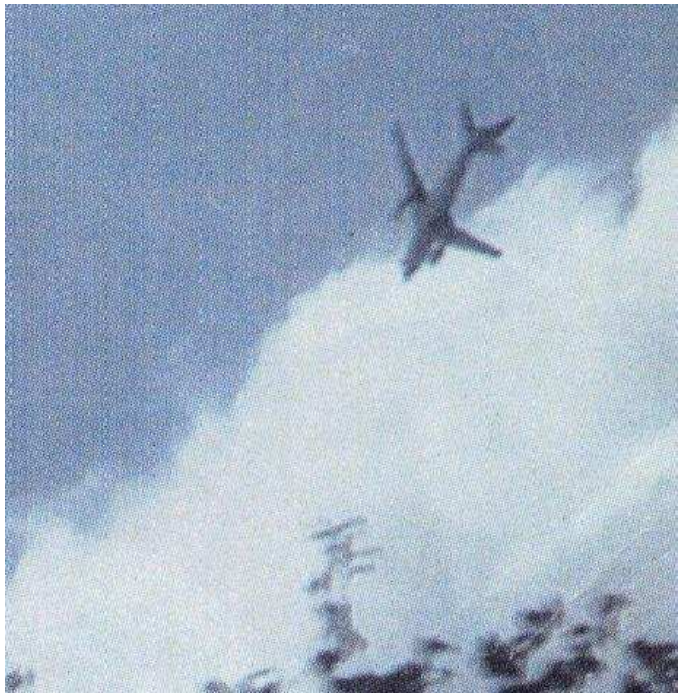
Calculating the Behavior of Software

John Rushby

Computer Science Laboratory
SRI International
Menlo Park, California, USA

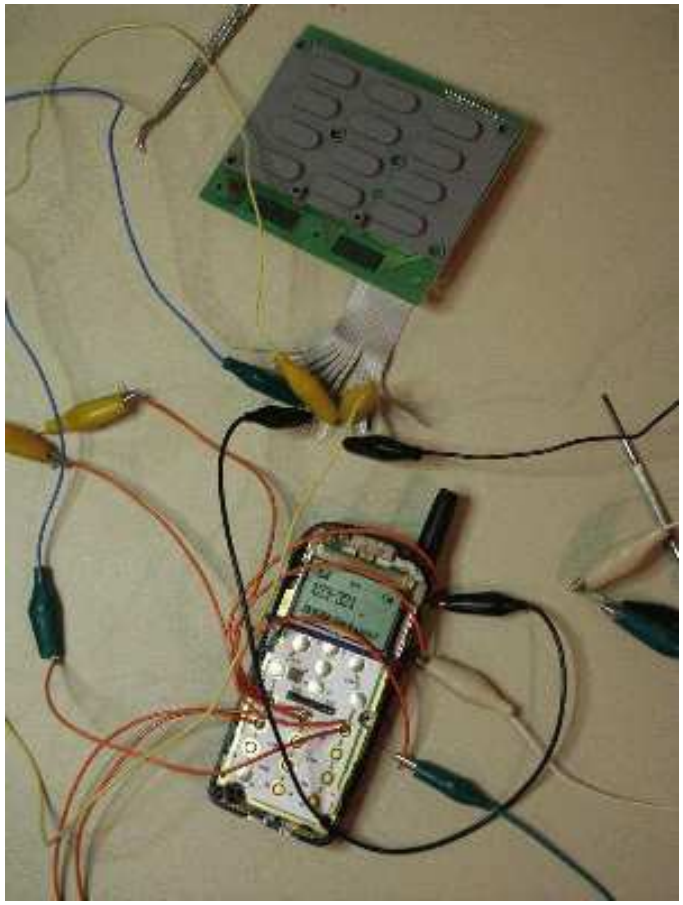
We All Know. . .

- That software is unreliable and can fail
- And even its "correct" behavior can surprise us
- But why is this, and what can we do about it?



Why is Software Unreliable?

- Because it's complicated
  - And it allows us to do complicated things
- And because it's not developed the same way as other engineered artifacts
  - It's more like a craft
- But also because it's different to physical systems
  - Aha!



Engineering vs. Craft

Engineering: Applying scientific knowledge to practical problems

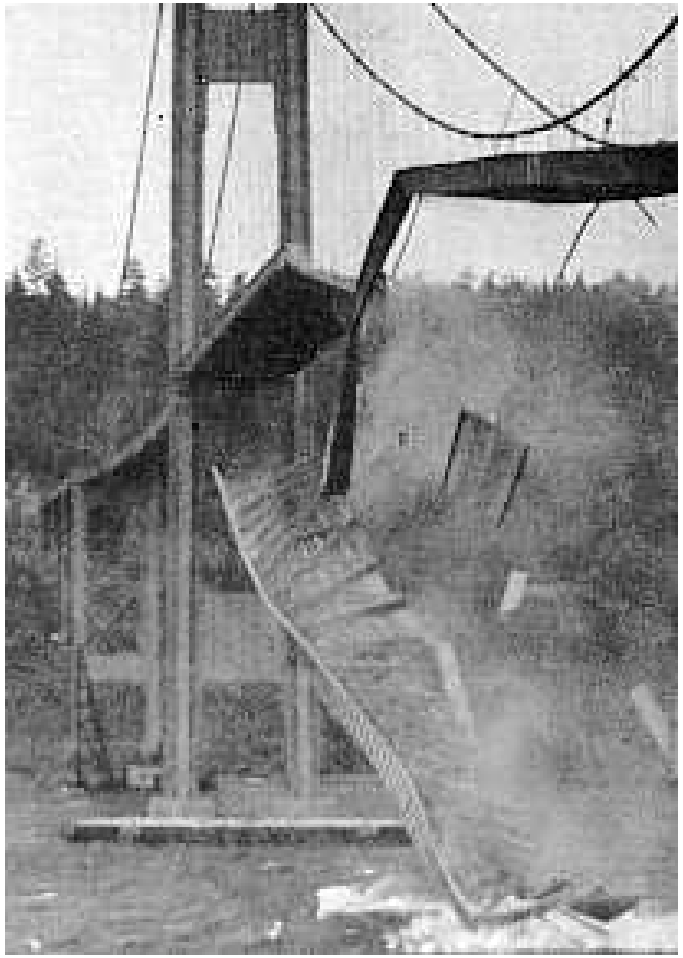
- Systematic methods of design and analysis leading to artifacts with predictable properties
- Heavy use of mathematical modeling and calculation

Craft: Skilled practice of a trade

- Relies mainly on experience
- Heavy use of "build it and test"

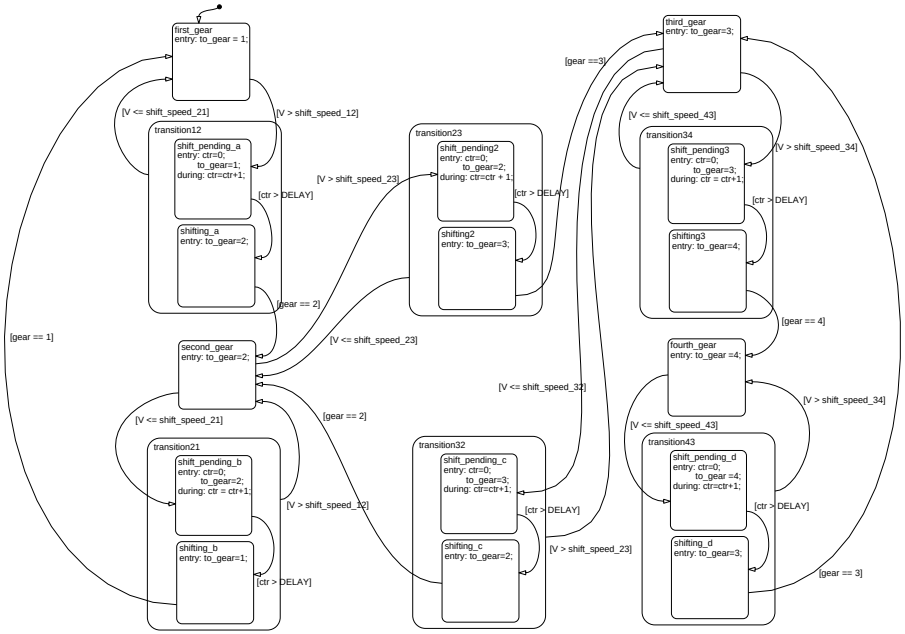
Build It And Test

- Marching troops over a bridge, for example
- Only examines the dimensions actually tested
- But plausible in the dimensions tested
- Because physical systems are continuous



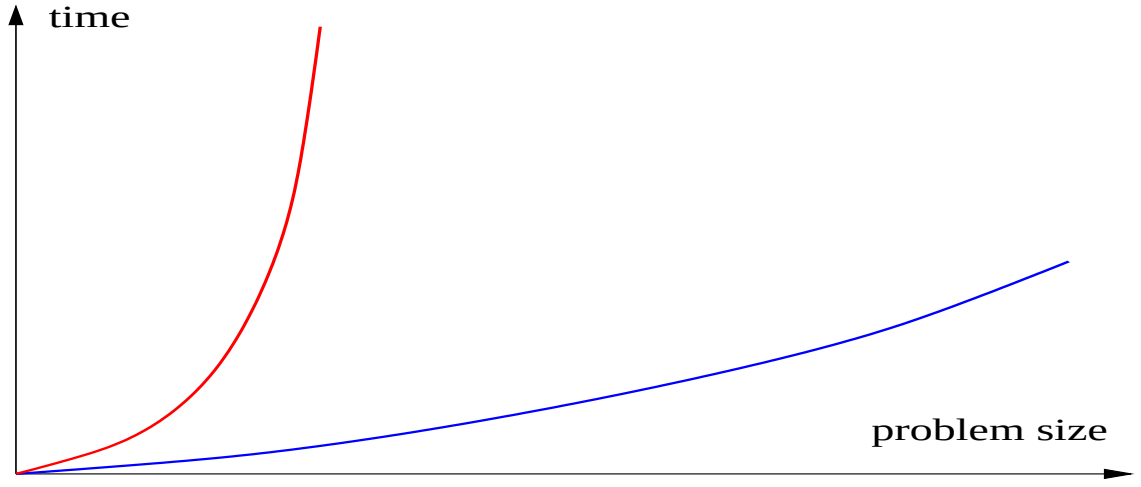
Software Is Not Continuous

- It's constructed out of discrete choices
  - `if...then...else...`
  - Billions of them
- So testing only examines a tiny fraction of total behaviors
- And we cannot interpolate from tested to untested behaviors



Mathematical Modeling and Calculation

- So software should adopt some of the methods of engineering
  - Build mathematical models of the design
  - And calculate their properties
- But the calculations are much harder than for physical systems for just the same reason that testing is harder
  - No continuity: have to consider all the discrete cases



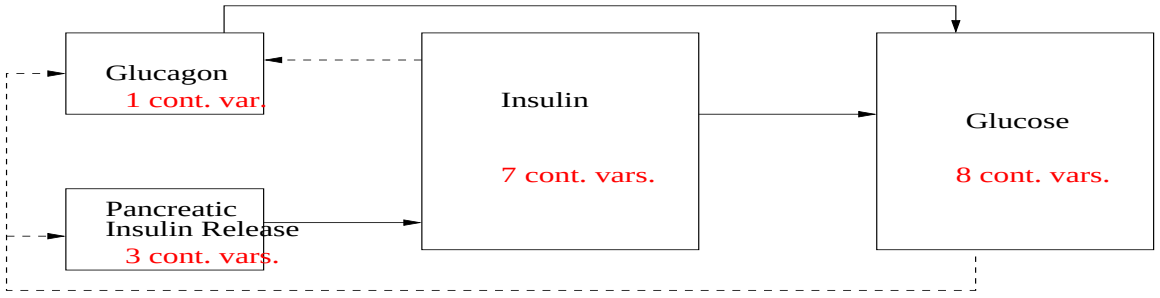
  - Hampered by computational infeasibility

But Now We Can Do It!

- 30 years of sustained research at SRI on automated deduction (AIC as well as CSL)
- Better ways of using human insight to guide the process
- Smarter modeling: less is more
  - First calculate an approximation to the system, then analyze that
- Pragmatic focus
  - e.g., improving test generation rather than proving abstract correctness
- And we're starting to get industrial takeup

Wild Claim: Calculation in the 21st Century

- The industrialization of the 19th and 20th century was based on continuous mathematics
  - And its automation
- That of the 21st century will be based on symbolic mathematics
  - Whose automation is now feasible
- Allows analysis of systems too complex and numerically too indeterminate for classical methods
- For example: symbolic systems biology



Why SRI?

- When I joined SRI in 1983, several other corporate, university, and startup groups were working on these topics
  - ISI, ORA, GE, SDC, CLI, UT Austin
- Only UT is still a force (and they came from SRI)
- Universities can seldom sustain large system development over many years
- Startups have been premature
- Corporate research focus is volatile
- We have a unique environment
  - Not university, not commercial, not corporate
  - But capable of providing the advantages of all these
- A place to pursue a long-term strategic vision, but with support for agile tactics

What I Hope to Do

- Well, we need some agile tactics right now
- A perfect storm of opportunity
- So I plan to spend time in industrial labs to understand how we can connect our technology to their needs in a way that will really make a difference

Thanks

- To my colleagues who have the ideas and do the work
  - Leonardo de Moura, Sam Owre, Harald Rueß, Shankar, Ashish Tiwari
- To all others in CSL for a stimulating research environment
  - And especially our director, Pat Lincoln
- And to SRI!