No One does Formal Methods, right?

Rod Chapman

Praxis High Integrity Systems

Contents

- Who’s using Formal Methods?
- Perception of FM…
- So why formalize?
- Maintaining formality…
- What can/can’t we formalize?
- Praxis FM projects
- Other FM projects
- Final thoughts

Contents

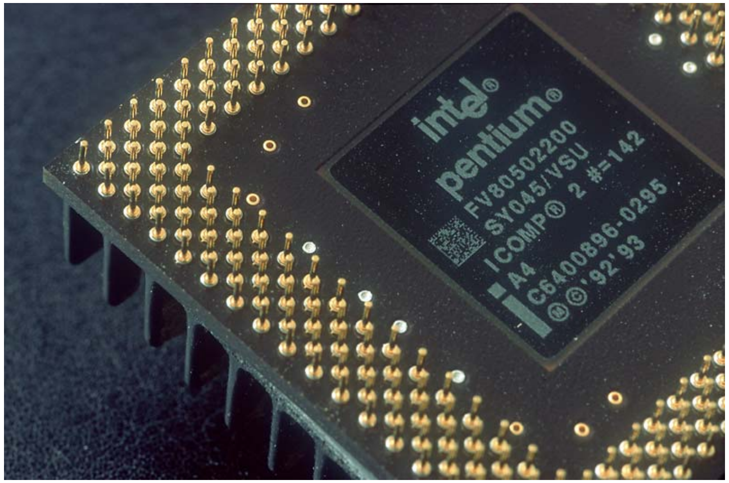
- Who’s using Formal Methods?
- Perception of FM…
- So why formalize?
- Maintaining formality…
- What can/can’t we formalize?
- Praxis FM projects
- Other FM projects
- Final thoughts

Who's using Formal Methods?

- How many of you use a formal programming language in your everyday software development projects?

A precise, formal language

```
10100110100100001000100010010001
10101000111101010100001010101010
10101010100101001010101010000101
01010010010101001101010101010101
00000101000101010010101001010010
01010101010101010101011110100101
00101010101010100101010101010010
11010010101010100101010010100000
11111110101001010011111100101010
01010110000100101010100010101001
00101010101010010100000000001111
01110000000101001000010100101010 1
11100000000011110101010000101111
```



Machine Code as a Formal Language

- Pros
  - – Has a formal operational semantics (one would hope…)
  - – Precise – you know exactly what it means.
- Cons
  - – Somewhat inpenetrable, for both human and/or machine-analysis
  - – “Abstraction loss” following compilation.

Contents

- Who’s using Formal Methods?
- Perception of FM…
- So why formalize?
- Maintaining formality…
- What can/can’t we formalize?
- Praxis FM projects
- Other FM projects
- Final thoughts

Perception of FM…

- In the 80s and 90s, FM was perhaps "oversold" by the acacdemic community…
- BUT…many say that FM "failed"…
- I disagree – it was mostly judged to be too hard, and *not tried at all*…

Perception of FM…

- There is a baffling confusion between popularity and success in computing.
- Perception is: popular=successful, and so unpopular=dead/no good/failed/obsolete.
- “A fashion industry with delusions of grandeur” (Les Hatton)

- Yet…niche technologies do exist that are fit-for-purpose, and successful businesses thrive on them.

Perception of FM…

- Finally, where's the data?
- Who says FM doesn't work?
- Most industrial projects don't publish (either success or failure!)
- Almost no-one sets up control experiments.

- *People* are highly variable in behaviour, talent, productivity etc.

Contents

- Who’s using Formal Methods?
- Perception of FM…
- So why formalize?
- Maintaining formality…
- What can/can’t we formalize?
- Praxis FM projects
- Other FM projects
- Final thoughts

So why formalize?

- Let’s start with Requirements Engineering and Specification writing.
- E.g. construct a formal functional specification based on “informal” requirements.

- What happens?

So why formalize?

- What happens when you try to formalize something?
  - You ask lots of hard questions…
  - You find out that the source material is junk!
    - Inconsistent
    - Ambiguous
    - Incomplete
- You can find this out now or later on…take your pick!

Why formalize? (2)

- Formal specifications become like the blueprints produced by an architect.
  - You can ask a contractor to prepare a bid against them.
  - A regulatory body can review them to make sure they comply with best-practice (like “planning permission”)
  - If the builder makes a mess, you have something to point at when you end up in court…

Why formalize? (3)

- Common complaint: “The requirements are bound to change, so there’s no point building a formal spec…”
- BUT…who says a formal specification will be any more or less difficult to change than an informal one…
- How do you know how much a change will cost anyway?

Change management with formal methods

- If you *do* have a formal description, you might have a decent chance of knowing what will change, what won’t, and how much it will cost.

- Key point: on projects, we find that most “change” in requirements isn’t a fundamental change of anything at all.
- It’s merely a clarification of stuff you never bothered to understand in the first place.

Why formalize? (4)

- Sometimes, we have given customers a warranty with a piece of software, but only if we have a formal specification as a “contractual baseline”.

- We are still in business… ☺

Contents

- Who’s using Formal Methods?
- Perception of FM…
- So why formalize?
- Maintaining formality…
- What can/can’t we formalize?
- Praxis FM projects
- Other FM projects
- Final thoughts

Maintaining formality

- If we’ve got a formal spec., why not “maintain the formality” by using a formal programming language?

- The same thing happens! You try to implement a formal spec formally, and you find…ambguity, incompleteness, inconsistency.

Maintaining formality

- Example: secure software
- Imagine a programming language what will always detect all instances of buffer overflow and failure to validate input data.
  - Actually, no need to imagine this: we have it right now..

Maintaining formality

- Result: you have to “beef up” your specification to cover all input data validation cases.
- You also have to specify error-handling behaviour for all cases.
- Gee…what a neat idea…programs that are robust in the face of white-noise or arbitrarily malicious input data…
- “Formality loss” caused by informal programming style allows weak implementation to go unnoticed or unreported.

Contents

- Who's using Formal Methods?
- Perception of FM…
- So why formalize?
- Maintaining formality…
- What can/can't we formalize?
- Praxis FM projects
- Other FM projects
- Final thoughts

What can we formalize?

- Some areas of system design and construction are well supported by formal languages and tools
- An incomplete list:
  - Functional specification: Z, B…
  - Concurrency: CSP, Pi Calculus…
  - Sync state machines: Lustre…
  - Control laws: MATLAB…
  - Protocols: SPIN…
- OK so far…

What can’t we formalize?

- In some areas, we’re still struggling
- Another incomplete list:
  - Architecture? What’s that then?
  - Non-functional properties: safety, security etc…
  - Composition of the above. How do you know that your Z spec, plus CSP process diagram, implemented on architecture Y will work at all?

Contents

- Who’s using Formal Methods?
- Perception of FM…
- So why formalize?
- Maintaining formality…
- What can/can’t we formalize?
- Praxis FM projects
- Other FM projects
- Final thoughts

Praxis FM projects

- Four projects:

- SHOLIS
- MULTOS CA
- Tokeneer
- iFacts

SHOLIS (1996)

- Ship/Helicopter Operating Limits Instrumentation System
- Advises ship's crew on safety of helicopter operations.
- First ever project to attempt Def Stan 00-55 SIL4. Huge technical risk!
- Spec: 200 pages Z
- Implementation: 27000 loc SPARK with static analysis and proof

MULTOS CA (2000)

- Formal Security Policy in Z (30 pages)
- Functional spec in Z (500 pages)
- Concurrency design in CSP + Model Checking
- 100,000 lines of code (mixed-language), 3500 person-days, 27 loc per day.
- Only 4 defects 1 year after delivery, corrected under our warranty of course!

Tokeneer (2003)

- Technology demonstrator of secure software development for the NSA
- Biometric access control.
- As MULTOS CA – Z used for security policy and functional spec and design.
- Implemented in SPARK – 10000 loc.
- NO BUGS ever reported since we delivered it.

iFacts (now…)

- New tools for management of en-route air traffic in the UK.
- Predictive medium-term conflict detection.
- Specification: over 2000 pages of Z now!
- Code: over 250000 loc SPARK and Ada.
- More results in a year or so… ☺

Contents

- Who’s using Formal Methods?
- Perception of FM…
- So why formalize?
- Maintaining formality…
- What can/can’t we formalize?
- Praxis FM projects
- Other FM projects
- Final thoughts

Other FM Projects

- Common complaint: “Yeah…but that’s just Praxis…”
- Well..there are a few others…
  - GDUK: specification of Harrier II “balance” algorithm
  - Rolls-Royce: jet engine control
  - Mastercard: MULTOS Operating System
  - QinetiQ: EuroFighter flight control laws verification

Contents

- Who’s using Formal Methods?
- Perception of FM…
- So why formalize?
- Maintaining formality…
- What can/can’t we formalize?
- Praxis FM projects
- Other FM projects
- Final thoughts

Final thoughts

- Many people ask:
  - “Is there a future for formal methods?”
- I respond:
  - Is there a future without formal methods?

Final thoughts

- There's much to be done though:
  - Tools, tools, tools,
  - "Hide the math..."
  - Automation, automation, automation

- There's one big subject I haven't even mentioned...
- Hardware...massive use of FM...but that's a topic for another day (and another speaker...)

Why don’t companies adopt methods that work?

We are like the **barber-surgeons** of earlier ages, who prided themselves on the **sharpness of their knives** and the **speed with which they dispatched their duty**
- either shaving a beard or amputating a limb.

Imagine the **dismay** with which they greeted some ivory-towered academic who told them that the practice of surgery should be based on a **long and detailed study** of human anatomy, on **familiarity with surgical procedures** pioneered by great doctors of the past, and that it should be carried out only in a **strictly controlled bug-free environment**, far removed from the hair and dust of the normal barber's shop.

Professor Sir Tony Hoare, Microsoft Research.

Praxis High Integrity Systems

20 Manvers Street
Bath BA1 1PX
United Kingdom
Telephone: +44 (0) 1225 466991
Facsimilie: +44 (0) 1225 469006

Website: www.praxis-his.com, www.sparkada.com

Email: sparkinfo@praxis-his.com