

Scientific Certification

John Rushby

Computer Science Laboratory
SRI International
Menlo Park, California, USA

Does The Current Approach Work?

- Fuel emergency on Airbus A340-642, G-VATL, on 8 February 2005 (AAIB SPECIAL Bulletin S1/2005)
- Toward the end of a flight from Hong Kong to London: two engines shut down, crew discovered they were critically low on fuel, declared an emergency, landed at Amsterdam
- Two Fuel Control Monitoring Computers (FCMCs) on this type of airplane; they cross-compare and the “healthiest” one drives the outputs to the data bus
- Both FCMCs had fault indications, and one of them was unable to drive the data bus
- Unfortunately, this one was judged the healthiest and was given control of the bus even though it could not exercise it
- Further backup systems were not invoked because the FCMCs indicated they were not both failed

Safety Culture

- It seems that current development and certification practices may be insufficient in the absence of **safety culture**
- Current business models are leading to a loss of safety culture
- Safety culture is **implicit** knowledge
- Surely, a certification regime should be effective on the basis of its **explicit** requirements

MC/DC Test Coverage

- Need criteria to indicate when we have done **enough (unit) testing**
 - This is for assurance, not debugging
 - Do not expect to find any errors
- Vast literature on this topic
- Many criteria are based on **structural coverage of the program**
 - E.g., branch coverage
- MISRA, DO178B Level A, require **MC/DC coverage**
- Generate tests from **requirements**, and measure coverage on the **code**

MC/DC and Automated Test Generation

- It's quite easy to automate test generation using model checking technology (check out [sal-atg](#))
- Trouble is, the model checker can be **too** clever: generally finds **shortest** test to reach a given test goal
- E.g., [autopilot](#) has two modes (used in pairs)
 - In **active** mode, complex rules determine when to enter roll mode
 - In **standby** mode, just follows active partner
- **Given test goal to exercise entry to roll mode, model checker puts system in standby mode, then tells it to go to roll mode**
- Naïve automated test generation can yield tests that achieve MC/DC coverage but have very poor error detection
- It's **implicit** that there's a rational **test purpose**

Purpose of MC/DC Testing

- “It has been said” that the real benefit of MC/DC testing is **not** that it forces reasonably thorough test coverage
- **But that it forces highly detailed requirements specifications**
 - Because code coverage must be achieved from tests derived from requirements
- And this is its real, **implicit** purpose

Approaches to Software Certification

- The **implicit** (or **indirect**) **standards-based** approach
 - **Airborne s/w** (DO-178B), **security** (Common Criteria)
 - Follow a prescribed **method** (or prescribed **processes**)
 - Deliver prescribed **outputs**
 - ★ e.g., documented requirements, designs, analyses, tests and outcomes, traceability among these
 - **Internal** (DERs) and/or **external** (NIAP) **review**
- **Works well in fields that are stable or change slowly**
 - Can institutionalize lessons learned, best practice
 - ★ e.g. evolution of DO-178 from A to B to C
- **But less suitable with novel problems, solutions, methods**
- **Implicit** that the prescribed processes achieve the safety goals
 - **No causal or evidential link from processes to goals**

Approaches to Software Certification (ctd.)

- The **explicit goal based** approach
 - e.g., **air traffic management** (CAP670 SW01), UK **aircraft**
- **Applicant develops an assurance case**
 - Whose outline form may be specified by standards or regulation (e.g., MOD DefStan 00-56)
 - Makes an **explicit** set of **goals** or **claims**
 - Provides supporting **evidence** for the claims
 - And **arguments** that **link the evidence to the claims**
 - ★ Make clear the underlying **assumptions** and **judgments**
 - ★ Should allow different viewpoints and levels of detail
- The case is evaluated by **independent assessors**

Evidence and Arguments

Evidence can be **facts**, **assumptions**, or **sub-claims**
(from a lower level argument)

Arguments can be

Analytic: can be repeated and checked by others, and potentially by machine

- e.g., logical proofs, calculations, tests
- **Probabilistic** (quantitative statistical) reasoning is a special case

Reviews: based on human judgment and consensus

- e.g., code walkthroughs

Qualitative: have an **indirect** or **implicit** link to claims

- e.g., CMI levels, staff skills and experience

Critique of Standards-Based Approaches

- The claims, arguments, and assumptions are usually only **implicit** in the standards-based approaches
- And many of the arguments turn out to be **qualitative**
 - Requirements to follow certain design practices
 - Requirements for “safe subsets” of C, C++ and other coding standards (JSF standard is a 1 mbyte Word file)
 - ★ cf. MISRA C vs. SPARK ADA (with the Examiner)
- **No evidence** these are effective, some **contrary** evidence

Critique of Standards-Based Approaches (ctd)

- Even when analytic evidence and arguments are employed, their **selection** and **degree of application** are often based on **qualitative** judgments
 - Formal specifications (but not formal analysis) at some EAL levels
 - MC/DC tests for DO-178B Level A
- “**Because we cannot demonstrate how well we’ve done, we’ll show how hard we’ve tried**”
 - And for really critical components, we’ll **try harder**
 - This is the notion of **software integrity levels** (SILs)
 - **Little evidence** what works, **nor that more is better**

Non-Critique of Standards-Based Approaches

- Often accused of too much focus on the **process**, not enough on the **product**
- Yes, but **some explicit processes** are required to establish **traceability**
- So we can be sure that it was **this** version of the **code** that passed **those tests**, and they were derived from **that** set of **requirements** which were partly derived from **that** fault tree analysis of **this subsystem architecture**

From Software To System Certification

- The things we care about are **system** properties
- **So certification focuses on systems**
 - E.g., the FAA certifies airplanes, engines and propellers
- **But modern engineering and business practices use massive subcontracting and component-based development that provide little visibility into subsystem designs**
- Strong case for “**qualification**” of **components**
Business case: Component vendors want it (cf. IMA)
Certification case: system integrators and certifiers do not have have visibility into designs and processes
- **But then system certification is based on the certification data delivered with the components**
 - Must certify systems **without looking inside** subsystems

Compositional Analysis

- Computer scientists know ways to do **compositional verification** of **programs**—e.g.,
 - Prove that component **A** guarantees **P** in an environment that ensures **Q**
 - Prove that component **B** guarantees **Q** in an environment that ensures **P**
 - Conclude that $A \parallel B$ guarantees **P** and **Q**
- Assumes programs interact only through explicit computational mechanisms (e.g., shared variables)
- Software and systems can interact through **other** mechanisms
 - **Computational context**: shared resources
 - **Noncomputational mechanisms**: the controlled plant
- So compositional **certification** is harder than **verification**

Unintended Interaction Through Shared Resources

- This must not happen
- Need an **integration framework** (i.e., an architecture) that guarantees **composability** and **compositionality**

Composability: properties of a component are preserved when it is used within a larger system

Compositionality: properties of a system can be derived from those of its components

- This is what **partitioning** is about (or separation in a MILS security context)
 - Except that partitioning may fall short in the presence of faults (e.g., ARINC 653, some avionics buses)
 - **We still lack a good formal definition of partitioning**
 - **And a cost-effective verification methodology for it**

Unintended Interaction Through The Plant

- The notion of **interface** must be expanded to include assumptions about the noncomputational environment (i.e., the plant)
 - Cf. Ariane V failure (due to differences from Ariane IV)
- **Compositional reasoning must take the plant into account** (i.e., composition of hybrid systems)
- Must also consider response to **failures**
- **And must avoid a race to the bottom**

A Science of Certification

- Certification is ultimately a **judgment** that a system is adequately safe/secure/whatever for a given application in a given environment
- But the judgment should be based on as much **explicit** and **credible** evidence as possible
- A **Science of Certification** would be about ways to develop that evidence

Making Certification “More Scientific”

- Favor **explicit** over **implicit** approaches
 - At the very least, expose and examine the claims, arguments and assumptions implicit in standards-based approaches
- Be wary of **qualitative** (implicit) evidence
 - Replace qualitative evidence by analytic evidence that supports sub-claims of a form that can feed into a largely analytic argument at higher levels
- Be wary of **qualitative selections of evidence** (SILs)
 - Rather than qualitatively weakening the evidence, weaken the claims instead, and absorb the resulting hazards elsewhere in the system design

Analytic Evidence

- The move to model based development presents a (once in a lifetime) opportunity to move analytic methods into the early lifecycle, mostly based on formal methods
- Modern **automated formal methods** can deliver **strong claims** about **small properties** very economically
 - Static analysis, model checking, **infinite bounded model checking and k-induction using SMT solvers**, **hybrid abstraction** (which uses theorem proving over reals)
- Larger properties will require combined methods (cf. the **Evidential Tool Bus**)
- The applications of formal methods extend beyond verification and refutation (bug finding): **test generation**, **fault tree analysis**, **human factors**, . . .
- Tool **diversity** may be an alternative to tool **qualification**

Compositional Certification

- This is the big research challenge
- It demands clarification of the difference between verification and certification (because we know how to do the former compositionally, but not the latter)
- And explication of what constitutes an interface to a certified component
 - The certification data is in terms of the interface only
 - You cannot look inside
- Compositional certification should extend to incremental certification, reuse, and modification
- It's also the big challenge for regulatory agencies
 - A completely different way of doing business

A Research Agenda

- The Science of Certification
 - Or a science **for** certification
- Specification and verification of integration frameworks
 - Partitioning, separation, buses
- High-performance automated verification for strong properties of model-based designs
 - Mostly infinite state and hybrid systems

And automation of related processes (test generation, FTA)

- Compositional certification
 - Composition of hybrid systems
- Tool qualification
 - Evidence management
- Integrated methods and arguments
 - Probabilities plus verification