AIAA Infotech@Aerospace, Seattle April 2009, loosely based on
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

A Safety-Case Approach For Certifying Adaptive Systems

John Rushby

Computer Science Laboratory
SRI International
Menlo Park CA USA

How Can We Certify Adaptive Systems?

- Implemented in software, so Means of Compliance will have to deal with Software Aspects of Certification
- Conventional systems generally cite RTCA/DO-178B
- Lays great stress on determinism and predictability
- Wants complete requirements specification, and demonstration that implementation correctly performs all and only the specified behaviors
- But adaptive systems intentionally leave some behaviors to be learned or tuned in flight
- So we need an Alternative Means of Compliance

Standards and Argument-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, argument-based (formerly called goal-based) assurance methods have been gaining favor: these make the elements explicit

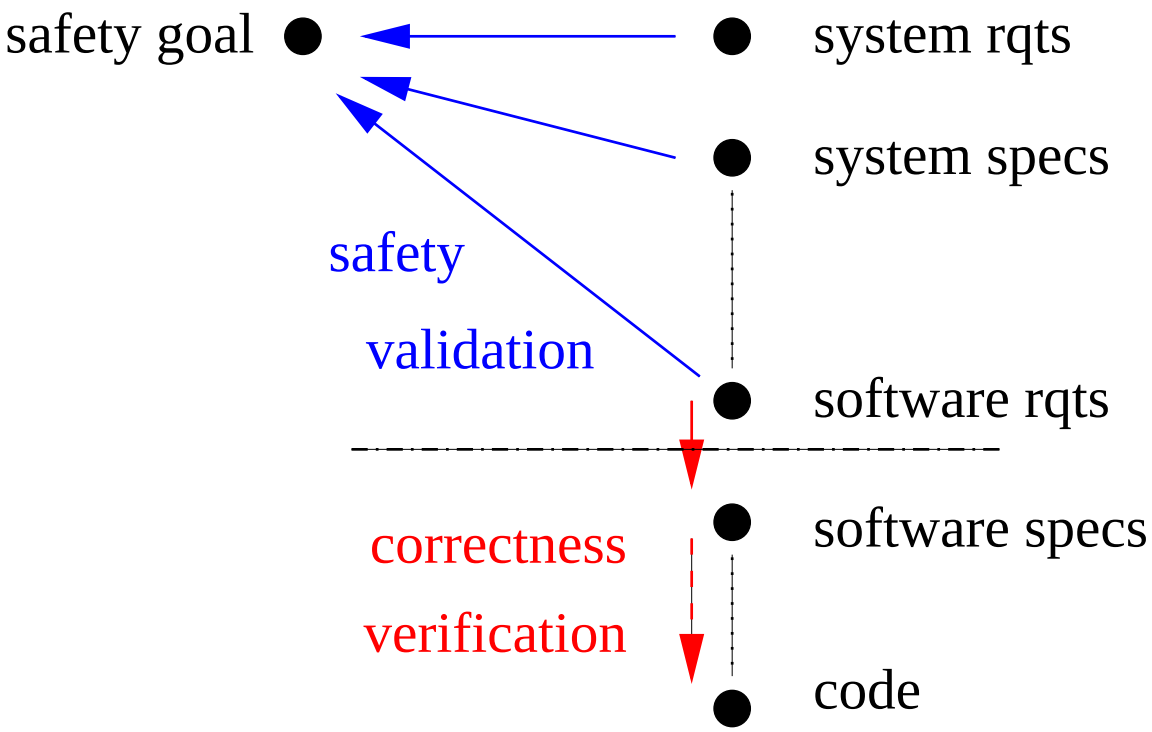
The Argument-Based Approach to Software Certification

- 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

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
  - 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
  - Not totally antithetical to DO-178B (could go in Accomplishment Summary)

Software Hazards:
Standards Focus on Correctness Rather than Safety



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

Full-Time or Part-Time Adaptation?

- If the adaptive controller is full-time, need some strong evidence that it is safe
- E.g., prove that it works for any values of the learned parameters
  - i.e., learning is about performance not safety
- Ashish Tiwari has techniques that can do this (paper next week at CPS Week in San Francisco)
- If it's part-time, may have lesser hurdle if it's only activated when the airplane is otherwise doomed
- But then have extreme concern about entry to that mode

American Airlines Flight 903

- 12 May 1997, an Airbus A300
- Anomaly detection and mitigation mechanisms reset the EFIS (flight displays) bus
- Because the indicated roll rate of more than 40 degrees/second was considered implausible
- In fact, the pilots were attempting recovery from a major upset and the roll rate was real
- Loss of all instruments at this critical time jeopardized the recovery

Runtime Monitoring

- Likely to use monitoring to trigger entry into adaptive mode
- So 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 FCMC example (later)
- And DO-178B is pretty good at ensuring bugs don't violate them
- We need a more independent source of properties

Monitoring Assumptions

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

Possibly Perfect Components

- Usually we make claims about reliability
  - e.g., probability of failure on demand
- But for formal monitors, we could claim they are perfect
  - i.e., will never experience a failure
- But would an assessor believe this?
- Might accept high confidence in perfection, so speak of possibly perfect components
  - Then have probability of imperfection
  - Of a large number of components designed this way, the proportion that will ever experience a failure
- Littlewood and Rushby show reliability of one channel and possible perfection of a second are conditionally independent at the aleatory level: hence can multiply the probabilities
- Then need to get epistemic estimates of these probabilities

Possibly Perfect Monitors

- Skipping a lot of details, get an expression for risk in a monitored system

$$\text{risk} \leq f \times c_1 \times (C + P_{A1} \times P_{B1}) + (1 - f) \times c_2 \times P_{B2}$$

- Here, first (blue) term is where system fails to change to adaptive mode when it should, and second (red) is where it changes when it should not
- And f is proportion of flights where adaptive mode is needed, c_1 is cost of not adapting when you should, c_2 is cost of adapting when you shouldn't, P_{B2} is probability of imperfection of the monitor wrt. undesired triggering
- Now $1 - f$, is close to unity, c_2 is presumably large, so P_{B2} must be very small indeed
- Likeliest source or imperfection is the safety case is flawed, so correctly monitor the wrong properties

Flawed Safety Cases

- The abstract argument for safety cases is compelling
- But real cases are large and complex
- How can you tell a good case from a flawed one?
- Prescriptive processes like DO-178B at least establish a floor
- Talks today discussed quality of evidence (e.g., color coding)
- But how do you assess the argument?
- John Knight of UVA has found several example cases to be flawed

Assurance for Arguments

- Notations like GSN
  - Goal Structuring Notation (Univ. of York, UK)

  and CAE
  - Claims, Arguments, Evidence (City Univ/Adelard, UK)

  help visualize an argument
- And tools like ACSE
  - Adelard Safety Case Editor

  let you explore it
- And there are OMG standards in development for evidence and argument
- But none of this helps you validate or verify an argument

Verifying Arguments

- Traditionally a role for interactive theorem provers
- But these require guidance and notations outside the experience and skill set of engineers
- Model Based Design (MBD) tools like Simulink can do some of this
  - Simulink Design Verifier is a model checker/theorem prover for Stateflow/Simulink
  - It's a standard Mathworks product, mainly developed by Gregoire Hamon
- But these deal with models that are too detailed for most safety analysis
- Want similar state machine and combinational models
- But much more abstract

Abstract Arguments and SMT Solvers

- Can express a lot, very abstractly, in terms of uninterpreted types and functions, with assumptions expressed as axioms
- SMT solvers are fully automated software tools for the problem of checking Satisfiability Modulo Theories
- One of the theories is equality and uninterpreted functions
- Annual competitions keep SMT Solvers hot: can solve problems with thousands of variables and hundreds of thousands of constraints in seconds or minutes
- Bug finding, test generation, and verification problems for state machine models over uninterpreted functions are solved by k-bounded model checking and k-induction
- Note these methods analyze all possible behaviors
- And are solved by SMT solvers

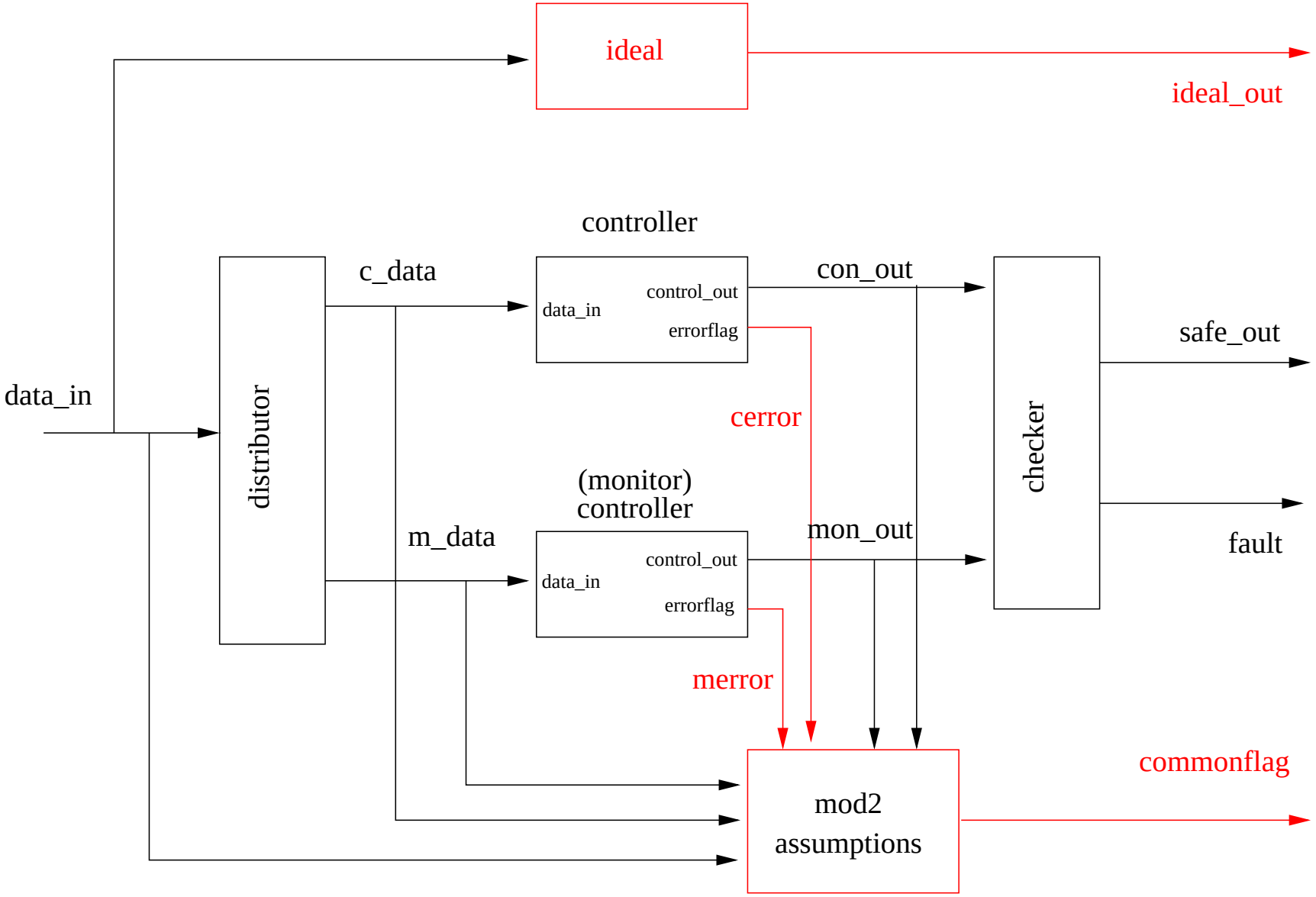
Example

- 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

A340 FCMCs

- Incident report describes the design in a fair amount of detail
- Two FCMCs for availability
- Each FCMC is a self-checking pair for safety, plus a backup
- So six computers total
- Use this as a small exercise in developing a model based safety case that is analyzed by push button automation
- The paper develops the self-checking pair subsystem in detail

Self-Checking Pairs



Assumptions for Self-Checking Pairs

- Verification refuses to "sign off" until all the assumptions are present and correct
- Counterexamples guide you in understanding what is wrong or missing
- Here, we need three assumptions
  - When both members of the pair are faulty, their outputs differ
  - When the members of the pair receive different inputs, their outputs differ
  - When both members of the pair receive the same input, it is the correct input (i.e., no Byzantine/SOS faults in the distributor)

Conclusions

- Safety cases provide an alternative means of compliance that may help certify adaptive systems
- Safety cases used directly in certification
- And as basis for highly assured monitors that trigger entry to adaptive mode (when assumptions are violated)
- Weakest link may be the soundness of the argument in a safety case
  - Since that's what guarantees the assumptions
- SMT solvers support pushbutton analysis of very abstract state machine descriptions
- And discovery/verification of assumptions
- Paper gives a small example in detail

Future

- Try this on more explicitly adaptive subsystem
- Explore the difference between a safety case and a verification
- Thanks to Kelly Hayhurst of NASA LaRC who monitors and advises this project
- And to support of NASA IRAC program