Practical Formal Methods: Presentation for GM, 1 June 2004

Software Verification/Validation Methods and Tools

. . . or Practical Formal Methods

John Rushby

Computer Science Laboratory

SRI International

Menlo Park, CA

Need: Growing Importance and Cost of Embedded Software

- Most of the innovation in new cars is enabled by embedded software
- There is more software in individual functions
- But the big gains come from integration across functions
- Integrated, distributed systems are hard to get right
  - Especially if they have to be fault tolerant
  - Or are safety-critical
- So it is common for more than 75% of embedded softweare development costs to go into verification and verification
- There is an opportunity to reduce costs and improve quality by applying automation to verification and verification of embedded systems

Approach: Formal Methods

- The basic idea is to use symbolic calculation to provide cheaper and better methods of verification and validation for software and systems
- A single symbolic calculation can subsume many individual numeric cases
  - Just as $x^2 - y^2 = (x - y) \times (x + y)$
  - Subsumes $36 - 16 = 2 \times 10$ and $49 - 4 = 5 \times 9$ and ...
- Can be used to find rare error scenarios as well as to verify their absence
- Symbolic calculation is mechanized using the methods of automated reasoning: theorem proving, model checking, constraint solving, etc.
- There has been sustained progress in these fields for several decades and they have recently broken through the barriers to practical application
- SRI has been a leader of this technology throughout its history

A Spectrum of Formal Methods

Interactive theorem proving: requires great skill and resources

- Can solve very hard problems
- E.g., Verify that Flexray's clock synchronization withstands any single fault

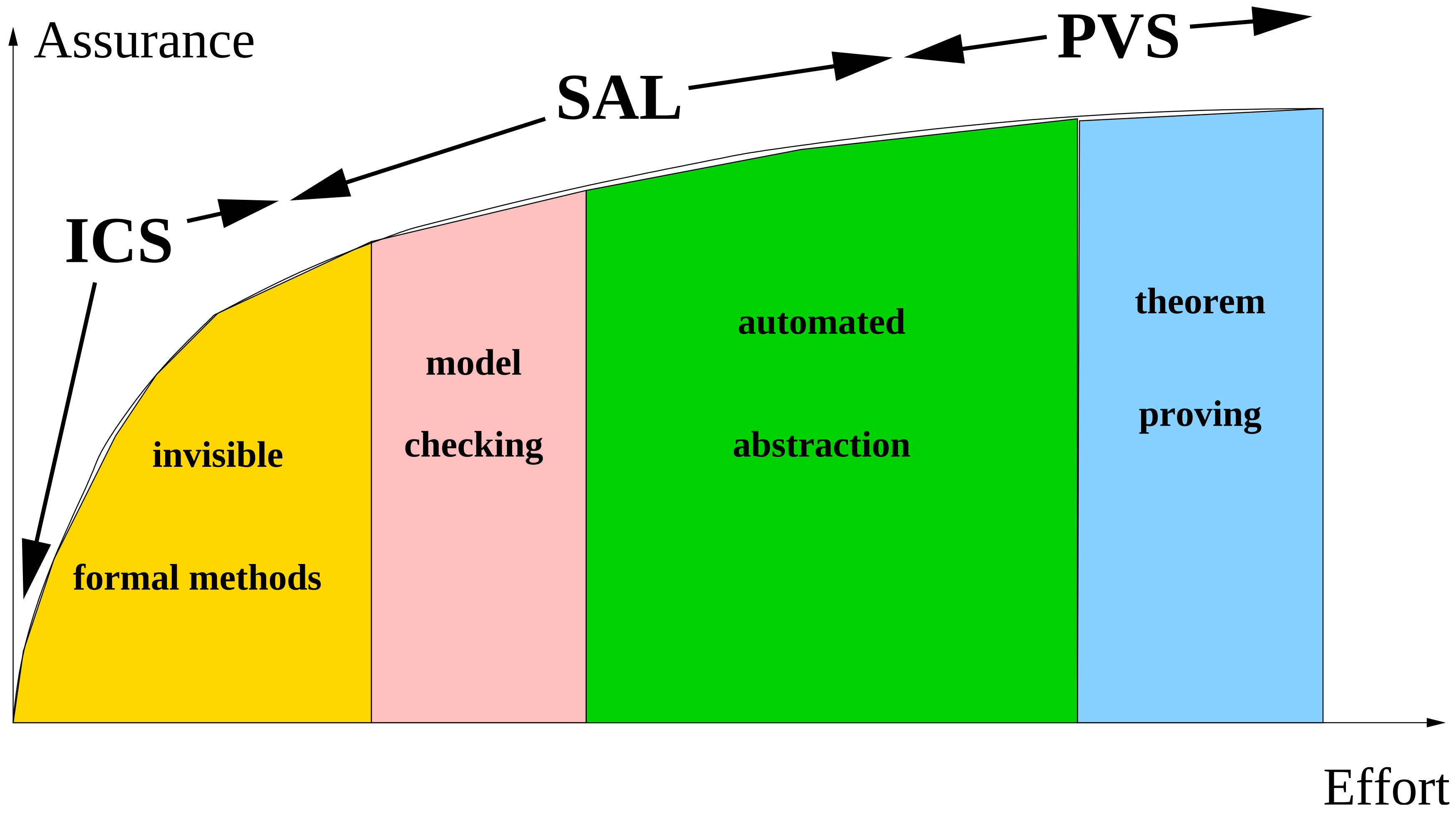
Model checking: analysis is automatic but must specify the model and property

- Can search huge state spaces (trillions of reachable states) efficiently
- E.g., Find the worst case start up delay for Flexray
- E.g., Check that horizontally integrated functions interact as expected

Invisible formal methods: driven directly off model-based developments

- Uses symbolic calculation to automate traditional work flows
- E.g., Generate unit test cases to provide MC/DC coverage
- E.g., "Find me an input vector that gets me to here with $x > 3$"
- Check compliance with guidelines (e.g., no 12 o'clock rule in Stateflow)

Our Tools Cover the Spectrum



Our Tools

- PVS: Industrial strength theorem prover (since 1993)
  - Probably the most widely used theorem prover in research and education
  - Used for verification of AAMP5 (Rockwell)
  - And Time Triggered Architecture (TTTech, NASA, Honeywell)
  - GM group in Asia has recently applied for a license
  - Some other commercial users (e.g., Sun)
- SAL: Industrial strength suite of model checkers (since 2003)
  - Used for analysis of TTA startup
  - A current application focus is automated test generation
- ICS: Core decision procedures and SAT solver used in PVS and SAL
  - Designed to be embedded in other tools
- See `fm.csl.sri.com` for descriptions and our roadmap

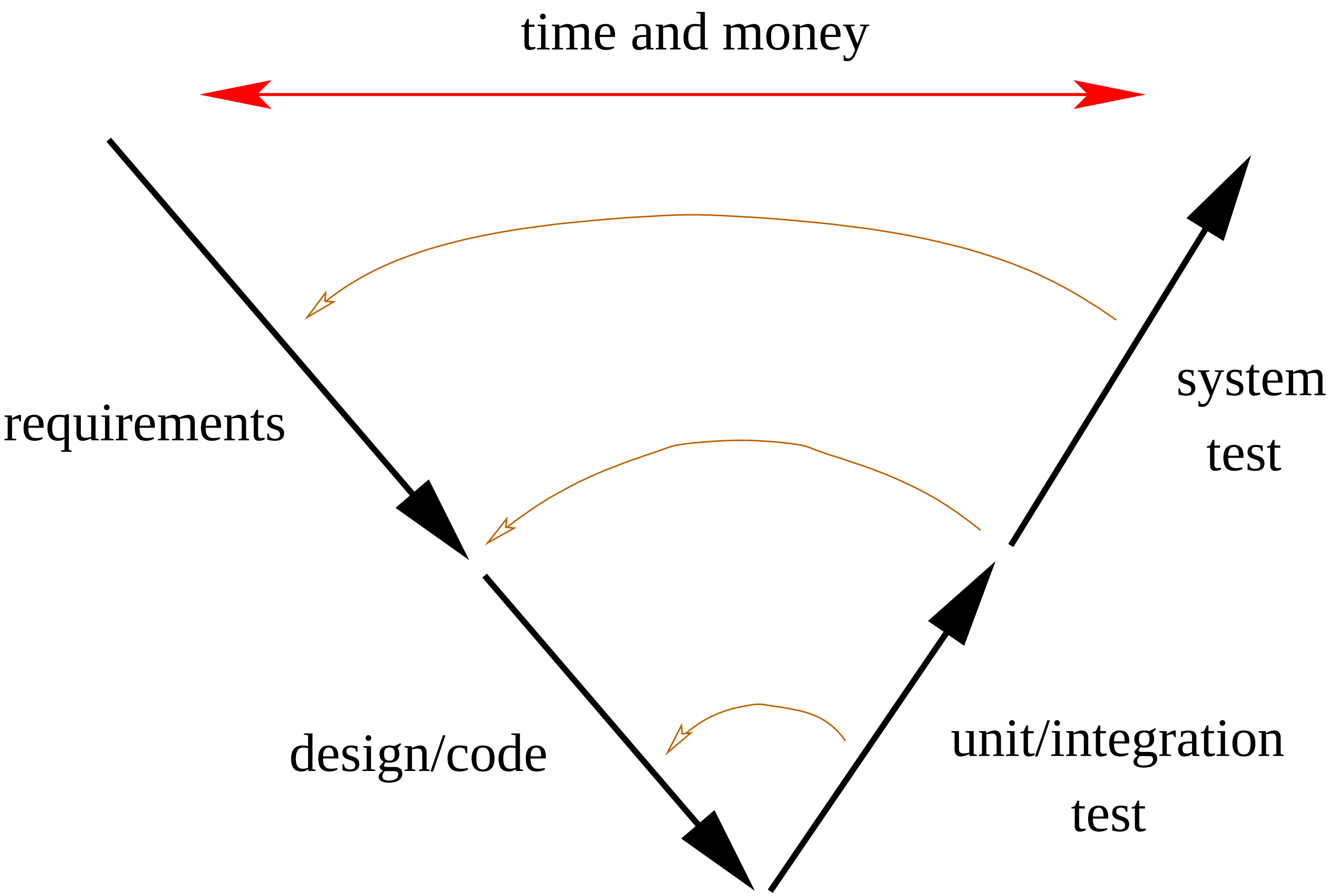
Invisible Formal Methods

- New design practices: model-based development methods provide the artifacts needed by automated analysis
  - Models serve as formal specifications
  - We have a formal semantics and translator for Stateflow
- New technology (in SAL): very fast, scalable model checkers that can handle arithmetic and other data types
- New ideas: invisible formal methods
- These combine to create new opportunities
- Example: Generate test vectors that will drive an implementation through all the states and transitions of its model

Automated Test Case Generation

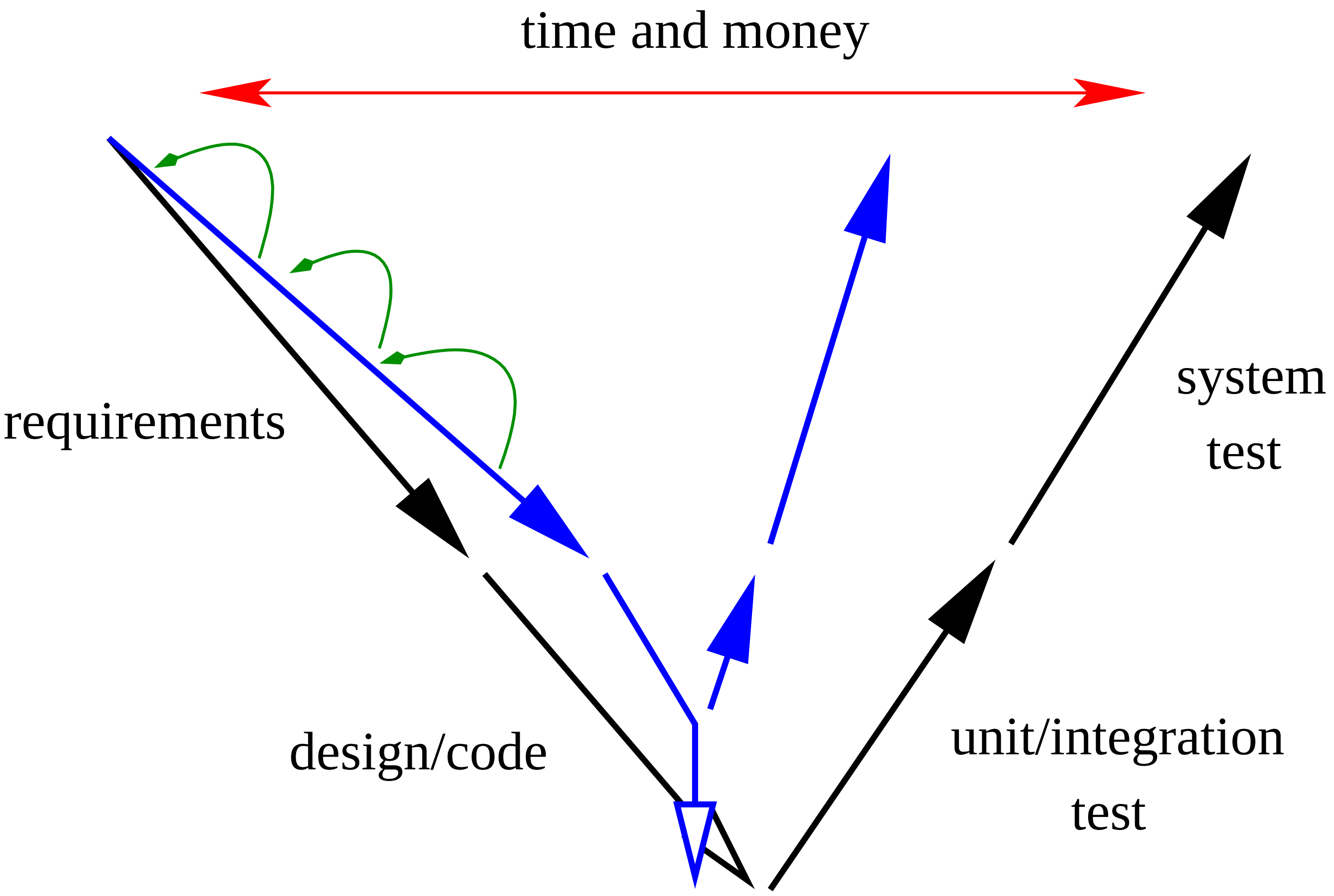
- Basic approach uses the counterexamples generated by a model checker
- Counterexample to you cannot get here is a test case that gets you there
- There are several technical issues dealing with arithmetic in specifications
  - Which we have solved (patents pending)
- Existing methods give many short tests with much redundancy
  - We have new methods that generate fewer deeper tests (patent pending)
  - E.g., State coverage for a 4-speed shift selector in one test of length 86
- We also have technology (automated analysis of hybrid systems) that could take test test generation beyond unit tests into integration and system tests

Benefits: Simplified Vee Diagram



Automated formal analysis can tighten the vee

Tightened Vee Diagram



Competition

- Test generation for Statemate is automated by Motorola in Veristate
  - Good integration, relies on user-written "test observers," weak FM technology
- For Simulink by T-VEC
  - Good integration and methods, weak FM technology
- For Stateflow by RSI in Reactis
  - Good integration and methods, weak FM technology
- We have the best FM technology, more powerful test generation methods, the ability to go beyond test generation, but less integration with commercial products

Summary

- We are the experts in practical formal methods, and can help others
  - Evaluate
  - Apply
  - Develop

  this technology
- Our PVS, SAL, ICS tools are mature (though continually enhanced) and available for licensing
- We are seeking partners to help us develop and evaluate our technology for automated unit test generation
  - And other applications for invisible formal methods