Open Group Paris 23 April 2007, slight revisions of
Open Group San Diego 31 January 2007, major rewrite of
HCSS Aviation Safety Workshop, Alexandria, Oct 5,6 2006
Based on University of Illinois ITI Distinguished Lecture
Wednesday 5 April 2006
based on ITCES invited talk, Tuesday 4 April 2006

Scientific Certification

John Rushby

Computer Science Laboratory
SRI International
Menlo Park CA 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

- See also incident report for Boeing 777, 9M-MRG (Malaysian Airlines, near Perth Australia)
- 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
    - Outsourcing, COTS
- Safety culture is implicit knowledge
- Surely, a certification regime should be effective on the basis of its explicit requirements

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
  - Goals, evidence, claims

Critique of Standards-Based Approaches

- Usually define only the evidence to be produced
- The goals and arguments are implicit
- Hence, hard to tell whether given evidence meets the intent
- E.g., use a "safe programming language (subset)"
    - Misra C: no demonstration of effectiveness, some contrary experience (cf. Les Hatton)
    - Coverity, Prefix etc.: probabilistic absence of runtime exceptions
    - Astrée, Spark Ada (with the Examiner): guaranteed absence of run time exceptions
- And the intent may not be obvious
- E.g., MC/DC testing
    - Is it evidence for good testing or good requirements

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

Two Kinds of Uncertainty In Certification

- One kind is failure of a claim, usually stated probabilistically (frequentist interpretation)
  - E.g., 10^{-9} probability of failure per hour, or 10^{-3} probability of failure on demand
- The other kind is failure of the assurance process
  - Seldom made explicit
  - But can be stated in terms of subjective probability
    - E.g., 95% confident this system achieves 10^{-3} probability of failure on demand
    - Note: this does not concern sampling theory and is not a confidence interval
- Demands for multiple forms of evidence are generally aimed at the second of these

Bayesian Belief Nets

- Bayes Theorem is the principle tool for analyzing subjective probabilities
- Allows a prior assessment of probability to be updated by new evidence to yield a rational posterior probability
- Math gets difficult when the models are complex
  - i.e., when we have many conditional probabilities of the form p(A | B and C or D)
- BBNs provide a graphical means to represent these, and tools to automate the calculations
- Can allow principled construction of multi-legged arguments

Unconditional Claims in Multi-Legged Arguments

- Can get surprising results
  - Under some combinations of prior belief, increasing the number of failure-free tests may decrease our confidence in the test oracle rather than increase our confidence in the system reliability
- The anomalies disappear and calculations are simplified if one of the legs in a two-legged case is unconditional
  - E.g., 95% confident that this claim holds unconditionally
  - Formal methods deliver this kind of claim
  - E.g., Spark Ada (with the Examiner): guaranteed absence of run time exceptions
- Extends to multiple unconditional claims

Rational Safety Cases

- Currently, we apply safety analysis methods (HA, FTA, FMEA etc.) to an informal system description
  - Little automation, but in principle
  - These are abstracted ways to examine all reachable states
- Then, to be sure the implementation does not introduce new hazards, require it exactly matches the analyzed description
  - Hence, DO-178B is about correctness, not safety
- Instead, use a formal system description
  - Then have automated forms of reachability analysis
  - Closer to the implementation, smaller gap to bridge
- Analyze the implementation for preservation of safety, not correctness
  - Favor methods that deliver unconditional claims

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 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 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
- Will be discussed in Thursday's MILS session

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
  - Avoid domino effect
  - Control number of cases (otherwise exponential)

Compositional Design and Development

- Compositional certification will be impossible unless there is a deliberate (and successful!) attempt to control subsystem interactions during design and development
- It's also what's needed for safety: cf. Perrow's tight coupling and high interactive complexity
    - Would be manifested through excessively complex mutual assumptions and guarantees
- The alternative is massive testing at every stage (cf. NASA), and you still have no guarantee of success

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
  - i.e., goal-based over standards-based
  - At the very least, expose and examine the claims, arguments and assumptions implicit in standards-based approaches
- Be wary of demands for multiple forms of evidence, with implicit appeal to diversity and independence
  - Instead favor explicit multi-legged cases
  - Use BBNs to combine legs
  - Favor methods that deliver unconditional claims
- Use formal (“machinable”) design descriptions
  - Automate safety analysis methods
  - Analyze implementation for preservation of safety

Formal Methods (aside)

- Formal methods are not about priestly ways to complicate life
- They are about automated analyses that consider all possible executions
- To make them tractable, may need to approximate
  - Crude: downscaling
  - Principled: predicate abstraction, abstract interpretation, etc
- Most of the action is in improved automation, and automated abstraction

Formal Methods

- 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 unconditional 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

Just-In-Time Certification

- Rather than anticipate all circumstances at design time
- Why not evaluate them at runtime?
  - Maybe with a receding horizon
  - 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 synthesis
  - Ramage and Wonham: controller synthesis
- Certification would examine the models, trust the synthesis

A Research Agenda

- The Science of Certification
  - Or a 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