Industrial Experience with SPARK

Roderick Chapman
Praxis Critical Systems
Outline

• Introduction
• SHOLIS
• The MULTOS CA
• Lockheed C130J
• A less successful project…
• Conclusions
Introduction

• Most Ada people know what SPARK is.

• But that’s only half the story - there are far more interesting questions!
  – Who’s using SPARK?
  – What factors separate successful from unsuccessful SPARK projects?
  – How does SPARK help to meet the various industry standards?
Introduction (2)

• Four very different projects

• SHOLIS - Def. Stan. 00-55, Naval Aviation.
• MULTOS CA - ITSEC E6, High-Security Finance.
• Lockheed C130J - DO-178B, Commercial/Military Aviation.
• A “less successful” project.
Fault Tolerant Real-Time Embedded System: SHOLIS
Ship Helicopter Operating Limits Instrumentation System

- System for UK MOD to improve safety information on take off and landing
- PMES Ltd prime (system) contractor
- Praxis Critical Systems software developer
- First full application of (interim) Defence Standard 00-55
- Major functions classed as ‘SIL4’ (safety critical)
SHOLIS on test...
Defence Standard 00-55: Example Requirements

- **Formal specification**
  - must to possible to verify formally that S/W meets spec

- **Formal design**
  - small on SHOLIS: most code proved directly against spec

- **Proof obligations at all stages**
  - e.g. formal design against spec, code against formal design

- **Resource modelling**
  - to show S/W meets spec of available hardware resources

- **Static analysis of code**
  - including: control, data and information flow analysis
The use of Proof in SHOLIS

• Proof was used on both the Z and Code.

• For the code, we proved:
  – Freedom from predefined exceptions (all code).
  – Partial correctness (SIL4 units).
  – Safety Properties (at the “main loop” level).
Proof Metrics

- In terms of faults found per day, Z and code proof were two of the most cost-effective verification activities.
- Both were significantly more cost-effective than traditional unit testing.
- Code proof discharged over 9000 verification conditions - probably the largest such effort at the time. (1997)
Observations on Proof

• Having to go to all the trouble of doing proof, means you really have to think about the code.
  – Correctness emerges as a side effect!

• Proofs which “don’t come out easily” often indicate faults in design or specification, which can be corrected earlier than if they had been found in system test.

• Machine resources available now make “regression proof” possible.
The MULTOS CA

• MULTOS is a multi-application operating system for smart cards.
• Applications can be loaded and deleted dynamically once a card is “in the field.”
• To prevent forging, applications and card-enablement data are *signed* by the MULTOS Certification Authority (CA).
• At the heart of the CA is a high-security computer system that issues these certificates.
The MULTOS CA (2)

• The CA has some unusual requirements:
  – Availability - aimed for c. 6 months between reboots, and has warm-standby fault-tolerance.
  – Throughput - system is distributed and has custom cryptographic hardware.
  – Lifetime - of decades, and must be supported for that long.
  – Security - most of system is tamper-proof, and is subject to the most stringent physical and procedural security.
  – Was designed to meet the requirements of U.K. ITSEC E6.

• All requirements, design, implementation, and (on-going) support by Praxis Critical Systems.
Use of languages in the CA

• Mixed language development - the right tools for the right job!
  – SPARK 30% “Security kernel” of tamper-proof software
  – Ada95 30% Infrastructure (concurrency, inter-task and inter-process communications, database interfaces etc.), bindings to ODBC and Win32
  – C++ 30% GUI (Microsoft Foundation Classes)
  – C 5% Device drivers, cryptographic algorithms
  – SQL 5% Database stored procedures
Use of SPARK in the MULTOS CA

• SPARK is almost certainly the only industrial-strength language that meets the requirements of ITSEC E6.
• Complete implementation in SPARK was simply impractical.
• Use of Ada95 is "Ravenscar-like" - simple, static allocation of memory and tasks.
• Dangerous, or new language features avoided such as controlled types, requeue, user-defined storage pools etc.
Lockheed C130J
Lockheed C130J Mission Computer

- 130,000 lines of safety related code in mission computer
- Process designed to
  - reduce V&V costs (and consequent delays)
  - meet certification requirements, UK MoD and FAA
- Based on rigorous specification and design
  - CoRE (Parnas tables)
  - SPARK
Meeting DO-178B - some observations

• DO-178B level A places great emphasis on evidence of testing, and in particular, the use of target-based coverage analysis.

• Such data can only be collected on real target hardware, the availability of which is often limited.

• Target-based testing is difficult, boring, and time-consuming.

• Debugging on a target rig is difficult (especially when programs may be erroneous).
Meeting DO-178B (2)

- Coverage analysis in the presence of predefined exceptions is nearly impossible - many paths, most of which you hope are dead, so how do you get coverage?
- Most project compile “with checks off” to simplify object code and make coverage simpler.
- Relies on confidence gained in testing and review that no real exceptions are “lurking.”
Meeting DO-178B - The SPARK Approach

- Using SPARK, we can:
- Statically eliminate erroneousness (e.g. dataflow errors), so such hard-to-find faults simply cannot reach integration or test.
- Statically prove, *for all input data*, that predefined exceptions cannot occur - much stronger than testing and/or review.
- *Justifiably* turn off checks.
- This *has* to be cheaper!
Lockheed on SPARK...

• Some errors immediately uncovered by formal analysis, such as conditional initialization errors may only emerge after very extensive testing.
• The technology for generating and discharging the proof obligations, based on the SPARK components of Ada, was crucial, in binding the code to the initial requirements.
• SPARK provides an extremely robust and efficient basis for formal verification.
• The process has proven effective with typical software developers and did not necessitate and inordinate amount of additional training.
• Experience has shown that SPARK coding occurs at near typical Ada rates.
• Code written in SPARK is deterministic and inherently statically analysable.
• Very few errors have been found in the software during even the most rigorous levels of FAA testing, which is being successfully conducted for less than a fifth of the normal cost in industry.
• Correctness by construction is no longer a theoretical abstraction; it is now a practical way to develop software that exceeds its technical goals while delivering sterling business performance.
Lockheed on SPARK...

- Some errors immediately uncovered by formal analysis, such as conditional initialization errors may only emerge after very extensive testing.
- The technology for generating and discharging the proof obligations, based on the SPARK components of Ada, was crucial, in binding the code to the initial requirements.
- SPARK provides an extremely robust and efficient basis for formal verification.
- The process has proven effective with typical software developers and did not necessitate and inordinate amount of additional training.
- Experience has shown that SPARK coding occurs at near typical Ada rates.
- Code written in SPARK is deterministic and inherently statically analysable.
- Very few errors have been found in the software during even the most rigorous levels of FAA testing, which is being successfully conducted for less than a fifth of the normal cost in industry.

Correctness by construction is no longer a theoretical abstraction; it is now a practical way to develop software that exceeds its technical goals while delivering sterling business performance.
More on Lockheed C130J

- Workshop - not to be missed!

  “Cost-Effective Approaches to Satisfy Safety critical Regulatory Requirements.”

- Friday, 17 November, 9:00 AM - 12:00 Noon
- Organizer: Jim Sutton, Lockheed Martin
A less successful project...

• Adopted CASE-driven Object-Oriented approach (Shlaer-Mellor)
• SPARK selected to ease certification but code was not developed in SPARK
• Conversion to SPARK started after testing
• All SPARK requirements were seen as “distortions” of the design
A less successful project (2)

- Extremely rapid progress was made with design and code
- Progress slowed markedly at integration phase
- 80% of the budget is now spent and the system does not work
- A complete redesign (probably SPARK-driven) or cancellation remain distinct possibilities
Conclusions

- SPARK works best if it is considered from Day 1 of a project. Knowledge of *design* issues is crucial.
- SPARK has proven success (both technically and commercially) on 00-55, DO-178B, and ITSEC E6 compliant projects.
- Proof is now a reasonable and deployable verification technology.
  - Project experience actually shows that proof is cost-effective, as well as getting you a better product.
  - You don’t necessarily need a PhD in maths...