AIAA GNC Conference 19 August 2008, Honolulu conf center, based on
Kickoff for "Formally Supported Safety Cases For Adaptive Systems", NASA LaRC, 9 April 2008
Uses 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

How Do We Certify For The Unexpected?

John Rushby

Computer Science Laboratory
SRI International
Menlo Park CA USA

Project Background

- New 3-year project under IRAC
  (Integrated Resilient Aircraft Control)
- Started January 2008
- Long title: Multi-legged safety cases for adaptive systems supported by formal methods and mechanized analysis
- Synergistic with NSF grant on Just-In-Time Certification
  - Paper of that title at ICECCS 2007 (Best Paper Award)
  - Informal collaboration with Adelard/City University in UK

Technical Context

- General focus is adaptive systems
- These range from those that tune control parameters within a fixed model (e.g., to best suit this airplane)
- To full-blown model-free learning
  - E.g., Neural nets, or
  - Cerebellar Model Articulation Control (CMAC)
- In unanticipated circumstances
  - E.g., Loss of control due to major damage, unusual attitudes, design faults

  Which is the focus of this session
- Health Monitoring and Management are in the same space
- How do we certify these?

Certification Context

- It's not productive to view a neural net, say, as merely a different means for implementing software
- And then to try to apply DO-178B to it
- It's a more radical change than that
- And also representative of other changes
- That challenge the assumptions of traditional certification
- For example, "delayed binding"

Delayed Binding

- Adaptive systems are an extreme point in a continuum
- Of innovations that delay the time at which the final configuration of flight software is determined
  - Reconfiguration in response to faults
  - Load-time configuration
  - Separate construction and assembly (IMA)
  - Reuse and COTS
- All of these challenge the idea that the flight software is developed specifically for this application and is available in its final completed form for certification well in advance of deployment
- And then there's NGATS

Frameworks for Certification

- 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
- How else can we cope with adaptive systems?

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

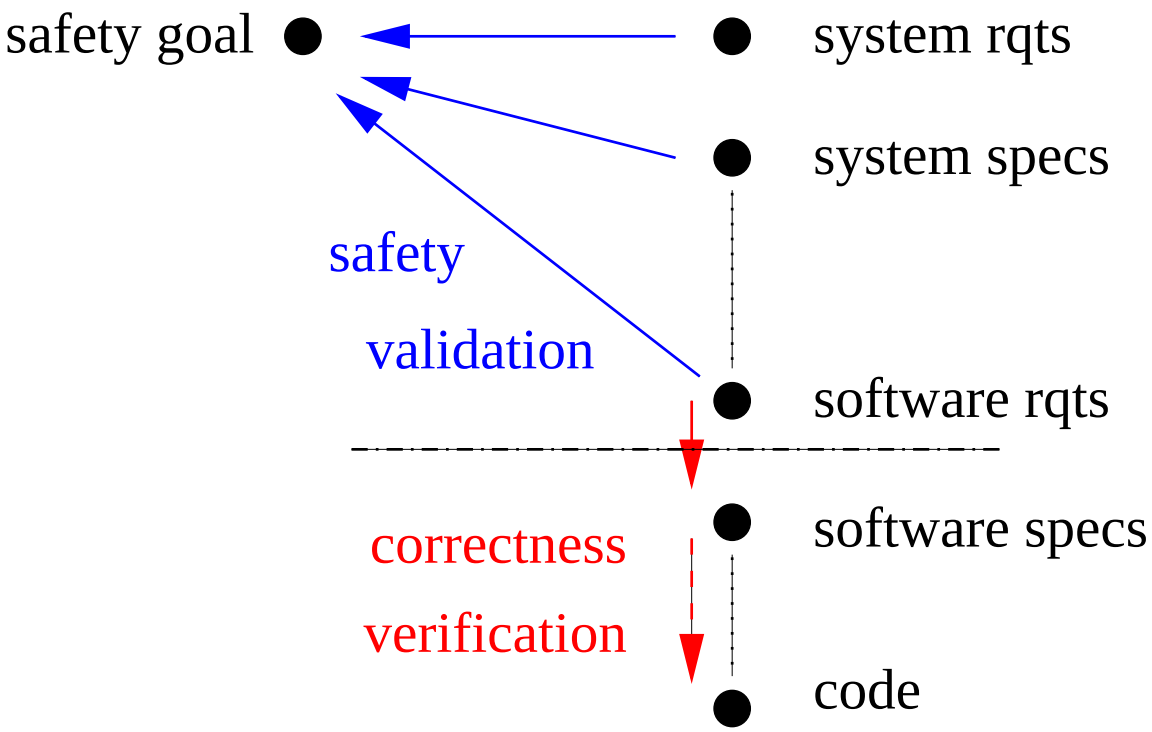
The Goal-Based Approach to Software Certification

- E.g., UK air traffic management (CAP670 SW01), UK defence (DefStan 00-56), growing interest elsewhere
- Applicant develops a safety 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
- Generalized to security, dependability, assurance cases
- The case is evaluated by independent assessors
  - Explicit claims, evidence, argument

Relation to Current Practice

- Fairly consistent with top-level certification practice
- Applicants propose means of compliance
  - cf. ARP4754, ARP4761
  - Apply safety analysis methods (HA, FTA, FMEA etc.) to an informal system description
- And a Plan for Software Aspects of Certification
  - Typically DO-178B
  - To be sure implementation does not introduce new hazards, require it exactly matches analyzed description
    - ⋆ Hence, DO-178B is about correctness, not safety
- It's the latter that we propose to change
  - Analyze the implementation for preservation of safety, not correctness
  - This may be a way to deal with adaptive systems

Software Hazards:
Standards Focus on Correctness Rather than Safety



- Premature focus on correctness inappropriate for adaptive systems, goal-based methods could reduce this

Runtime Monitoring

- Adaptive systems move some of the system design to runtime
- So shouldn't some of the assurance case do likewise?
- There's a lot of empirical experience on the utility of runtime monitoring (e.g., paper here on Monday morning)
- But what are good properties to monitor?
- Monitoring requirements is not likely to be effective
- Too local: seldom violated even when system is in a bad state
  - Cf. the A340 example
- And DO-178B is pretty good at ensuring bugs don't violate them
- We need a more independent source of properties

Monitoring Assumptions

- In a goal-based safety case, we have an argument or proof that system S satisfies the claims C under assumptions $A_1, \ldots, A_n$

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

- And then subsidiary analysis on each assumption A_i
- Assumptions provide good properties to monitor
  - Though not all are readily sensed
  - e.g., fault assumptions (cf. 777 incident)
- Runtime verification is a subfield of formal methods
  - Automated, correct synthesis of monitors for formally specified properties

Runtime Assurance for Adaptive Systems

- Modest approach: use runtime monitoring for assumptions in safety case
  - Failure triggers higher-level recovery
  - Formal runtime verification delivers strong evidence
- Ambitious approach: extend formally-based synthesis from monitors to active controllers
  - Synthesize controllers to guarantee safe behavior
  - This is Ramage and Wonham: controller synthesis
  - Analyzed as a game: guarantee a winning strategy
  - Mechanized by techniques related to model checking

Runtime Checking and Synthesis: Examples

- AI planning
  - Check generated plans
  - Do the generation (cf. bounded model checking)
- Model-based diagnosis and repair
  - Check the diagnoses and proposed repairs
  - Do the diagnosis and repair generation: cf. qualitative reasoning and hybrid abstraction
  - Can include the operators' mental model to diagnose automation surprises and to help keep them in the loop
- Adaptive control
  - More difficult: hybrid systems models

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 would examine the models, trust 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!

Current and Planned Activity

- Obtain/develop some paradigmatic safety cases
  - Mostly medical: pacemaker, M7 surgical robot

  To be used for experimental evaluations
- Additional class of monitors based on anomalies
  - Warns when outside scenarios seen in test
- Goals, evidence, argument, and confidence
  - Emerging understanding that verified claims of perfection are independent of empirical reliability claims
  - Hence confidence in verification (as subjective probability) can be a multiplying factor in overall reliability estimate
  - Feeds into multi-legged BBN analyses
- Compositional certification