Runtime Verification Workshop, Budapest, March 2008
Loosely based on FDA Assurance Cases, 21, 22 Feb 2008
Loosely based on 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

Runtime Certification

John Rushby

Computer Science Laboratory
SRI International
Menlo Park, California, USA

Certification

- Certification and high-assurance and have long been supporters and customers of verification technology
- Big changes are under way in these areas
  - I'll describe some of them
- These create new opportunities for runtime verification
  - I'll point out some of these

Current Certification Practice

- Certification provides assurance that deploying a given system does not pose an unacceptable risk of adverse consequences
- Current methods 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

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
- 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

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 flamed out, crew found certain tanks 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)
- How could gross errors like these pass through rigorous assurance standards?
- 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

Standards and Goal-Based Assurance

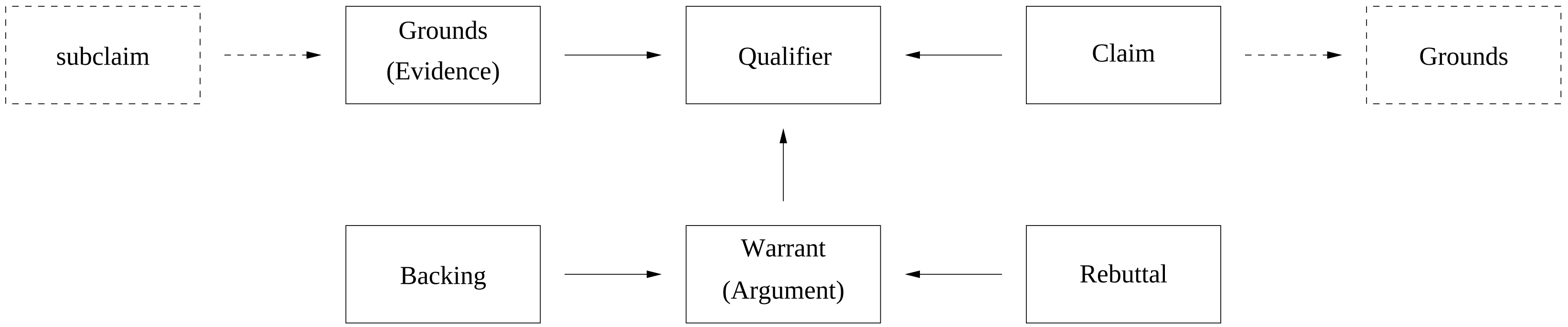
- All assurance is based on **arguments** that purport to justify certain **claims**, based on documented **evidence**
- Standards usually define only the evidence to be produced
- The claims and arguments are implicit
- Hence, hard to tell whether given evidence meets the intent
- E.g., is MC/DC coverage evidence for good testing or good requirements?
- Recently, goal-based assurance methods have been gaining favor: these make the elements explicit

The Goal-Based Approach to Software Certification

- E.g., air traffic management (CAP670 SW01), UK defence
- 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

Toulmin's Model of Argument

- Certification is ultimately a judgement
- So classical formal reasoning may not be entirely appropriate
- Advocates of assurance cases often look to Toulmin's model of argument
- Toulmin stresses justification rather than inference



Toulmin's Model of Argument (ctd.)

Claim: This is the expressed opinion or conclusion that the arguer wants accepted by the audience

Grounds: This is the evidence or data for the claim

Qualifier: An adverbial phrase indicating the strength of the claim (e.g., certainly, presumably, probably, possibly, etc.)

Warrant: The reasoning or argument (e.g., rules or principles) for connecting the data to the claim

Backing: Further facts or reasoning used to support or legitimate the warrant

Rebuttal: Circumstances or conditions that cast doubt on the argument; it represents any reservations or "exceptions to the rule" that undermine the reasoning expressed in the warrant or the backing for it

Reconciling Toulmin's Approach with Formal Methods

- We do formal methods
- So the qualifier is always $\vdash$ or $\models$
- How can we reconcile these with the reasonable doubts manifested in Toulmin's approach?
- One idea
  - Implicit in the work of Jackson and Zave, Goodenough and Weinstock, and others

  Is to put them in the assumptions $A_1, \ldots, A_n$ under which the system S satisfies the requirements R

$$A_1, \ldots, A_n, S \vdash R$$

- Then do subsidiary analysis on each assumption A_i

Analysis of Assumptions

How do we know assumption A_i is valid?

One or more of:

- It is justified as a subsidiary claim
- All our system tests succeeded
- It was not implicated in failed system tests
- Runtime check

Runtime Verification for Assumption Failure

- It is part of the assurance case
- So must be credible (sound, complete, . . .)
- Probably need a sensible recovery action
  - Not like Ariane 501
  - Systematic approaches may be feasible
- What runtime verification methods and specification languages are appropriate?—over to you

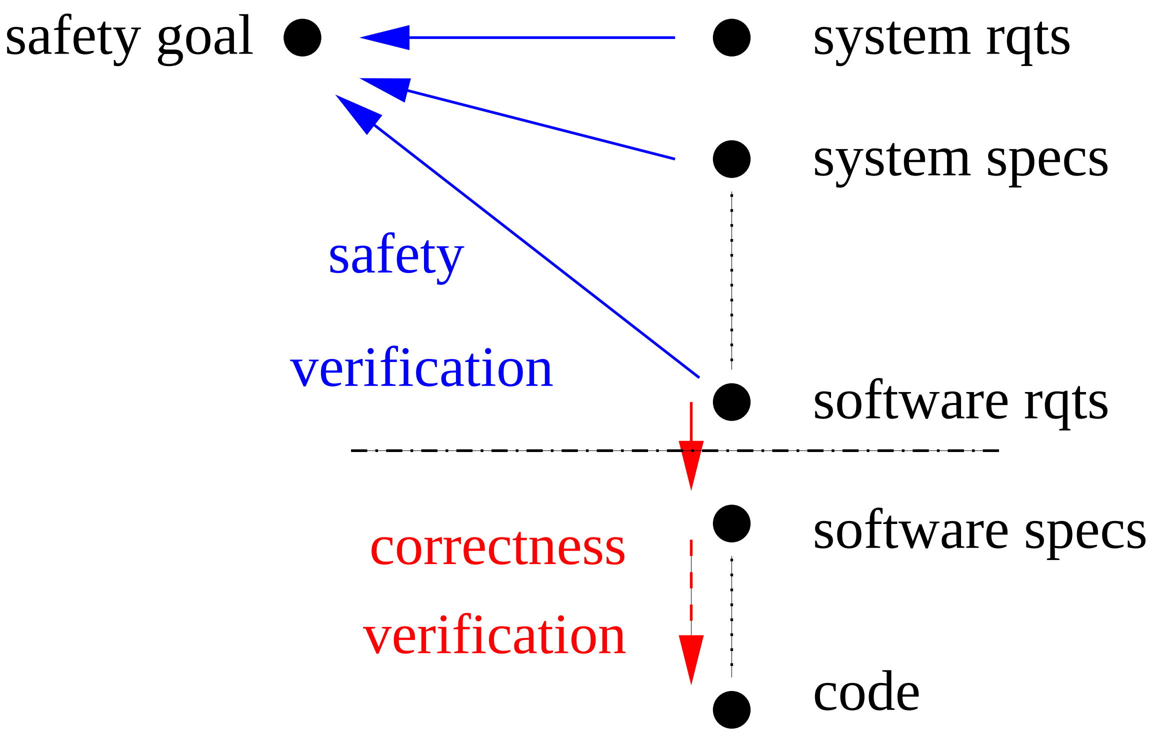
Other Proof Hazards

- The system specification S and requirements R should be analyzed similarly
- And the implementation of the specification
  - Usually a subsidiary claim or claims
- And there's a possibility the proof is flawed
- Or deliberately unsound
  - E.g., static analysis
- Diversity may mitigate this
- Observe this framework provides an uncontroversial and constructive treatment for the hysterical concerns of Fetzer

Implementation Hazards

- 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

Implementation Hazards:
Standards Focus on Correctness Rather than Safety



- Premature focus on correctness is hugely expensive
- Goal-based methods could reduce this
- And runtime verification may be able to check some safety properties directly—over to you

Multi-Legged Arguments

- 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
- Want to distinguish rational multi-legged cases from nervous demands for more and more and . . .
- Need to know the arguments supported by each item of evidence, and how they compose

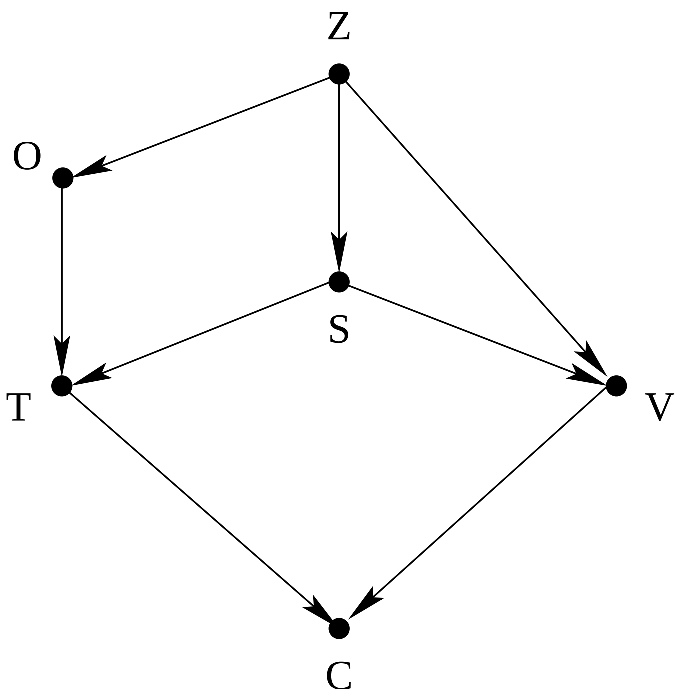
Two Kinds of Uncertainty In Certification

- One kind concerns 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 concerns 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 sources of evidence are generally aimed at the second of these

Bayesian Belief Nets

- Bayes Theorem is the principal tool for analyzing subjective probabilities
- Allows a prior assessment of probability to be updated by new evidence to yield a rational posterior probability
  - E.g., P(C) vs. P(C | E)
- 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 representation for hierarchical models, and tools to automate the calculations
- Can allow principled construction of multi-legged arguments

A BBN Example



Z: System Specification

O: Test Oracle

S: System's true quality

T: Test results

V: Verification outcome

C: Conclusion

Example joint probability table: successful test outcome

Correct System		Incorrect System	
Correct Oracle	Bad Oracle	Correct Oracle	Bad Oracle
100%	50%	5%	30%

Absolute Claims in Multi-Legged Arguments

- Can get surprising results (Littlewood and Wright)
  - E.g., 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 absolute
  - E.g., 95% confident that this claim holds... period
  - Formal methods deliver this kind of claim
- Aside: philosophers studying confirmation theory (part of Bayesian Epistemology) formulate measures of support differently than computer scientists
  - e.g., $P(E \mid C) - P(E \mid \text{not } C)$ vs. $P(C \mid E) - P(C)$

  However, these are related

Practical Considerations

- This approach assumes the verification leg considers the same system description and requirements as the other leg
- But this is seldom the case
  - Verification of weak properties: static analysis etc.
  - Verification of abstractions of the real system
  - Verification of specific critical properties (subclaims)
- Research needed to develop the theory to cover these issues
- And to factor runtime verification methods into the treatment—over to you

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
- But modern engineering and business practices use massive subcontracting and component-based development that provide little visibility into subsystem designs
- Furthermore, the binding times for system architectures and for component behaviors are being delayed to load-time, or even runtime
- So we are forced to contemplate compositional and incremental approaches to certification

Compositional and Incremental Certification

- These are immensely difficult
  - The assurance case may not decompose along architectural lines
- But, in some application areas we can insist that it does
- Need to ensure interactions use only known, intended mechanisms
  - No unprotected IPC channels
  - No signaling through cache occupancy, etc.
  - No unmodeled interaction through the controlled plant
- This is what the MILS approach to security is about
- Other applications, such as spacecraft, medical device plug’n’play, are more difficult

Controlled Interfaces

- If we have successfully controlled what interfaces exist
- The next task is to ensure they are used correctly
- That is, ensure interactions follow their prescribed protocol
- Can be done statically for preplanned compositions
- Or dynamically for opportunistic ones
  - E.g., interface automata
  - With runtime verification—over to you
- But we may still have problems with emergent behavior

Monitoring and Synthesis

- Certification rests on consideration of reachable states
- Science-based 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
- Or to synthesize controllers to generate safe behavior
  - Ramage and Wonham: controller synthesis

Runtime Assurance

- Instead of design-time analysis of the actual 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

Runtime Assurance: Examples

- AI planning
  - Check generated plans
  - Do the generation (cf. bounded model checking)
- Model-based diagnosis and repair
  - Check the diagnoses and proposed repeairs
  - Do the diagnosis and repair generation: cf. qualitative reasoning and hybrid abstraction
- Adaptive control
  - Fixed model, tune the parameters
  - Hybrid systems model checking (box stability etc.)
- CMAC (cerebellum model articulation control)
  - And other connectionist models: discover the model
  - Can possibly synthesize a safe envelope
- Over to you

Runtime 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 runtime certification!

Summary

- Standards-based approaches to certification have run out of steam
- Goal-based certification methods create a framework in which potential contributions of runtime monitoring and synthesis can be better explored and understood
- The big challenges and opportunities are in enabling compositional and incremental certification
- And certifiable runtime adaptation and synthesis
- Runtime verification can make important contributions here
- Some of this material is from "Just-In-Time Certification" and "What Use Is Verified Software?" IEEE ICECCS, Auckland New Zealand July 2007, available at `http://www.csl.sri.com/~rushby/biblio`
- Over to you