Industrial case study: Zero defect secure software for the National Security Agency

Martin Croxford CEng MBCS

Praxis High Integrity Systems
Bath, UK

“Management summary”

- EAL5+ viewed as too difficult, too expensive or both. But EAL5+ standard applications increasingly in demand.
- NSA evaluated CbyC to determine its effectiveness at EAL5+
- NSA reported CbyC “Produces code more quickly and reliably and at lower cost than traditional methods”
- CbyC represents state of the art secure software engineering practice. CbyC can be learnt by competent engineers.

Agenda

- Project Goals
- Tokeneer Identification System
- What is Correctness by Construction?
- CbyC applied to TIS
- Feedback from NSA
- Conclusions

Agenda

- Project Goals
- Tokeneer Identification System
- What is Correctness by Construction?
- CbyC applied to TIS
- Feedback from NSA
- Conclusions

The MULTOS CA System

"This project has provided further evidence for the benefits of formality, in terms of rigour and predictability"
- MXI Head of Security

- System to support smart card security
- Demanding requirements:
  - Security: ITSEC Level E6 (EAL 7)
  - Availability: 6 months non-stop (24x7)
  - COTS h/w and s/w (eg NT) plus bespoke h/w and s/w
- Key metrics:
  - 0.04 defects/KSLOC; 28 SLOC/day; 1 year warranty

Project Goals

- Demonstrate that Common Criteria requirements for EAL5 are achievable in a cost effective manner
- Show how the Praxis “Correctness by Construction” approach matches up to EAL5.
- Measure productivity and defect rates under controlled conditions.

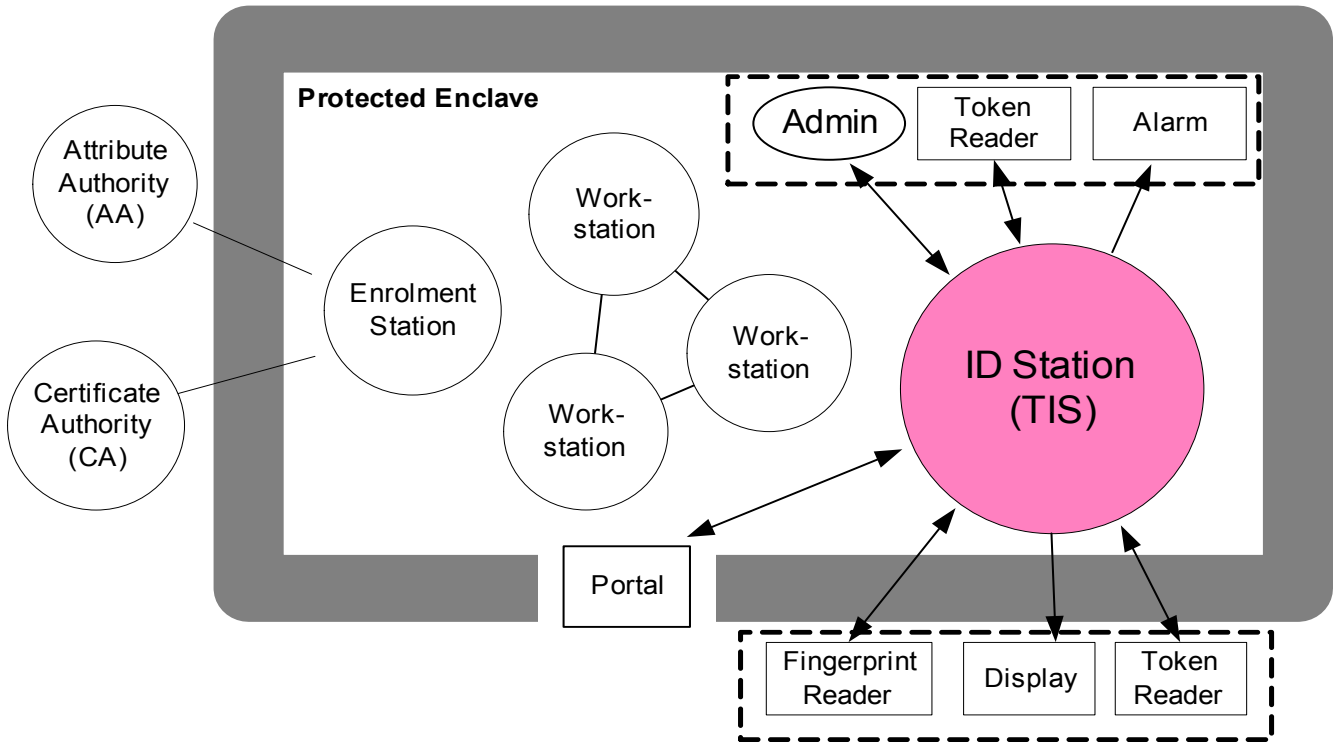
Agenda

- Project Goals
- Tokeneer Identification System
- What is Correctness by Construction?
- CbyC applied to TIS
- Feedback from NSA
- Conclusions

Tokeneer Identification Station Background

- Tokeneer
  - originally developed by NSA
  - Demonstrates use of smart cards and biometrics for access control
- ID Station
  - One component of Tokeneer
  - Provides user authentication

What is TOKENEER ?



- Provides protection to secure information held on a network of workstations situated in a physically secure enclave.
- Demonstrates use of **Smart Cards** and **Biometrics**

Agenda

- Project Goals
- Tokeneer Identification System
- What is Correctness by Construction?
- CbyC applied to TIS
- Feedback from NSA
- Conclusions

What is Correctness by Construction?

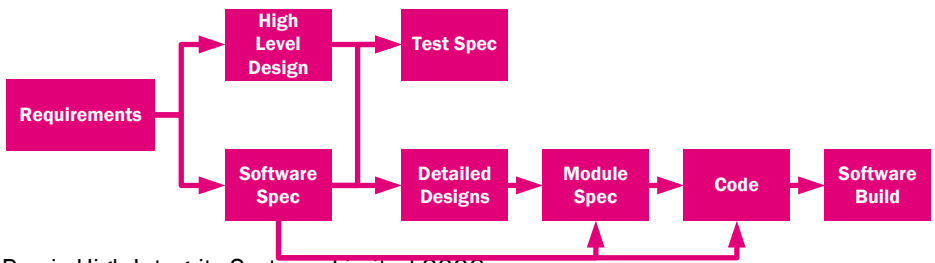
- A process, techniques and tools for developing high-integrity software
  - Has been applied by Praxis and its customers for fifteen years, on projects in aerospace, defence, transportation, telecomms and finance
- Philosophy: make it difficult to introduce defects
  And: eliminate defects as early as possible
- Possible ONLY by introducing precision and logical reasoning about each step in the development of the software
  - Demonstrate correctness through the way it is produced…
  - …not JUST on observing operational behaviour (or test results)
- Avoids the code/test/debug bottleneck
- Consistent with SW01, ESARR 6, (00-55), (DO-178C), Common Criteria EAL5+

The seven principles of CbyC

- Write right (use sound, formal notations)
- Step, don't leap (small semantic steps)
- Say something once, why say it again? (avoid repetition)
- Check here before going there (verify early)
- Argue your corner (document justifications)
- Screws: use a screwdriver, not a hammer (use the tool appropriate for the job)
- Brains 'R' Us (don't turn the handle)
- *Generate evidence as you go*

Summary of key aspects of Correctness by Construction

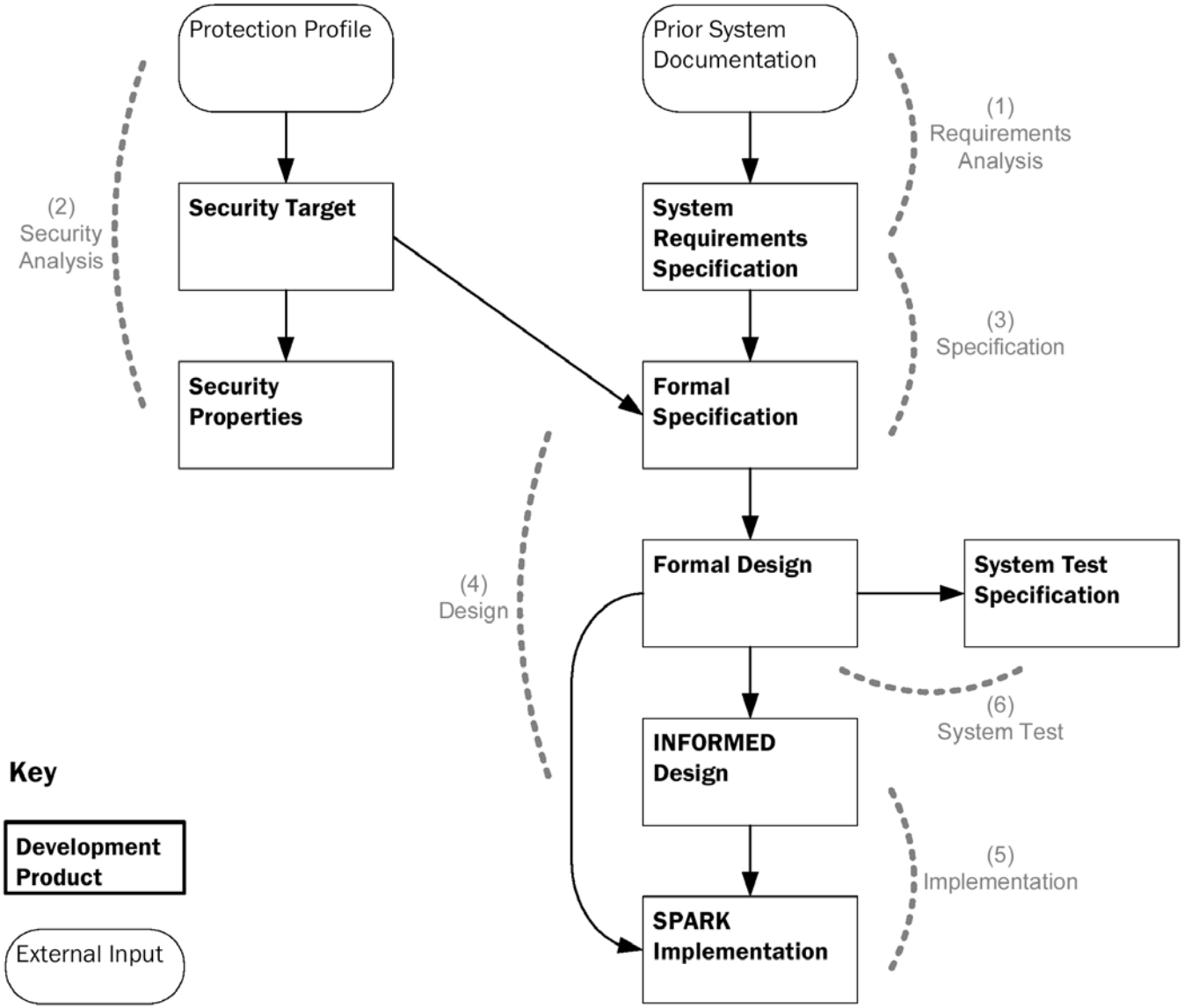
- Industrial strength requirements engineering (REVEAL)
  - Understand underlying business objectives
  - Understand domain: environment/context
  - Explicitly address conflict and change
- Cost-effective, pragmatic application of mathematical methods
  - Eg completeness of specification, proof of absence of deadlock in design
- Emphasis on static analysis
  - Both tool-supported (SPARK) and manual
  - Eg proof of absence of run-time errors



Agenda

- Project Goals
- Tokeneer Identification System
- What is Correctness by Construction?
- CbyC applied to TIS
- Feedback from NSA
- Conclusions

TIS Development Process



1. Requirements Analysis

- Praxis REVEAL® Process
  - Identify system boundaries
  - Clarify dependencies on the environment
  - Key scenarios identified

2. Security Analysis

- Security Target and Security Policy Model using Protection Profile
- Identify key properties to ensure security
- Specify security properties using Z

3. Specification

- Abstract Model of System using Z Notation
  - Defines and documents functional requirements
  - includes interfaces and observable behaviour
  - excludes internal detail (such as what was audited and format of certificates)
- Assurance
  - Customer feedback
  - validation that specification meets security properties
  - rigorous checks on initial state and operation preconditions

4. Design (Formal)

- Concrete Model of System using Z Notation
  - Refinement of Formal Specification
  - Introduces internal detail
  - Concrete model of certificates and audit log
  - Resolved all non-determinism in Spec
- Assurance
  - Sample of Refinement proofs (inc. audit log)

4. Design (INFORMED)

- Defines structure of implementation modules
- Identify and locate state, types and operations
  - Relates Abstract SPARK state to Z state
  - Relates SPARK types to Z types
  - Relates SPARK Ops to Z operation Schemas
- Other Design issues
  - Constraints not modelled in Z
  - Detailed file formats

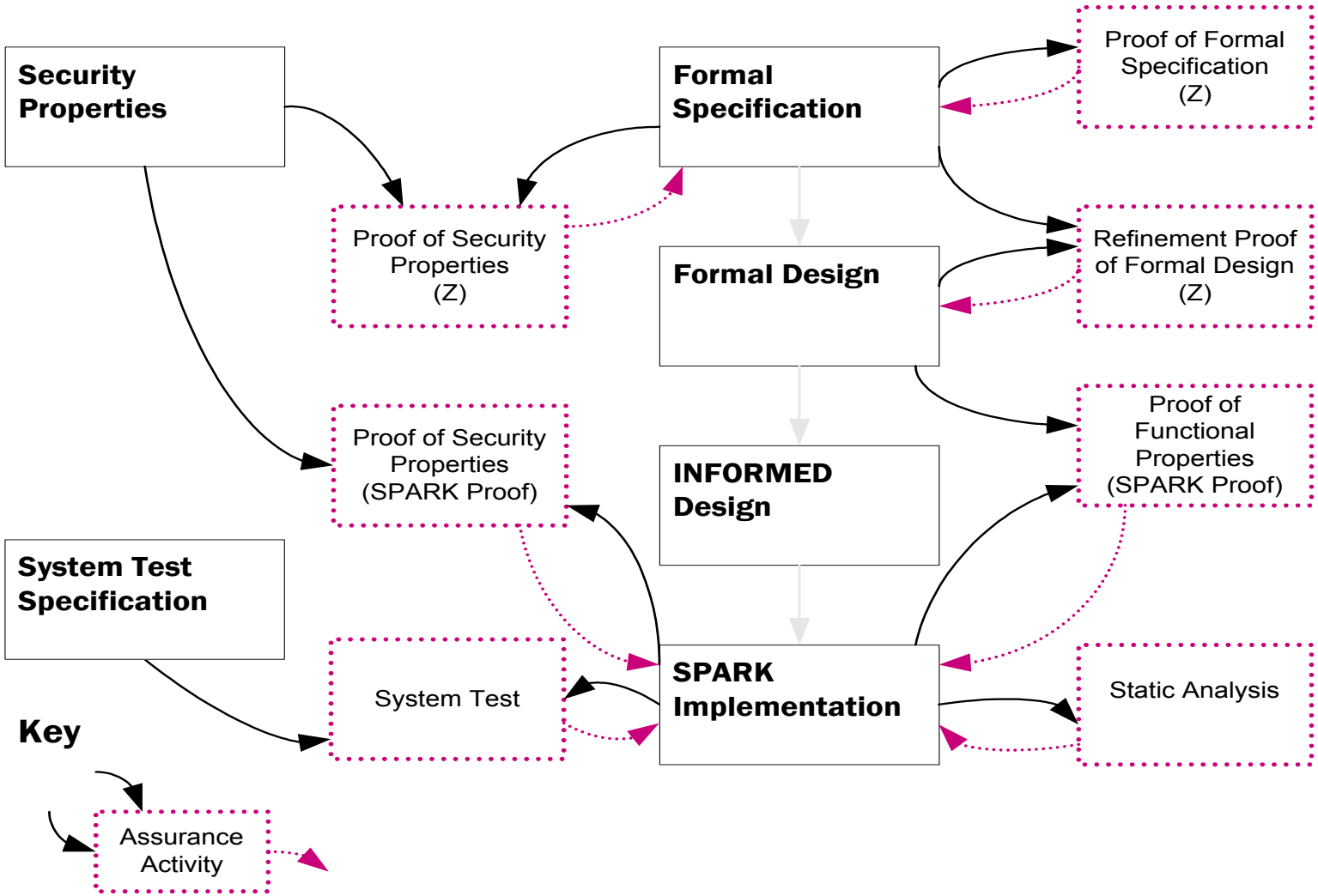
5. Implementation

- SPARK Ada for core
  - Static analysis using SPARK Examiner
    - Flow and proof annotations
    - Data-Flow / Information-Flow Analysis
    - Run-time Error checking
    - done before code review
- Thin layer of Ada interfacing to support software
- Assurance
  - Proof of some security properties
  - 93% VCs discharged automatically

6. System Testing

- System Testing only
  - Incremental builds with increasing functionality
  - Tests derived from refined specification
  - Specified as test scenarios with expected results (visual and audited events)
  - Coverage analysis
    - 100% statement coverage
    - 100% branch coverage
- Static analysis removes need for expensive module testing

Assurance process



Development Statistics

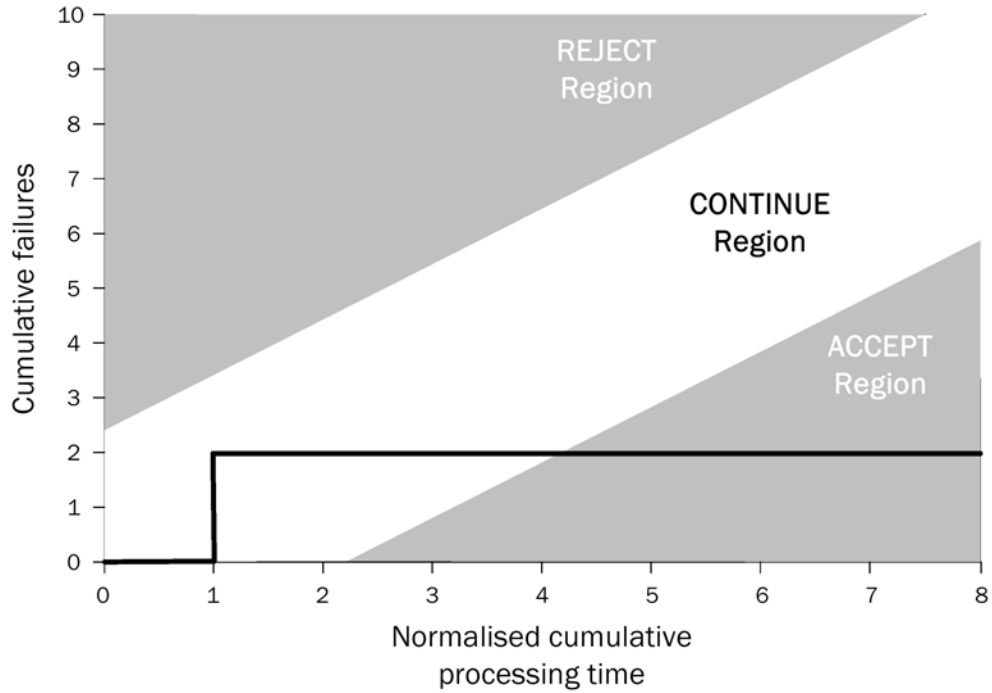
	Ada Source Lines	SPARK	LOC/day coding	LOC/day overall
Core	9,939	16,564	203	38
Support	3,697	2,240	182	88

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

Independent Assessment

- Performed by SPRE Inc.
- Developed Reliability Requirements
- Automated test Environment simulated 1-year's activity.
- Analysis of Results

Reliability Demonstration Chart (minor failures)



- Reliability objectives for TIS were met
  - 2 minor failures in user documentation.
  - No major or critical failures (i.e. zero defects)

Agenda

- Project Goals
- Tokeneer Identification System
- What is Correctness by Construction?
- CbyC applied to TIS
- Feedback from NSA
- Conclusions

Customer's View – Using Z

- Early step from English to Z is crucial part of process
- Customer must validate that this step has been achieved correctly
- Experience indicates that with good instruction a reading knowledge of Z can be acquired in a few days.
  - Good explanatory English important
  - Plan access to a Z expert

Customer's View – Other steps

- Review of later development output simplified by:
  - small steps taken at each stage
  - preservation of structure
- Required SPARK expertise
  - Need a reading knowledge of SPARK
  - Familiarity with SPARK tools to perform spot-checking of proofs.

NSA on Correctness by Construction

"Produces code more quickly and reliably and at lower cost than traditional methods"

"Reasonable learning curve"

"Correctness by Construction is proven and practical"

Randolph Johnson, National Security Agency

Agenda

- Project Goals
- Tokeneer Identification System
- What is Correctness by Construction?
- CbyC applied to TIS
- Feedback from NSA
- Conclusions

National Cyber Security Partnership - Secure software task force report

- Formed in response to the White House National Strategy to Secure Cyberspace

- A primary cause of security problems is software with vulnerabilities caused by defects
- Practices leading to low-defect software are to be encouraged
- Only three effective practices identified:
  - Praxis's Correctness by Construction
  - Team Software Process
  - Cleanroom

- www.cyberpartnership.org/SDLCFULL.pdf

Conclusions from NSA case study

- CbyC
  - Produces high quality, low defect software
  - Is cost effective
  - Conforms to Common Criteria EAL5+
  - Can be learnt quickly and effectively
- EAL5 and above is achievable and cost effective using current best practice software engineering
- There is no longer any excuse for avoiding EAL5+

Praxis High Integrity Systems Limited

20 Manvers Street
Bath BA1 1PX
United Kingdom

Martin Croxford CEng MBCS
Business Manager
Email: martin.croxford@praxis-his.com
Cell: +44 (0) 7881 516750

Telephone: +44 (0) 1225 8237941
Facsimile: +44 (0) 1225 469006
Website: www.praxis-his.com