Protean Code ltd
Process, Technology and Training for Critical Software

Soundness, Evidence and Discipline in High-Assurance Software

Roderick Chapman,
Director, Protean Code Limited
Honorary Visiting Professor, Dept. of Computer Science, University of York

Protean Code ltd
Process, Technology and Training for Critical Software

Contents

- **Formal? Sound?**
- Why bother with Formal?
- Observations
- Golden Rules
- Homework

Formal? Sound?

- DO-333 (para FM.1.6.1) says

"…to be formal, a model should have an *unambiguous, mathematically defined* syntax and *semantics*."

Formal? Sound?

- And goes on (para FM.1.6.2) to say

"…an analysis method can only be regarded as formal analysis if its determination of property is sound. Sound analysis means that the method *never asserts a property to be true when it is not true*."

Protean Code ltd
Process, Technology and Training for Critical Software

Formal? Sound?

- In English... This means...

1. Languages where you (and a tool) know *exactly* what it means..

2. Tools that you can trust (with justification)...

    - Sound tool says: “There are *definitely no* defects...”

    - Unsound tool says: “I’ve tried my hardest and I can’t find *any more* defects...”

Protean Code ltd
Process, Technology and Training for Critical Software

Quiz time...

"The machine interprets whatever it is told in a quite definite manner without any sense of humour or sense of proportion...

Unless in communicating with it one says exactly what one means, trouble is bound to result."

Alan Turing

Lecture to the London Mathematical Society, 1947

Protean Code ltd
Process, Technology and Training for Critical Software

“Proof?”

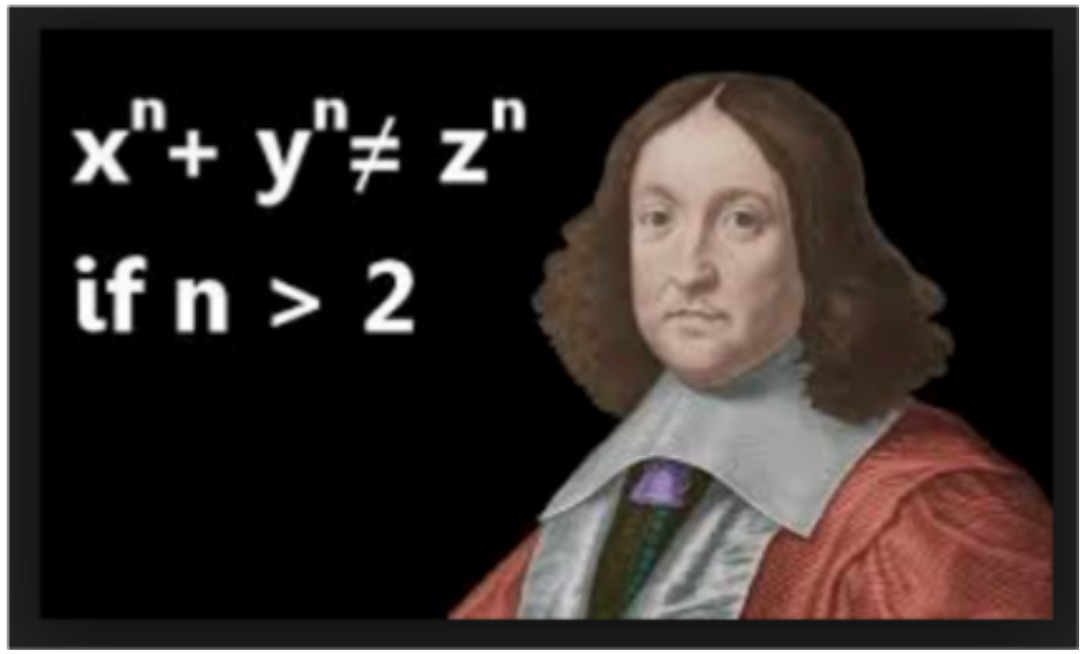
- If a supplier says

“Our code has been proven to be OK”

- Should you believe them?
- What has been “proven” anyway?
- What assumptions have they made?
  - (What assumptions did the tool vendor make for them?)

Protean Code ltd
Process, Technology and Training for Critical Software

“Proof” is a social thing...



- Is this True? Who says?

Protean Code ltd
Process, Technology and Training for Critical Software

Contents

- Formal? Sound?
- **Why bother with Formal?**
- Observations
- Golden Rules
- Homework

Protean Code ltd
Process, Technology and Training for Critical Software

The dreaded “F Word...”

- At Altran UK, my colleagues have used Formal languages and analyses for many years:
  - In specification: e.g. Z, CSP, SCADE etc.
  - In “code”: e.g. SPARK Ada subset + tools.
- What happens when you “Go Formal”? Is it worth it?

Protean Code ltd
Process, Technology and Training for Critical Software

Why bother with Formal?

$$\Delta IDStation; \Delta RealWorld \mid$$

$$\ldots TISOpThenUpdate$$

$$\ldots = locked \land latch' = unlocked$$

$$\vdash$$

$$(\exists Valid \ldots goodT(\theta ValidTo \ldots currentUserTok$$

$$\land User \ldots KNoCurr \ldots eck$$

$$\land FingerO \ldots$$

$$\lor$$

$$(\exists TokenWithValid \ldots TokenWithValidAuth)$$

$$\land UserT \ldots thOKAuth \ldots CurrencyCheck)$$

$$\lor$$

$$(\exists Val \ldots \bullet goodT(\theta ValidToken) = \ldots AdminTo$$

$$\ldots thCert \neq \emptyset \land (the\ authCert).role \ldots)$$

> See: *TISOpThenUpdate* (p. 5), *UserTokenOKNoCurrencyChe*
> *UserTokenWithOKAuthCertNoCurrencyCheck* (p. 5)

Protean Code ltd
Process, Technology and Training for Critical Software

Why bother with Formal?

Protean Code ltd
Process, Technology and Training for Critical Software

Thinking and Tooling exposes...

Ambiguity…

Protean Code ltd
Process, Technology and Training for Critical Software

Thinking and Tooling exposes...

Contradiction…

Protean Code ltd
Process, Technology and Training for Critical Software

Thinking and Tooling exposes...

Incompleteness…

Protean Code ltd
Process, Technology and Training for Critical Software

Thinking and Tooling exposes...

#include <customer_conversation.h>

Protean Code ltd
Process, Technology and Training for Critical Software

Formal notations...

- ...exhibit *semantic consistency*.
- In short, my “code” *means exactly the same thing* to:
  - All compilers
  - All “target” machines
  - All verification tools
  - The person that wrote it...
  - The person that reviews it...
  - The person that has to maintain it in N years time...

Protean Code ltd
Process, Technology and Training for Critical Software

Semantic consistency... A side-benefit

- If my “Code” has exactly one meaning, then *all* verification tools should give the same results, right?
  - Well... nearly... False-positives will be different...
  - Some tools will be “better” than others at detecting certain defects.
  - Enables a rational market for *diverse* verification tools.
- Counter-example: try to get consistent results from 2 or more MISRA C tools... See what happens!

Protean Code ltd
Process, Technology and Training for Critical Software

And finally...
Formal notations have *longevity*...

- If my “Code” has the same meaning in twenty years time as it does now.

- I can change compiler and target machine and *it should just work...*

- This happens with long-lived systems, for example... “mid life upgrade” etc.

Protean Code ltd
Process, Technology and Training for Critical Software

Contents

- Formal? Sound?
- Why bother with Formal?
- Observations
- Golden Rules
- Homework

Protean Code ltd
Process, Technology and Training for Critical Software

Does Soundness matter?

- “Soundness doesn’t matter”
  - Who says...
    - Err...Tools vendors with unsound tools...
- The market has spoken...
- The market is *broken*?

Protean Code ltd
Process, Technology and Training for Critical Software

Using sound verification tools...

Theorem prover says “no”

SAT solver says “counter-example”

Protean Code ltd
Process, Technology and Training for Critical Software

Using sound verification tools...

- The first few months can be pretty depressing...
- Everything I do is wrong! ☹
- But you get used to it...Soundness makes me Humble...
- Eventually...You learn to beat the tools...
- You learn *trust* the tools...

Protean Code ltd
Process, Technology and Training for Critical Software

Do you trust a tool?

- Trust in verification tools is hard-won and easily-lost.
- Problem: almost all the “big name” static verification tools are blatantly (and some...honestly) *unsound* for verification of non-trivial properties...
- Why?
- See Coverity’s “Billion lines of code later...” article in CACM 2010. In short: the market decided...

Do you trust a tool?

Protean Code ltd
Process, Technology and Training for Critical Software

- Time for “soundness cases”?
- Tool vendor presents “Soundness case” (i.e. “Why you should trust us...”) including
  - Assumptions
  - Defect history and corrective actions...
  - Reference to underlying maths
  - Lack of counter-evidence.

Protean Code ltd
Process, Technology and Training for Critical Software

A social proof for Soundness?

- If “Formal Proof” of tool soundness is too hard, can we find a “Social Proof”???

- Possibly...

- Question: if tools says “No defects of class X”, do you still look for X in any later verification activity (like review and testing)? Do you ever find a class X defect?

- No? Congratulations... You are treating the tool “as-if sound.”

- Problem: convince your customer and regulator that the tool’s evidence is trustworthy...

Protean Code ltd
Process, Technology and Training for Critical Software

Observation

- We need to advocate and teach

"Verification Driven Design"

- Including consideration of *all* the forms (both static and dynamic) of verification that are needed for a particular project.

- The current fad for "Test Driven Design" places too much emphasis on dynamic verification criteria to establish fitness-for-purpose. (e.g. "We tested it lots...")

Protean Code ltd
Process, Technology and Training for Critical Software

Observation

- Google for “Formal Verification” – what do you notice?
- “Formal” is now regarded as *standard practice* in the design of hardware, especially in the design of silicon SoCs, VLSI, FPGAs etc...
- How come?

Protean Code ltd
Process, Technology and Training for Critical Software

A Fumble Future?

- What about software?
- Turing's advice was ignored... ☹
  - Languages became *more* complex and *more* ambiguous.
  - Even Rust repeats design errors from C... ☹ ☹
- Signs of hope: SCADE, SPARK Ada, Frama-C, Eiffel, CakeML, Cryptol, Formal ARM ISA etc. etc.

Protean Code ltd
Process, Technology and Training for Critical Software

Contents

- Formal? Sound?
- Why bother with Formal?
- Observations
- **Golden Rules**
- Homework

Protean Code ltd
Process, Technology and Training for Critical Software

Golden Rules

- An old one, but a good one...

Garbage In, Garbage Out

Or...

“You can’t polish dirt”

Protean Code ltd
Process, Technology and Training for Critical Software

Golden Rules

No Verification without Specification

But...

It’s OK if the “specification” is a universal and implicit rule such as “No buffer overflows”.

Protean Code ltd
Process, Technology and Training for Critical Software

In Conclusion...

- Formal + Sound + Humble can be done... And it works...
- Formality in verification makes you Humble...

  "It's like Jazz – hard at first, but worth it in the long run..."
  Peter Amey, SPARK Team.

- Formality makes you better.

Protean Code ltd
Process, Technology and Training for Critical Software

Contents

- Formal? Sound?
- Why bother with Formal?
- Observations
- Golden Rules
- **Homework**

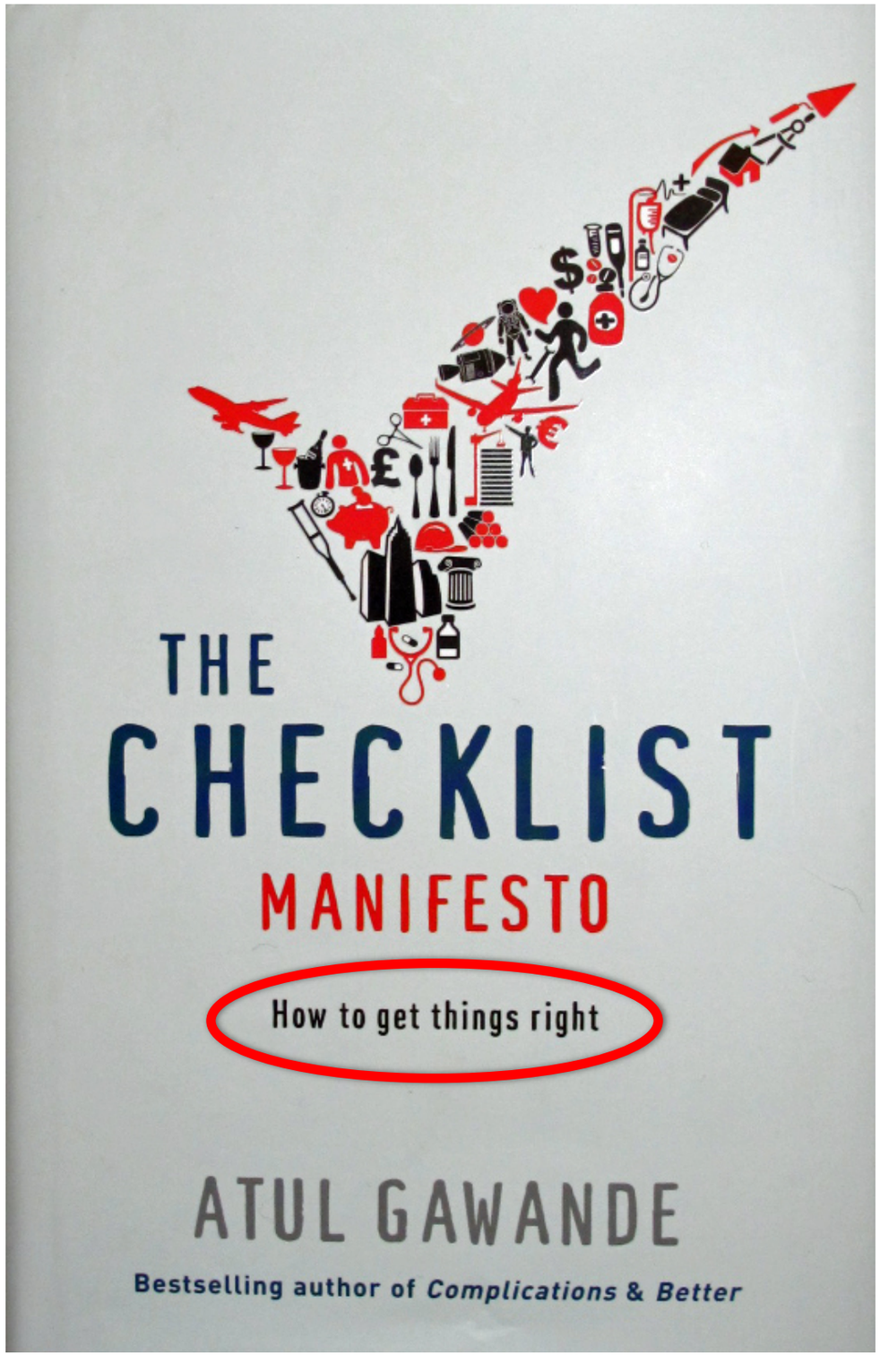
Protean Code ltd
Process, Technology and Training for Critical Software

Homework (1)

- If you're a "Programmer"
- Read Alan Turing's 1947 Lecture to the London Mathematical Society...
- See just how far ahead Turing was!

Homework (2)

Protean Code ltd
Process, Technology and Training for Critical Software



Questions…

Protean Code ltd
Process, Technology and Training for Critical Software