** as possible."*

Ambiguity in Computing Languages

- This idea is not new…

*"… one could communicate with these machines in any language provided it was an **exact** language …"*

*"… the system should resemble normal mathematical procedure closely, but at the same time should be as **unambiguous** as possible."*

Alan Turing (1948)

Ambiguity in Software Engineering

- Unfortunately, ambiguity plagues us at every turn:
  - English requirements
  - UML and other “OO” notations
  - Programming languages
    - Does anyone understand C++ Templates?!?
- Machine code is often the first unambiguous representation we get, which can be tested but not much else…oh dear…

Programming Languages...

- Standard languages? C, C++, Java?
  - All fall down on ambiguity and therefore verifiability.
  - "Modern" language design is going the **wrong way!** E.g. OO polymorphism, exceptions etc.
- Special purpose languages?
  - Ever heard of "NewSpeak"? Nope...

Programming Languages...

- High-Integrity Language subsets?
  - Potentially combine the best of both worlds: desirable properties for H-I, using standard compilers, tools, staff etc.
  - Integrity achievable critically depends on selection of base language.
  - For the highest integrity levels, subsetting alone may not be enough. Addition of **annotations** to strengthen the language ("design by contract"™) may be required.

So…What is SPARK?

- The “SPADE Ada Kernel”
  - What does the “R” stand for?
- A sub-language of Ada95 with particular properties that make it ideally suited to the most critical of applications:
  - Completely unambiguous
  - All rule violations are detectable
  - Formally defined
  - Tool supported
- SPARK facilitates Correctness by Construction

SPARK Features

- SPARK is **statically free** from all
  - Aliasing
  - Function side-effects
  - Erroneous behaviour
  - Implementation-dependent behaviour

- These analyses are all decidable in polynomial time. i.e. tool is very fast! This enables **constructive** use.

Static Analysis of SPARK

- The Examiner tool implements a number of analyses, again all in P-Time:
  - Subset checking and static semantics
  - Information flow analysis
  - Verification Condition Generation - allows proof of properties such as exception freedom, partial correctness, and safety properties.
- Theorem prover tool (the Simplifier) does a good job of proving VCs.

Contents

- What are C-by-C and SPARK?
- Projects and Results
- SPARK technical update…
- C-by-C and the SEI PSP/TSP

C-by-C Projects

- CDIS - Critical ATC System (London Airport!)
- SHOLIS - Naval Ship/Helicopter Information System. First ever Def Stan 00-56 "SIL4" project.
- MULTOS CA - ITSEC E6 (=CC EAL7) secure certification authority.
- A - Naval stores management system.
- B - Biometric access control system. CC EAL5 and above demonstrator project funded by a government agency.

C-by-C - what's a "Defect" anyway?

- A "Defect" is *any* error in a design artefact once placed under change control or delivered to a client, including documents, designs, manuals etc. as well as code.
  - Expected behaviour is defined by the (formal) system specification.
- CDIS, SHOLIS and MULTOA CA were delivered with a *Warranty.*
- During the warranty period, *we* fix Defects at no charge.

- (Yes..we are still in business...)

C-by-C Projects

Project	Year	Size (loc)	Productivity (loc/day)	Defects (per kloc)
CDIS	1992	197000	12.77	0.75
SHOLIS	1997	27000	7.0	0.22
MULTOS CA	1999	100000	28.0	0.04
A	2001	39000	11.0	0.05
B	2003	10000	38.0	0.0

Results…

- C-by-C productivity and defect rate is as good or better than data reported for TSP, CMM 5, CleanRoom.
- C-by-C provides the means to also meet the most stringent regulatory requirements and standards *without* undue additional pain and/or expense.
- We find that *better can be cheaper* - ultra-reliable does *not* mean ultra-expensive!

Contents

- What are C-by-C and SPARK?
- Projects and Results
- SPARK technical update…
- C-by-C and the SEI PSP/TSP

SPARK Technical Update

- Many, Many things…far too many to mention in detail:
  - RavenSPARK
  - SPARK Academic Package
  - Tool integrations and partners
  - The new SPARK book
  - New "Black Belt SPARK" Course
  - Security
  - Exception Freedom and Theorem Proving improvement

RavenSPARK

- Tasking is back!
- Brings a subset of the Ada95 Ravenscar Profile directly into the core of SPARK.
- Deterministic scheduling scheme, suitable for hard-real time schedulability analysis
- RavenSPARK eliminates potential errors and much more…

SPARK Academic Package

- *Full* Professional SPARK Toolset is free-of-charge to academic faculty.
- *Full* support service to faculty member (but not to 50 students… :-))
- We have joined AdaCore's Ada Academic Initiative.
- Universities teaching SPARK right now: Manchester, York, Virginia, Northern Iowa, Oakland, James Madison, Idaho, Roger Williams…
- SPARK a big hit at ACM SIGCSE 2004/5.

Tool Integrations and Partners

- Tool vendors now supporting SPARK…
  - ARTiSAN Real-Time Studio
  - Ilogix Rhapsody-in-Ada
  - ADI Beacon
  - High Integrity Solutions VDS
  - Plus significant support from compiler vendors
  - More to come…
- Marketing/Sales/Training partnership in the USA with Pyrrhus Software

The new SPARK Book

- High Integrity Software: the SPARK Approach to Safety and Security by John Barnes and Praxis
- Published in April 2003.
- Good reviews on SlashDot, Amazon, comp.risks, ACM Computing Surveys.
- Has generated much "buzz"

"Black Belt SPARK" course

- New, advanced course for those with experience of using SPARK.
- Focus on how to make best use of the proof facilities, and in particular the proof of the absence of exceptions.
- "Proof Directed Software Design" - how do you write *provable* code?
- Next courses:
  - Next week!
  - September 2005

Security

- The security community are taking high-integrity software very seriously.
- US SEI/DHS report on software development for secure systems
  - Only 3 processes identified that can deliver fewer than 0.1 defects per kloc: TSP, IBM CleanRoom and Praxis CbyC.
- Even Microsoft Research have noticed and recognized SPARK (!)

Exception Freedom and Theorem Proving

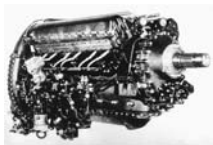
- A SPARK program can be shown to be free of all "predefined exceptions"
  - e.g. buffer overflow, division by zero, range violation etc.
- We do this by generating small **conjectures** from a program, the proof of which show that the exception could **never** occur,
  - Good news - Proof process is automated by the **Simplifier** - a theorem prover.

Turning the dials up...

- Three ways to improve performance...

– Smarter VC Generation and Theorem Proving Tactics

– Streamline existing tactics and algorithms



– Get a bigger engine...

The Test Data...

- "Project A"
  - Embedded, real-time stores-management system
  - Some functions are SIL3
  - 39000 declarations and statements

The Test Data...

- "Project A" Verification Conditions

VC Class	
Assertion or Postcondition	7142
Precondition	69
Exception freedom	10890
Refinement	554
Inheritance	0
Total	18655

Test Conditions

- Which combination of Examiner/Simplifier/Hardware to use?
- Principle: use tools and hardware that's available to users.
  - Commercial releases of tools
  - Commodity PC hardware
- Measure:
  - Execution time of tools
  - Simplifier "hit rate"
  - Number of VCs left to be reviewed or proven manually.

Performance data

Toolset	Hardware	Time/mins	Hit rate %	VCs left
6.3 (Dec 2002)	1.8GHz P4 Mobile	111	94.5	1025
7.0	1.8GHz P4 Mobile	109	94.69	990
7.0	2.4 GHz P4 Xeon	73	94.69	990
7.1	2 * 2.4 GHz P4 Xeon	49	95.75	791
7.2 (Jan 2005)	2 * 2.4 GHz P4 Xeon	82	97.24	515

Contents

- What are C-by-C and SPARK?
- Projects and Results
- SPARK technical update…
- C-by-C and the SEI PSP/TSP

SEI, PSP and SPARK…

- SEI have discovered Correctness-by-Construction and SPARK…
- Praxis have discovered PSP/TSP.
- SPARK projects can deliver <= 0.1 defects per kloc.
- So can PSP/TSP projects…
- What happens if you put the two together?

SEI, PSP and SPARK...

- PSP emphasis on personal practice.
  - Measurement of performance
  - Statistical analysis of data
  - Use of data to aim future planning.
  - BUT – technology neutral...
- C by C takes a strong technical stance
  - Well-defined languages
  - Strong static verification

SEI, PSP and SPARK...

- Rod has taken PSP for Engineers course, using SPARK for all programming exercises.
  - Metrics look good.
- Next:
  - PSP Instructor training.
  - SEI will be trained in SPARK in September.
- Try it on a project!

Conclusions

- SPARK continues to grow in size, maturity and use.
- A marked change in attitude has been observed:
  - Tool vendors are coming to see us…
  - People have read about SPARK and are interested enough to come and find us at shows…
  - The book…
  - The security community…