Verified ~~Software~~ Systems Roadmap
The Certification Perspective

John Rushby

Computer Science Laboratory
SRI International
Menlo Park, California, USA

External Presentation of the Roadmap

- How should we explain the VSR to the (paying) public?
- And what motives should guide our own internal agenda?
- Science for its own sake
  - A search for knowledge without utilitarian motives
  - Though it is anticipated to have beneficial impact
- Or an engineering program
  - Primarily intended to improve the quality of software
  - The approach is one selected by scientists and engineers
- In either case, we need to connect the proposed roadmap to credible claims for societal benefit

Societal Benefit

- Better software doesn't cut it
  - Software doesn't impact the world
  - It's the embedding of software in systems that does that
- So let's say it's about better software-intensive systems
  - i.e., pretty much any kind of system encountered today
- What does better mean?
  - Improving the positives
    - ⋆ More features, faster time to market, lower cost
  - Or reducing the negatives
    - ⋆ Failures, unreliability, dysfunction, insecurity, cost overruns, delays, infelicities

  It's the difficulty of controlling the negatives that limits our ability to improve the positives
- So better means we aim to reduce the negatives

Certification

- When potential negatives could endanger life, the environment, national security, or corporate assets
- Then we typically enter the realm of certification:
  - Judgment that a system is adequately safe for a given application in a given environment
- Can substitute secure, or whatever, for safe
  - Invariably these are about absence of harm
- Generically, certification is about controlling the negatives or downsides of system deployment
- So the benefit sought by VSR is closely associated with the goal of certification
- And we should relate VSR to the best practices of that field
  - But without the implication of regulation

The Practice of Certification

- Judgment that a system is adequately safe/secure/whatever for a given application in a given environment
- Based on a documented body of evidence that provides a convincing and valid argument that it is so
- Generically, certification is about controlling the downsides (negatives) of system deployment, so we need
  - To know what the downsides are
  - And how they could come about
  - And we need to have controlled them in some way
  - And to have credible evidence that we've done so

"System is Safe for Given Application and Environment"

- It's a system property
  - e.g., the FAA certifies only airplanes and engines (and propellers)
- And it's about the development lifecycle, not just the code
- And the application and environment must be considered
- Some fields do this at a separate stage
  - e.g., security: evaluation vs. certification
  - Evaluation can be seen as subsystem certification
- Others combine them
  - e.g., assumptions about how passenger planes fly—such as no aerobatics—is built in to their certification
    - ⋆ Though Tex Johnston did a barrel roll (twice!) in a 707 at an airshow in 1955

View From Inside Inverted 707

During Tex Johnston's barrel roll

Knowing What The Downsides Are

- Derives from the system context
  - And recursively down through subsystems
- Institutionalized in regulated contexts
  - e.g., "inability to continue safe flight and landing"
- But could be proposed for many systems
  - "Won't lose Granny's photos once they've been stored"
- It would revolutionize many fields if developers were expected to make explicit claims of this sort

Knowing How The Downsides Could Come About

- The problem of "unbounded relevance" (Anthony Hall)
- There are systematic ways for trying to bound and explore the space of relevant possibilities
  - Hazard analysis
  - Fault tree analysis
  - Failure modes and effects (and criticality) analysis: FMEA (FMECA)
  - HAZOP (use of guidewords)
- Industry-specific documents provide guidance
  - e.g., SAE ARP 4761, ARP 4754 for aerospace

Controlling The Downsides

- Downsides are usually ranked by severity
  - e.g. catastrophic failure conditions for aircraft are "those which would prevent continued safe flight and landing"
- And an inverse relationship is required between severity and frequency
  - Catastrophic failures must be "so unlikely that they are not anticipated to occur during the entire operational life of all airplanes of the type"

Subsystems

- Hazards, their severities, and their required (im)probability of occurrence flow down through a design into its subsystems
- The design process iterates to best manage these
- And allocates hazard "budgets" to subsystems
  - e.g., no hull loss in lifetime of fleet, 10^7 hours for fleet lifetime, 10 possible catastrophic failure conditions in each of 10 subsystems, yields allocated failure probability of 10^{-9} per hour for each
- Another approach could require the new system to do no worse than the one it's replacing
  - e.g., in 1960, big jets averaged 2 fatal accidents per 10^6 hours; this improved to 0.5 by 1980 and was projected to reach 0.3 by 1990; so set the target at 0.1 (10^{-7}), then subsystem calculation as above yields 10^{-9} per hour again

Software Subsystems

- Hazards flow down to establish properties that must be guaranteed, and their criticalities
  - Unrequested function
  - And malfunction
  - Are generally more serious than loss of function
- How to establish satisfaction of such requirements?

Approaches to System and Software Certification

The implicit standards-based approach

- e.g., airborne s/w (DO-178B), security (Common Criteria)
- Follow a prescribed method
- 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 (in progress)

But less suitable when novelty in problems, solutions, methods

Implicit that the prescribed processes achieve the safety goals

Approaches to System and Software Certification (ctd.)

The explicit goal based approach

- e.g., aircraft, air traffic management (CAP670 SW01), ships

Applicant develops an assurance case

- Whose outline form may be specified by standards or regulation (e.g., MOD DefStan 00-56)
- The case is evaluated by independent assessors

An assurance case

- 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

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/Indirect: establish only indirect links from evidence to desired attributes

- 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 indirect
  - 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 many 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) required at some EAL levels
  - MC/DC tests required for DO-178B Level A

  Little evidence which are effective, nor that more is better

- "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)

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

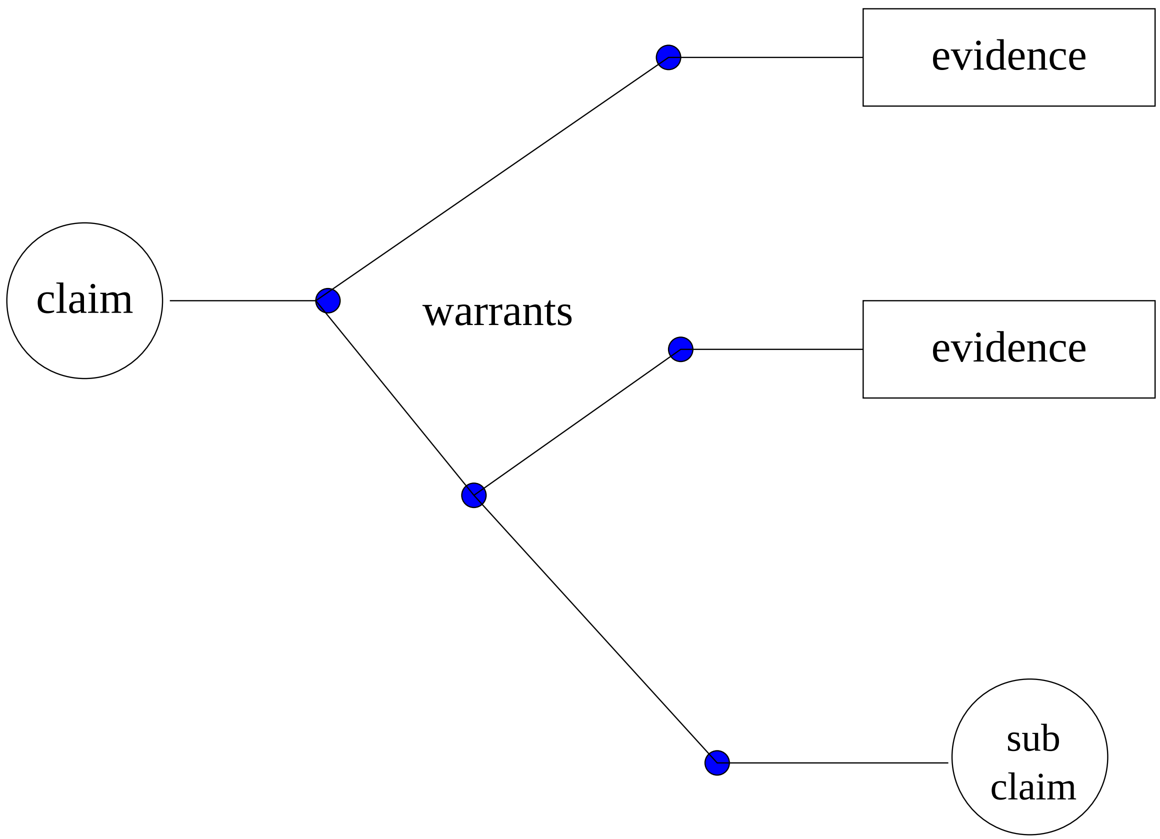
Making Certification "More Scientific"

- 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 indirect arguments
  - Replace qualitative arguments by analytic arguments that support 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

Toulmin Arguments

- The local argument that links some evidence to the claim is called a warrant
- Not all the warrants in an assurance case are of the strictly logical kind
  - e.g., "experience suggests that. . . "
  - This should not be a backdoor to qualitative evidence and indirect arguments—the warrant needs to establish a "causal link" between the evidence and the claim
- So the overall style of argument is not of the kind considered in classical (formalized) logic
- Advocates of explicit assurance cases generally look to Toulmin for guidance on argument structure
  - "The Uses of Argument" (1958)
  - Stresses justification rather than inference

Argument Structure for an Assurance Case



The VSR and Scientific Certification

- Verification should be about supplying analytic warrants in support of explicit assurance cases
- And that means throughout the assurance case
  - Automated formal support for FMEA, human interaction errors, and other aspects of hazard analysis
  - Formal requirements specifications
  - Automated formal design verification
    - ⋆ Exposes assumptions that feed upper levels of analysis
  - Static analysis for absence of runtime errors
  - Full program verification against detailed specifications
  - Automated formal test case generation
- And maybe even to support the overall assurance case
  - Replace Toulmin?

The VSR and Scientific Certification (ctd 1)

- Observe that all of these have formal content but rather few of them involve classical proofs of correctness
- So we will need versatile and diverse tools that can be used in opportunistic combination
- And even proofs of correctness will use lemmas derived by other means
  - e.g., static analysis
- Risks will be assessed, allocated, and controlled in the overall assurance case, so it should not be assumed that proofs must be reduced to trivial steps
  - Diverse analyses may be preferable
- Nor even that everything must be proved
  - Statistically valid testing is good to about 10^{-4}

The VSR and Scientific Certification (ctd 2)

- Formal methods are the applied math of software and computer systems engineering
- Just as PDEs are the applied math of aerodynamics
- Automated formal methods perform logical calculations, just as CFD performs numerical calculations
- VSR should approach automated formal methods as other engineering disciplines approach their computational models and tools
  - Focus on delivering maximum value in the overall process
  - Soundness is important, and so are performance and capacity: cannot sacrifice one for the other

The VSR and Scientific Certification (ctd 3)

A full science of certification will need to be

Hierarchical: the Toulmin level

Incremental: can add components or properties to a certified system without having to revisit the whole thing

Compositional: can calculate the properties and the assurance case for a system from those of its components

These remain major research challenges

The last of these is the touchstone: can safely use the formal description of the artifact when building other artifacts

The VSR Certification Roadmap

Yves Bertot, Helen Gill, Connie Heitmeyer, Jim Horning, Warren Hunt, John Rushby, Hassen Saïdi, Ashish Tiwari, Tomas Uribe

Preliminary Certification Roadmap

1–2 Years:

- Formal methods employed in development, certification
- Local victories (e.g., verification replaces MC/DC testing)

5 Years:

- Scientific certification methods developed for core safety case activities across all fields
- Automated formal methods employed in support of scientific certification

10 Years:

- Automated formal methods employed top to bottom in support of scientific certification, and incrementally

15 Years:

- Compositional certification