New Directions in V&V
Evidence, Arguments, and Automation

John Rushby

Computer Science Laboratory
SRI International
Menlo Park, California, USA
V&V for Fault Management

Ideally, we’d like to understand, consider, examine, test

- all possible behaviors

Which raises some interesting issues

- Define all possible

- However you define it, that’s a lot of behaviors

- How can we handle that many?

- Can we do it subsystem by subsystem?

- Can we start the work early?

We need a framework and some technology and a methodology
Existing Frameworks for V&V

- V&V and the larger processes of certification/approval provide assurance that deploying a given system does not pose an unacceptable risk of failure or adverse consequences.

- Current methods explicitly depend on
  - Standards, regulations, process
  - 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 Assurance

- E.g., airborne s/w (DO-178B), security (Common Criteria)
- Developer 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

John Rushby, SRI

V&V: Evidence, Arguments, Automation 5
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 methods depends on implicit factors such as safety culture, conservatism

- Current business/contracting models and mission ambitions are leading to a loss of these
  - Outsourcing, COTS, complacency, innovation, complexity

- Surely, a credible certification regime should be effective on the basis of its explicit practices

- How else can we cope with the changes and challenges ahead?

John Rushby, SRI

V&V: Evidence, Arguments, Automation 6
Standards and Goal-Based Assurance

- All assurance is intellectually 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 Assurance

• E.g., UK air traffic management (CAP670 SW01), UK defence (DefStan 00-56), growing interest elsewhere

• Developer provides an assurance case
  ○ Whose outline form may be specified by standards or regulation (e.g., 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

• Can be specialized to safety, security, dependability cases

• The case is evaluated by independent assessors

• Key point: explicit claims, evidence, argument

John Rushby, SRI

V&V: Evidence, Arguments, Automation 8
Assurance Cases Allow Customization

- Standards such as DO-178B focus on correctness
  - i.e., on verification more than validation

- Whereas assurance cases liberate us to customize our V&V

John Rushby, SRI
System-Focused Claims

- **Goal-based** assurance cases are driven by *risk assessment*
- Focus on hazards, risks, and their mitigations
- At the *system level*
- Flow down into subsystems and allow *prioritization*
- **Multi-legged** cases allow evidence for testing, say, to be *combined* with analysis in a rational way using *Bayesian Belief Nets* (BBNs)
A BBN Example

Z: System Specification

O: Test Oracle

S: System’s true quality

T: Tests

V: Analysis

C: V&V decision

Example joint probability table: successful test outcome

<table>
<thead>
<tr>
<th>Correct System</th>
<th>Incorrect System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct Oracle</td>
<td>Bad Oracle</td>
</tr>
<tr>
<td>100%</td>
<td>50%</td>
</tr>
<tr>
<td>5%</td>
<td>30%</td>
</tr>
</tbody>
</table>

John Rushby, SRI

V&V: Evidence, Arguments, Automation 11
Technology and Automation

- Goal-based assurance cases give us a framework to approach V&V in a customized but rational way, focusing on system-level hazards.

- Traditional methods for assurance at the systems level, such as hazard analysis (HA), FMEA, FTA, HAZOP.

- Are really abstracted (i.e., approximate) ways to do reachability analysis.
  - Enumeration of all the states that a system can get into through interaction with its environment.

- In other words, they are ways of exploring all possible behaviors.

- How about if we could do this for more detailed levels of design?

John Rushby, SRI
V&V: Evidence, Arguments, Automation 12
Informal Reachability Analysis

- Given a system model made up of interacting state machines
- i.e., the software design, hardware components
  - And the environment
  - Which can inject faults (think of it as the test harness)
- Work forward from the initial states to see if you can reach a state where something bad happens (HA)
- Or work back from the bad states to see if you can reach an initial state (FTA)
- Made feasible to do by hand by focusing on only certain transitions (FMEA)
- And by using abstracted models (HAZOP)
- But suppose we could automate it?
Automated Reachability Analysis

- We need “machinable” models of the system and its environment; not PowerPoint pictures, not code
- E.g., Statecharts, UML, AADL, Simulink/Stateflow
- If we “downscale” these to finite state
  - E.g., discretize continuous values
- Then we can do brute-force reachability analysis
- By running or simulating the system, backtracking to take alternate paths, and remembering where we have been
- This is what an explicit state model checker (e.g., Spin) does
- Can handle tens of millions of reachable states
- Gives counterexample when an error found
- Errors defined by observer models, or property language
Formal Reachability Analysis

- Ten million states is only 23 or 24 state bits
- Symbolic methods of reachability analysis can often handle bigger systems... trillions of states, even infinite
- By representing states as formulas rather than explicit values
  - e.g., $x < y$ represents an infinite number of explicit states: $(0,1), (0,2), ... (1,2), (1,3)...$
- Symbolic model checkers (e.g., nuSMV, SAL)
  - Use Binary Decision Diagrams (BDDs)
- Bounded model checkers (e.g., nuSMV, SAL)
  - Use Boolean satisfiability (SAT) solvers
- Infinite bounded model checkers (e.g., SAL)
  - Use solvers for satisfiability modulo theories (SMT)
- BDDs, SAT, SMT solvers are commodities
Reachability Analysis for Fault Management

- Construct state machine models for components, environment, the FM algorithms (e.g., monitors and responses) in some modeling notation

- Connect a model checker to the modeling tool set
  - E.g., Mathworks' own Design Verifier for Simulink/Stateflow
  - Or build your own—as Rockwell has

- And you will absolutely find large numbers of issues such as those described for New Horizons fault management, or Space Station architecture with negligible effort

- Find vastly more problems by examining all the behaviors of a simplified model than by testing some of the behaviors of the real thing
A Spectrum of V&V Activities

A wealth of opportunities to the left; can apply them early, too.

Number of cases examined

Fidelity of model

new opportunities

current practice

state machines models simulations h/w in loop flight h/w

John Rushby, SRI

V&V: Evidence, Arguments, Automation 17
Reachability Analysis for Fault Management V&V

- V&V is more than debugging
- Want to make strong inference when the model checker no longer finds bugs
- Requires judgement in modeling
  - Often less is more: constraints rather than details
- And more sophisticated automation (research topics)
  - K-induction rather than bounded model checking
  - Counterexample-guided abstraction refinement (CEGAR)
  - Hybrid systems (state machines plus differential equations)
- And we need ways to keep different models, simulations, real system in sync

John Rushby, SRI
Test Automation for Fault Management V&V

- The counterexamples from model checkers can be used to generate test cases to run on the implementation
  - Tests can target model coverage, corner cases, specific kinds of scenarios: focus shifts from constructing tests to specifying test objectives
- Unit tests are pretty easy to generate automatically
- Integration tests are more challenging
  - Depends how much control you have of other components
- Hardware in the loop is more difficult still (research)
  - Some of the models are hybrid systems
- Automation can be used to extend random tests into corners
  - There are very potent mixed concrete symbolic (concolic) methods

John Rushby, SRI
From Analysis to Synthesis

- The same reachability methods we use to analyze monitor-response fault management rules
- Could be used to synthesize the rules
  - **Supervisory controller synthesis (Ramadge and Wonham)**
  - Set up as a game between fault management and the environment
    - Use reachability analysis to synthesize rules so that from any state, no move by the environment can force us into a losing state
- Could be used **statically** on the ground
- Or **dynamically** onboard the spacecraft (next talk)
Vee Diagram Tightened with Formal Analysis

Example: Rockwell-Collins

John Rushby, SRI

V&V: Evidence, Arguments, Automation 22
Systems and Subsystems

- The FAA certifies airplanes, engines and propellers

- Components and subsystems 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 provide assurance for these compositionally

- But modern engineering and business practices use massive subcontracting and component-based development that provide little visibility into subsystem designs

- So we are forced to contemplate compositional and incremental approaches to assurance and V&V

- Manifestation of noncompositionality in FM is the need to run tests for days or weeks to get into interesting states
Compositional and Incremental Assurance

- Compositional assurance means deriving the assurance case for the system from those of its subsystems.
- Without going into all the subsystem details.
- It is difficult because
  - The assurance case may not decompose along architectural lines.
- Spacecraft have inherent subsystem coupling (through the plant).
- But we should surely eliminate unnecessary coupling
  - Computer to computer and bus communication issues
  - Partitioning
  - Information hiding interfaces
Computer to Computer and Bus Communications

- **It's easy to mess these up**
  - Bad fault modes (babbling—e.g., Clementine)
  - Timing (e.g., recent spysat?)

- **It is known how to do it right** (e.g., TTA, SPIDER)

- **These are more than just buses—they are frameworks for integration**

- That is, they facilitate compositional design
Integration Framework Anecdotes

**Powertrain integration**: car engines from one plant, gearboxes from another
- Typically *months* of work to get them to work together
- A few *hours* using TTA

**Multi-channel FADEC integration**: get single channel working, then add second channel
- Typically *months* of work to get both channels cooperating
- A few *hours* using TTA

*Assurance benefits beyond those in integration*
Partitioning

- Subsystems may share processor resources
- Don’t want a fault in one subsystem to wreck others
  - By messing with its state, timing, etc.
- Integrated modular avionics (IMA) for aircraft use
- Partitioning RTOSs
- Similar RTOSs (but with higher assurance, called separation kernels) used in embedded applications for high security
- Again, best seen as integration frameworks rather than just protection mechanisms
Information Hiding Interfaces

- Partitioning buses and RTOSs prevent propagation of faults
- And have the side effect of facilitating compositional design
- By eliminating unintended interactions and coupling
- We need to do this throughout the design
  - “Complexity containment regions”
  - That’s what interfaces are
  - And architecture at a higher level
Information Hiding Interfaces: Sensor Example

- Typically, send raw sensor samples with timestamp
- To integrate multiple samples, need to know the fault-status and detailed behavior of each sensor
- Use complex variants of mid-value select to mask faults
- Instead, we could use intelligent sensor (knows its own status, does local diagnosis)
- Sends sample as an interval: true value guaranteed to be somewhere inside (if nonfaulty)
  - Narrow interval when healthy, good sample; wider if not
- With a “use by” date
- Known how to combine intervals, even when some are faulty
- System does not need to know subsystem details

John Rushby, SRI

V&V: Evidence, Arguments, Automation 29
True Value In Overlap Of Nonfaulty Intervals

\[ S(1) \quad S(2) \quad S(3) \quad S(4) \]
Compositional V&V

- Reachability analysis with a model checker examines whether interacting components satisfy some requirement
  - e.g., device, control, environment $\models$ requirement

- We can try to find the weakest model $D$ for the device that still does the job (might have to adjust control)
  - i.e., $D$, control', environment $\models$ requirement

- Then, later, show that the real device satisfies $D$
  - i.e., device $\models D$

- So reachability tools can help develop interfaces that promote compositional assurance
Summary

- If we want to improve cost and effectiveness of V&V, we need a framework to help us rethink it
  - Goal based assurance cases are a promising framework
  - Explicit claims, evidence, argument

- Model-based design opens the door to reachability analysis
  - aka. model checking, formal methods
  - This is automated, can be done early, examines vast numbers of behaviors including interactions
  - Preserves the valuable high-fidelity testbed

- Strong interfaces promote compositional assurance
  - Reachability analysis can help develop these

- Autonomy is surely the way of the future; let’s get the V&V right (reliable, early, affordable; enabler, not impediment)