DHS-DOD Software Assurance Forum, McLean VA 6 Oct 2008

Very loosely based on Daniel's 2007 briefing

Software For Dependable Systems:
Sufficient Evidence?

John Rushby

Computer Science Laboratory
SRI International
Menlo Park CA USA

What

- Report by a committee of the National Research Council of the National Academies
- More precisely, the Committee on Certifiably Dependable Software Systems of the Computer Science and Telecommunications Board
  - Many briefings and meetings over a two-year study period
- Report issued just under a year ago
- Public presentation in October 2007, and continuing
  - Such as this one
- Paperback available from the National Academies Press

Why

- Sponsored by several government agencies
  - FAA, NSA, NSF, ONR

  With encouragement from others
- Due to concern about the pervasiveness of software and its increasing presence in mission-critical roles
- And the risks of undependability in software
- And uncertainty about the value of certification
- Not to mention the high cost

Who

Committee

Daniel Jackson, Massachusetts Institute of Technology, Chair
Joshua Bloch, Google Inc.
Michael Dewalt, Certification Systems, Inc.
Reed Gardner, University of Utah School of Medicine
Peter Lee, Carnegie Mellon University
Steven Lipner, Microsoft Trustworthy Computing Group
Charles Perrow, Yale University
Jon Pincus, Microsoft Research
John Rushby, SRI International
Lui Sha, University of Illinois at Urbana-Champaign
Martyn Thomas, Martyn Thomas Associates
Scott Wallsten, American Enterprise Institute/Brookings Joint Center
David Woods, Ohio State University

Staff

Lynette I Millett, Study Director
David Padgham, Associate Program Officer
Joe Eisenberg, Director, CSTB

Summary

Can software be made dependable in a cost-effective manner?

- Assessment of the state we're in
- Suggested Approach
- Broader Issues
- Findings and recommendations

Assessment

Things we know

- Software has directly led to some deaths and injuries
- And to legions of lesser failures, infelicities, and dysfunction
- Bugs in code account for 3% of software failures
- Most failures are caused by unanticipated interactions among subsystems and with the environment
- Due to poorly understood requirements
- Quality achieved is highly variable
- Certification regimes and standards have mixed record

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

Assessment

Things we don't know

- Extent to which good safety record in some areas is due to implicit factors more than certification
  - Conservatism, safety culture, experience

  Which are undergoing rapid change
  - Outsourcing, COTS, complacency, innovation
- True extent and frequency of software failures
- True efficacy of various development approaches
- True benefits of different certification approaches

Assessment

Consequences

- Mandating a particular process won't guarantee dependability
- Cannot be too prescriptive on tools and techniques
- Favor an approach based on explicit evidence
- That supports an argument for satisfaction of stated claims
- Advocate collection and dissemination of data so that we learn what works

Approach

Three Es

- Explicitness
  - About claims made, properties established
  - About assumptions on environment and usage
  - About the level of dependability
- Evidence
  - Supporting an assurance case that the claims hold
  - Open to independent audit
  - Transparency in collection and publication of data
- Expertise
  - Systems approach needed
  - But also CS knowledge and skill
  - Desired evidence is a stretch even for best practice

Standards and Goal-Based Assurance Cases

- 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
  - E.g., UK air traffic management, UK defence, US FDA, next Common Criteria (maybe)

  These make the elements explicit
- We favor them because they are founded on reason

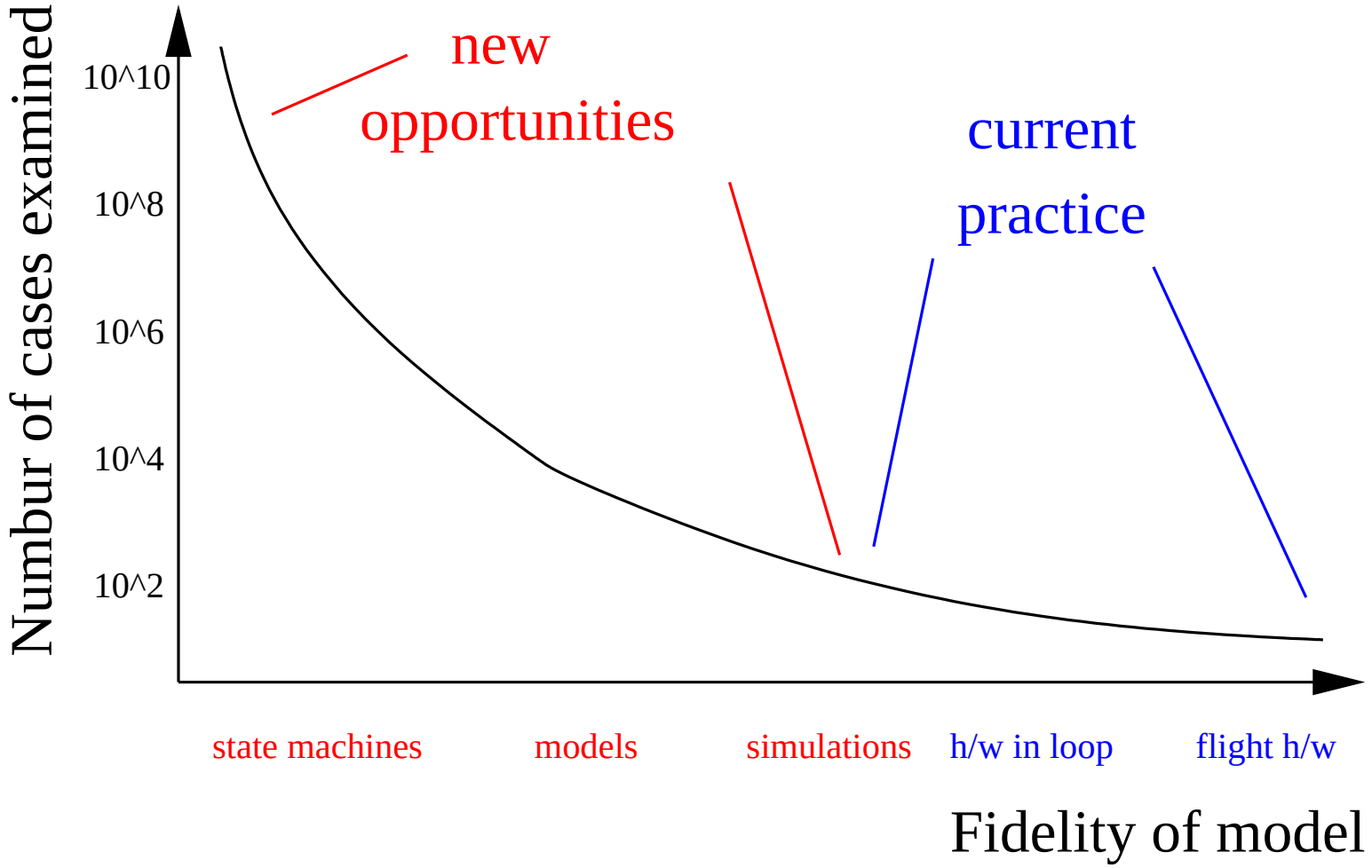
Process and Testing

- Huge reliance on these currently
- A good process is necessary
  - e.g., to preserve the chain of evidence
- But not sufficient
  - We want evidence about the product
- Testing is necessary
  - but comes too late
- And is not sufficient
  - Examines only a tiny fraction of possible scenarios
- Look toward analysis
  - e.g., static analysis, model checking, automated formal verification and test generation

  These can examine all possible scenarios
  - Albeit often under simplifying assumptions

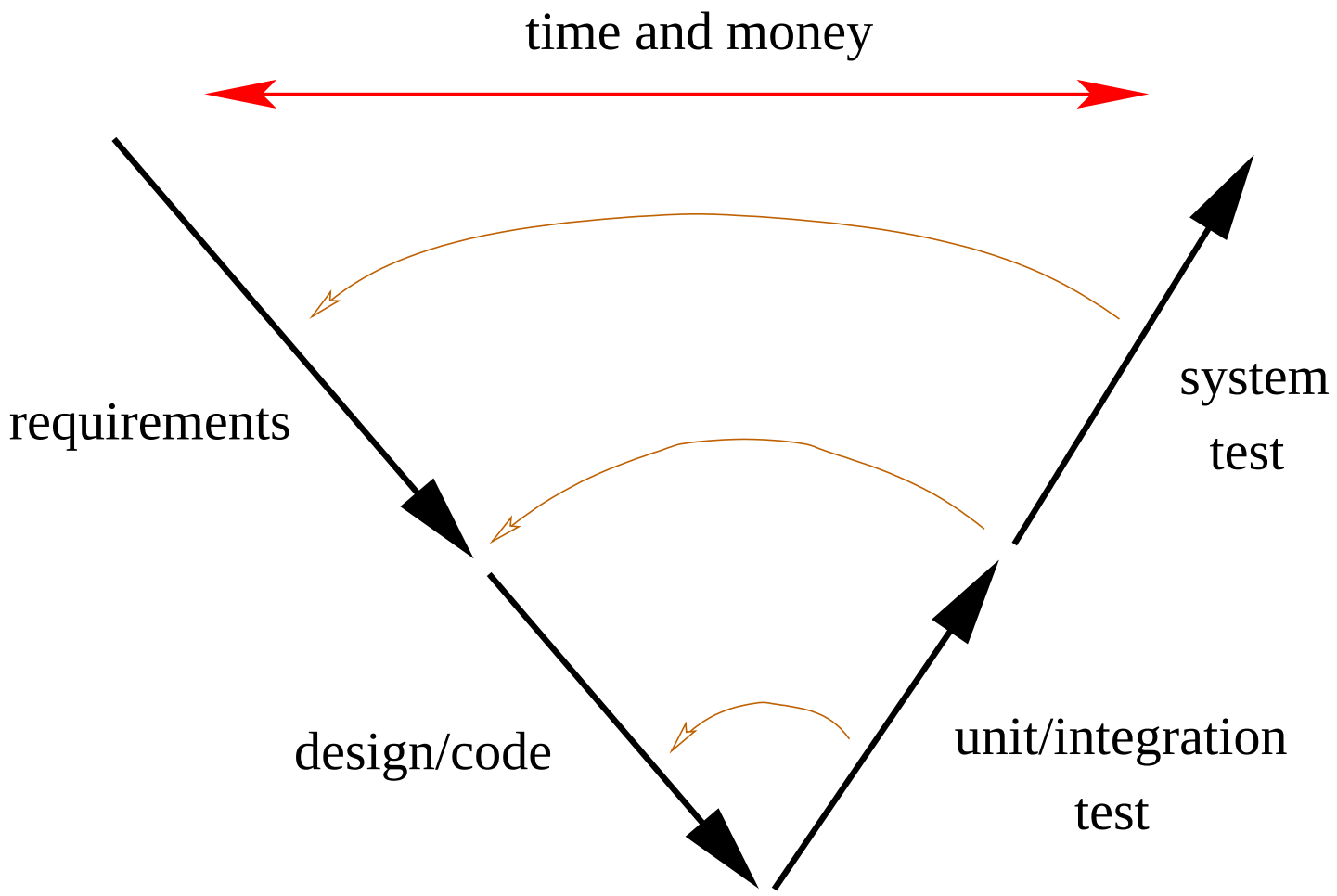
Even Weak Models Have Value

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

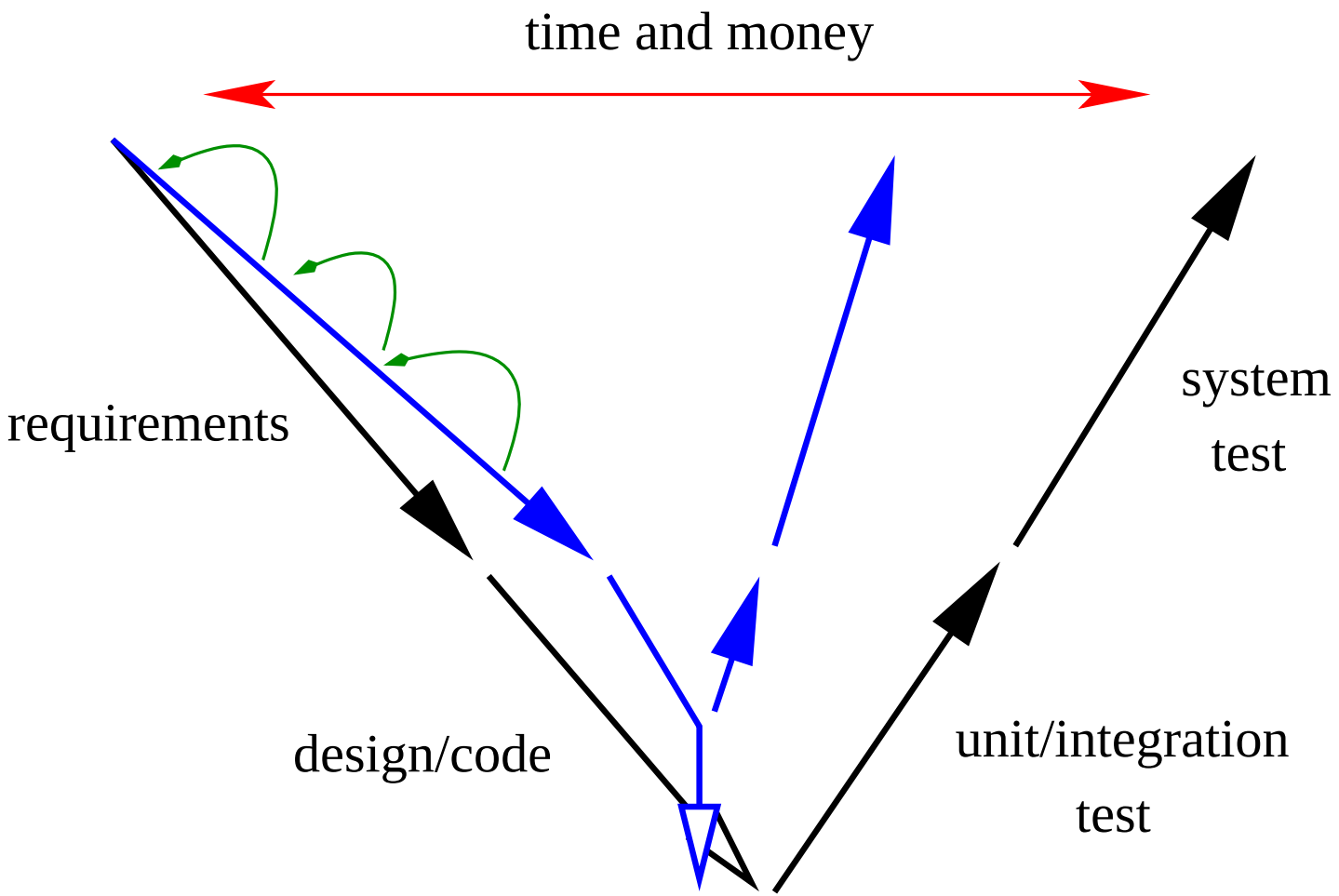


Overall V&V Process

Traditional Vee Diagram (Much Simplified)



Vee Diagram Tightened with Formal Analysis



Example: Rockwell-Collins

Getting Started and Making the Change

- A culture change is needed

First steps

- Make some claims
- Provide some evidence and an argument
- Let the market show interest and reward

Next steps

- Powerful customers demand a case
- And transparency about failures, processes, evidence

Making the change (from a standards-based regime)

- How about evidence-based standards?

Broader Issues

Education

- Software construction as systems building
- High school: less mechanism, more problem solving
- University: more on requirements, analysis, argument

Research

- Tools and techniques for assurance cases
- Compositional assurance for system-level properties
  - The assurance argument may not decompose on architectural lines
  - So what is architecture?
  - Systems are often tightly and accidentally coupled
  - So what is coupling?

Summary

Assessment

- Need improvements to keep pace with demand for dependable software

Recommended Approach

- Dependability case based on explicit claims, evidence
- Process and testing: necessary but not sufficient
- Certification = analysis of dependability case
- demand accountability

Policy Issues

- Transparency essential for a dependable software market
- Failure data should be collected, published and analyzed
- Education and research should be focused on dependability

Please read the full report—and help start a movement!