Dependable Software: A View

Peter Bernard Ladkin

University of Bielefeld CITEC and Causalis Limited

21 June 2011

Epiphany

- Recent Events
  - Fukushima Dai-ichi Accident
  - Family Court application
  - Publishing my paper
  - Modern university "research" environment
- In Common: we've lost the picture (but I'm not sure we ever had it)
- How to regain the Picture?
- Conclusion: ....

Thinking

- Are we thinking hard enough about what we are doing?

Before I Get Stuck In (or Sidetracked)

- The technical paper:
- Go to *www.rvs.uni-bielefeld.de*
- From the links left, choose *Publications*
- Scroll down a bit to *What's New*
- There it is
- Or would you rather have the URL? Why?

Human Requirements Conflicts, Example 1

- From my experience 15 years ago
  - Complex, possibly perfect, specification language and method
  - "Method" not described
  - I devised one, with large hints from the developer
  - It worked! But it proved difficult to transmit to students
  - I remain one of about only a half-dozen people who can use it
  - And even I probably can't, any more

Human Requirements Conflicts, Example 2

- One of the more visible international conferences in this field
  - I have been often on the Committee
  - And I read a lot of the papers (also at selection time)
  - Industrial people complain that it's "academic"
  - Committee members say: there are few "quality" submissions from industry
  - All agree! The question is: what to do about it
  - Things stay as they are. But everyone wants them different

Human Requirements Conflicts, Example 3

- "Formal methods don't work!" (reputed: B. Boehm, 1980's)
- Some of us: "they do, you know!" (Sir Tony H, Martyn Thomas, AdaCore, me, my pals, my cat)
  - "But we can't learn them"
  - "We'll develop some you can learn"
  - "Costs too much (people, time) for the benefit"
- Moral question: should we any longer be building systems which we don't guarantee are fit for purpose?
- Business/social question: why does it (still) "cost too much for the benefit"?

Aside: Resolving Example 3

- SW is a mathematical object (Sir Tony H.)
- But it obviously isn't (many C compilers)
- Response 1: it should be (it's about time)!
- Response 2 (the "mature" thought): SW behavior can be assured fit-for-purpose **in so far as** the SW (behavior) can be -is- construed as a mathematical object
- Moral question: see above

Other Human Issues. Example 1

- One reads how nuclear power plant operators and authorities in Japan (and who knows who else?) were apparently insensitive to the tsunami hazard
- Where is the HazAn?
  - Whereever the official one is, it is not publically available
- Answer: in the book The Next Catastrophe" by Charles Perrow, pp 134 and 173
- Why does it take a *sociologist* to perform a HazAn? Isn't it our job?

The Broader Resulting Issue, Firmed Up

- Answer: Prof. Perrow has part of the picture which we don't have
- How do we get it?
  - By thinking harder
  - By paying more, and wider, attention to what we do *and its environment*
- Perrow is an organisational theorist
  - He knows how people work, and don't work, in organisations
  - He knows about the environment, that is, politics
  - He knows how easily things get suppressed and circumvented
- But that's our problem! Shouldn't *we* own it? Apparently we don't

What I Promised to Talk About

- I was going to talk about standards development for "safe software" in Germany
- (Systems people will say that phrase involves a category mistake)
- It involves saying what assurance methods were used and why, and
- (the key issue, I think) which methods were *not used* and why not
- I gave a preliminary version of the talk in Fulda in May
  - It went down amazingly well!
  - No riots. Everybody relaxed.
  - Triumph! Formal methods don't scare people any more!
  - Then it occurred to me ....
- I couldn't do that to the Edinburgh audience, now, could I?
- The major point: people weren't listening
  - Maybe because they didn't care
- Again: the human issue

Down to Brass Tacks

- Reliable SW *does what we want it to do*
  - Do we know what we want? Really know?
  - How do we tell that we know what we want?
  - How do we tell the SW does it?
- But it also *doesn't do what we don't want it to do*
  - How do we know what we don't want?
  - How do we assure ourselves that we know?
  - How do we tell that the SW doesn't do any of that?
- I am told by IFIP not to say "reliable" but rather "dependable"

Brass Tacks: Application

- My C compiler *does what I want it to do*
  - Do I know what I want? Yes. Really know? Yes.
  - How do i tell that we know what I want? 40 years experience.
  - How do we tell the SW does it? Duuuuhhh.
- But it also *doesn't do what we don't want it to do*
  - How do we know what we don't want? Trickier.
  - How do we assure ourselves that we know? I don't think we do
  - How do we tell that the SW doesn't do any of that? I don't think we do.
- What do I conclude about the "dependability" of my object code?

How Good Are We?

.........after all these years?

- Major military airplane, SW developed according to civil standards (DO-178B)
- SW developed according to DA Level A and B
- No significant quality difference found between Levels A and B
- "Module" quality generally very poor
  - Let me call the pieces of SW "modules", not a technical term here
  - The worst had a defect rate of 1 in 10 lines of executable code (LOC)
  - The best had a defect rate of 1 in 250 LOC
  - Errors found are a litany of run-time-type problems, including some that should count as solved since the late 1960's but apparently aren't

Types of Errors 1

(With thanks to Martyn Thomas and ....)
The following defects were among those reported in the software after certification:

- Erroneous signal de-activation.
- Data not sent or lost
- Inadequate defensive programming with respected to untrusted input data
- Warnings not sent
- Display of misleading data
- Stale values inconsistently treated
- Undefined array, local data and output parameters

Types of Errors 2

- Incorrect data message formats
- Ambiguous variable process update
- Incorrect initialisation of variables
- Inadequate RAM test
- Indefinite timeouts after test failure
- RAM corruption
- Timing issues - system runs backwards
- Process does not disengage when required

Types of Errors 3

- Switches not operated when required
- System does not close down after failure
- Safety check not conducted within a suitable time frame
- Use of exception handling and continuous resets
- Invalid aircraft transition states used
- Incorrect aircraft direction data
- Incorrect Magic numbers used
- Reliance on a single bit to prevent erroneous operation

How Good We Are, cont'd

- One airplane: 1 in 250 LOC or worse
- Rumored industry standard for safety-critical SW: 1 in 1000 LOC
- Best documented quality: 1 in 25,000 LOC (guess who!)

I say we aren't very good.

What To Do About It

- It must be a people problem
  - There is no other explanation for why mistakes are still being made whose technical solution has been known for four decades
- People problems are notoriously intractable
  - Recall the examples in my epiphany
- In short, I don't know what to do about it, except by bringing it to technical people's attention
- Still, address the memes and mantras

Memes and Mantras

- Actually, I prefer the term "trope"
- A meme is an idea that promulgates (Dawkins and Dennett)
- A mantra is a short statement or belief (see below)
- A trope is a mantra with reasoning
  - "Formal methods don't work". Baloney. But some proposals, even most, are impractical.
  - "There is no silver bullet". Maybe, but garlic is readily available
  - "Programming language Q is as good as programming language S if you take care". Taking care didn't help with all those errors in the aircraft SW. If the client had insisted on using a strongly-typed language with adequate compiler, most of them could not have occurred
  - " Our SW has been proven reliable in use" Can you show us the statistics? Is your statistical reasoning valid?

What Not To Do About It

- Write a generic software safety standard that is 50pp long
  - That is in part demonstrably logically incoherent
  - Based on a set of concept definitions that are appallingly sophomoric (if you happen to have a degree in a subject for which analysing definitions is essential)
- Then, thirteen years later, extend it to 150pp!
  - Not my fault! I came in later
  - But I will be involved with the next version
  - Will it be an improvement? We'll have to see
- Some prominent SW specialists think the standards process is broken
  - They have plenty of evidence to choose from!
  - If it's "broken", can it be fixed?
  - How?

The Core of an Ideal Safety Standard

- Determine what we don't want the system to do (Accidents)
  - As completely as possible
  - Provide the assurance that you have everything
- Determine how it could happen (Hazards)
  - As completely as possible
  - If you go too far, that's OK
  - Provide the assurance of coverage (completeness)
- "Apportion" the hazard behavior to the system components (including SW)
  - Show the apportionment covers the hazards
- For the SW, indeed any component: repeat the above

Evaluating the Core

- Did I include everything essential?
- Can you remember it?
- Can you remember it well enough to recite it at will to management?
- **If not, can you improve it?**
  - Keep it to a page, like a pilot's checklist
  - It's a "memory item"

Bringing in Architecture

- SW has four general life stages
  - Requirements development and specification
  - Design specification
  - "Source" code: more generally, the intermediate constructed object
  - The object code (linked)
- Apply the generic method to these four stages
- Hint: you pretty much have to use *formal refinement*

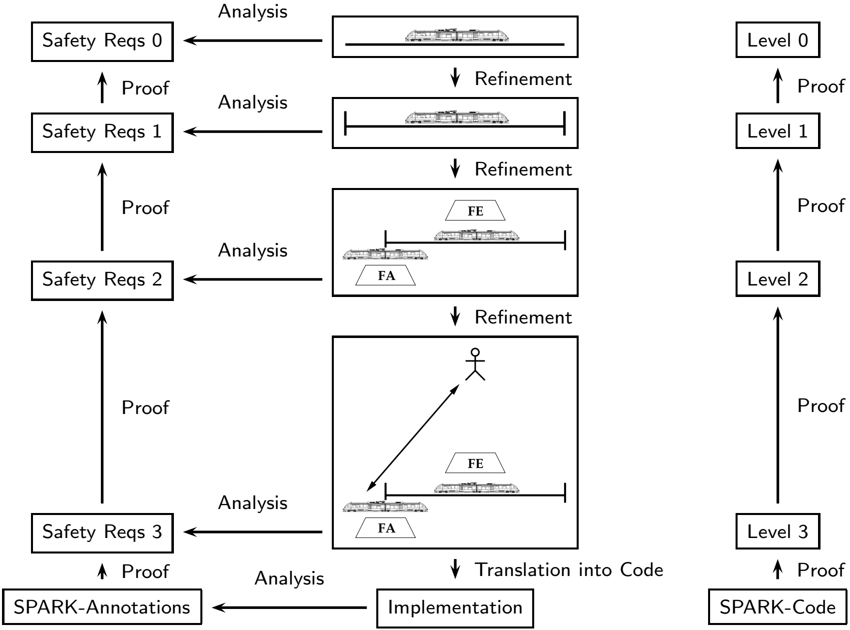
Bringing in Architecture

So, for example, you need

- To assess requirements
- Compare design against requirements
- Compare source code against design
- Compare object code against source code
- Consider run-time monitoring

There are 26 industrially-mature steps and techniques which can be applied. I won't list them here (not a "memory item", even for me). There are in the proposal we are discussing in DKE GK 914, and in the paper accompanying this talk

Example — Ontological Hazard Analysis



Finis

- But this talk is long enough
- Technical details are hardly ever remembered
- The technical material is available on the WWW
- The important thing is that people who are interested and concerned know it's there, and how to get to it
- ...... and are motivated to engage in the process of improving matters
- ... which I hope is what this talk has been about

Thanks for listening!