Just-In-Time Certification

John Rushby

Computer Science Laboratory
SRI International
Menlo Park, California, USA

Certification

- Provides assurance that deploying a given system does not pose an unacceptable risk of adverse consequences
- Certification methods should be effective (i.e., they work) and credible (i.e., they work for the reason we think they do)
- Current methods have been effective, but are they credible?
- Current methods of assurance explicitly depend on
  - Standards and regulations
  - Rigorous examination of the whole, finished system

  And implicitly on
  - Conservative practices
  - Safety culture
- All of these are changing

Overview

- Scientific certification
- Compositional certification
- Just-in-time certification

A Recent Incident

- 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

Implicit and Explicit Factors

- See also ATSB incident report for in-flight upset of Boeing 777, 9M-MRG (Malaysian Airlines, near Perth Australia)
- Maybe effectiveness of current certification methods depends on implicit factors such as safety culture, conservatism
- Current business models are leading to a loss of these
  - Outsourcing, COTS, complacency, innovation
- Surely, a credible certification regime should be effective on the basis of its explicit practices
- All assurance is based on **arguments** that purport to justify certain **claims**, based on documented **evidence**
- There are two approaches to assurance: standards-based, and goal-based

The Standards-Based Approach to Software Certification

- E.g., airborne s/w (DO-178B), security (Common Criteria)
- Applicant follows a prescribed method (or processes)
  - Delivers prescribed outputs
    - ⋆ e.g., documented requirements, designs, analyses, tests and outcomes, traceability among these
- Standard usually defines only the evidence to be produced
- The claims and arguments are implicit
- Hence, hard to tell whether given evidence meets the intent
- 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

The Goal-Based Approach to Software Certification

- 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
  - Explicit claims, evidence, argument

Multiple Forms of Evidence

- More evidence is required at higher Levels/EALs/SILs
- What's the argument that these deliver increased assurance?
- Generally an implicit appeal to diversity
  - And belief that diverse methods fail independently
  - Not true in n-version software, should be viewed with suspicion here too
- Need to know the arguments supported by each item of evidence, and how they compose
- Want to distinguish rational multi-legged cases from nervous demands for more and more and . . .
  - Bayesian Belief Networks (BBNs) can formalize these

A Science of Certification

- Certification is ultimately a judgment
- But the judgment should be based on rational argument supported by adequate explicit and credible evidence
- A Science of Certification would be about ways to develop that argument and evidence
- Favor goal-based over standards-based approaches
  - At the very least, expose and examine the claims, arguments and assumptions implicit in standards
- Be wary of demands for more and more evidence, with implicit appeal to diversity and independence
  - Instead favor explicit multi-legged cases
- Use formal ("machinable") design descriptions
  - Can then use automated analysis methods

Systems and Components

- The FAA certifies airplanes, engines and propellers
- Components are certified only as part of an airplane or engine
- That's because it's the interactions that matter and it's not known how to certify these compositionally
- So no alternative to looking at the whole system
- But modern engineering and business practices use massive subcontracting and component-based development that provide little visibility into subsystem designs
- Strong case for "pre-certification" of components

  Business case: Component vendors want it (cf. IMA)

  Certification case: simple extensions to current approach are too onerous or lack credibility (cf. DO-297)

Compositional Analysis

- Computer scientists have ways to do compositional verification of programs—e.g., prove
  - Program A guarantees P if environment ensures Q
  - Program B guarantees Q if environment ensures P

  Conclude that $A \,||\, B$ 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

Composability

Partitioning ensures **composability** of components

- Properties of a collection of interacting components are preserved when they are placed (suitably) in the environment provided by a collection of partitioning mechanisms
- Hence partitioning does not get in the way
- And the combination is itself composable
- Hence components cannot interfere with each other nor with the partitioning mechanisms

Additivity

Partitioning mechanisms compose with each other **additively**

- e.g., **partitioning(kernel)** + **partitioning(network)** provides **partitioning(kernel + network)**
- There is an asymmetry: partitioning network stacks and file systems and so on run as clients of the partitioning kernel

Partitioning (composability and additivity) make the world safe for compositional reasoning

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 extend to take the plant into account (i.e., composition of hybrid systems)
- Control engineers do this, computer scientists are less familiar with it
  - Assumption generation is attractive
- Must also consider response to failures
  - Avoid domino effect
  - Control number of cases (otherwise exponential)

Compositional Certification

- This is a big research challenge
- It demands clarification of the difference between verification and certification, and the role of partitioning
- And explication of what constitutes an interface to a certified component
  - e.g., the notion of interface automata
  - The certification data is in terms of the interface only
  - You cannot look inside when analysing compositions
- 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

Late(r) Binding

- More and more functionality is being determined later than the time at which certification is performed
- E.g., kernel configuration determined at load time
  - 15 KSLOC in certified kernel
  - 50 KSLOC of XML for configuration
- SOA and self-assembly
- AI planning
- Runtime adaptation and learning
- How can these be certified?

Monitoring and Synthesis

- Certification rests on consideration of reachable states
- Scientific certification uses formal methods to calculate and analyze these at design time
- Instead, we could use these methods to construct monitors that check behavior at runtime
    - `www.runtime-verification.org`
- Or to synthesize controllers to generate safe behavior
    - Ramage and Wonham: controller synthesis

Runtime Assurance

- Instead of design-time analysis of implementation
- Use run-time monitoring or synthesis of behavior from models
  - Typically with a receding horizon (bounded lookahead)
  - Fewer possibilities to examine, known current state
- Each component makes its model available to others, pursues its own goals while ensuring that possible moves by others cannot trap it into following a bad path, or cause violation of safety
  - Analyzed as a game: guarantee a winning strategy
- Instead of using model checking and other formal methods for analysis, we use them for monitoring and synthesis

Just-In-Time Certification

- Some of the verification and certification activity is moved from design-time to run-time
- We trust automated verification methods for analysis, so why not trust them for monitoring and synthesis?
  - Certification examines the models, trusts the synthesis
- Will need to consider time-constrained synthesis
  - Anytime algorithms
  - Seek improvements on safe default
- Some analysis methods can deliver a certificate (e.g., a proof), used for synthesis that would truly be just-in-time certification!

A Research Agenda

- A Science of Certification
  - Or the science for certification
- Specification and verification of integration frameworks
  - Partitioning, separation, buses, kernels
- 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
- Just-in-time certification and runtime synthesis