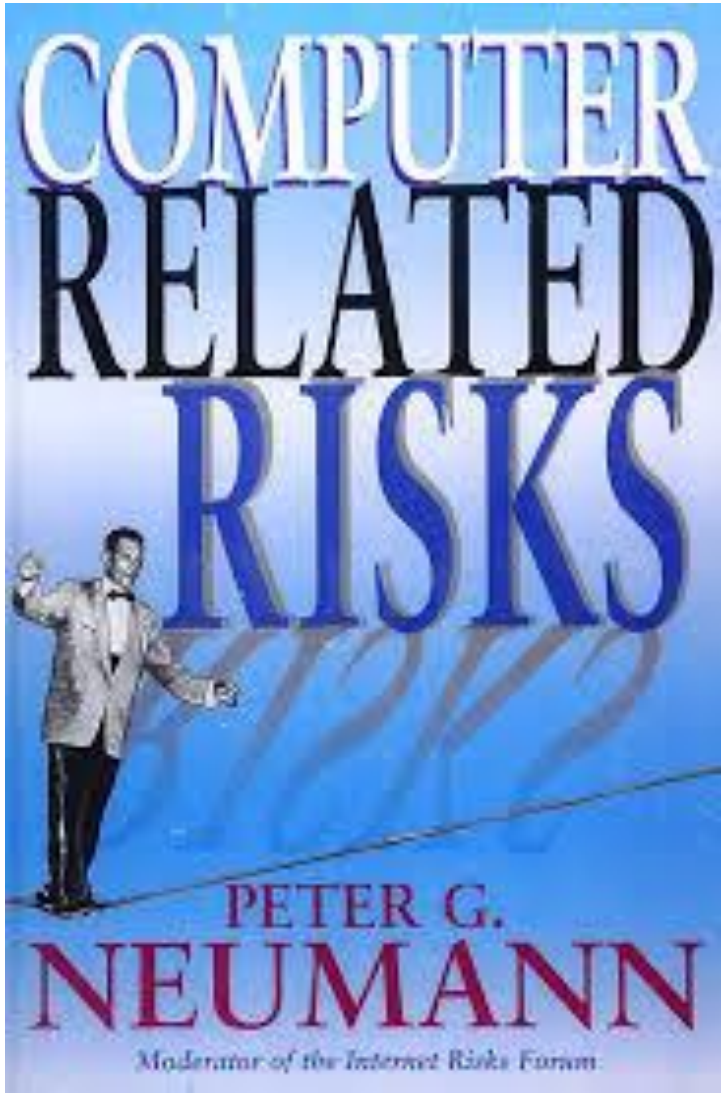
# Formal Methods Program

Menlo, May 25, 2025

# Disruption, Failures, and Adversity

| Channel | Incidences |
|---|---|
| Hardware | Intel FDIV, Spectre/Meltdown |
| Side Channel | Electromagnetic, acoustic, power, timing, optical, radiation, wear-and-tear (Row Hammer) |
| Calculation | NASA Mariner, Mars Polar Lander, Mars Climate Orbiter, Ariane-5 |
| Memory/Type | Buffer overflow, null dereference, use-after-free, bad cast |
| Crypto | SHA-1, MD5, TLS Freak/Logjam, Needham-Schroder, Kerberos |
| Input Validation | Buffer over-read (Heartbleed, Cloudbleed) |
| Race/Reset Condition | Therac-25, North American Blackout, AT&T crash of 1990, Mars Pathfinder |
| Code injection | SQL injection, cross/site scripting, malvertising, data poisoning |
| Provenance / backdoor | Athens Affair, Solar Winds |
| Social engineering | Pretexting, Honeytrap, Tailgating/Piggybacking |



# The Cost of Failure

The estimated engineering cost of fixing poor quality code exceeds $1 trillion annually in the U.S. alone

- with failure to patch known vulnerabilities being the largest contributor to these costs

Cybercrime thrives on code vulnerabilities, and is estimated to be another $8 trillion a year business and growing

- that is nearly $1 billion every hour

Sources:

www.synopsys.com/blogs/software-security/poor-software-quality-costs-us

cybersecurityventures.com/cybercrime-to-cost-the-world-8-trillion-annually-in-2023

Source: Pieter Bruegel the Elder

# Formal Methods to the Rescue

FM a viable, if not the only alternative, to traditional bug hunting

- Integration with industrial development processes (Intel, Collins, AWS, …)
- Microprocessors, separation kernels, real-time operating systems, fault-tolerant algorithms, and crypto libraries nowadays formally verified almost routinely
- Billions of small theorems machine-proved every day

Satisfiability revolution (SMT, BMC, k-induction, IC3) is making Vannevar Bush's prophecy come true

AS WE MAY THINK

A TOP U. S. SCIENTIST FORESEES A POSSIBLE FUTURE WORLD IN WHICH MAN-MADE MACHINES WILL START TO THINK

by VANNEVAR BUSH

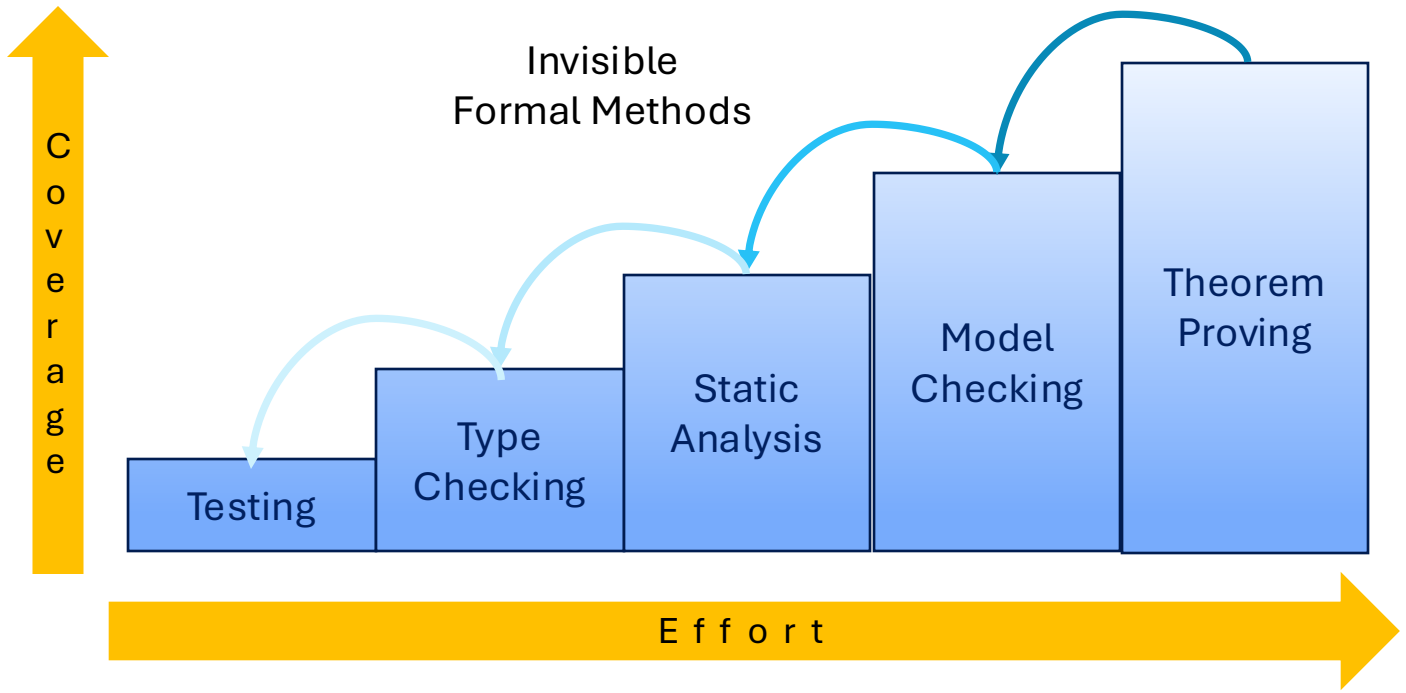
"We may someday click off arguments on a machine with the same assurance that we now enter sales on a cash register"

https://www.theatlantic.com/magazine/archive/1945/07/as-we-may-think/303881/

# SRI's Formal Methods Program

SRI's **Formal Methods Program** has the insight, experience and tools to deliver demonstrably safe, secure, and resilient software and AI systems




*"... formal methods is an umbrella term for a range of mathematically rigorous techniques for producing software and machine-checked evidence that the systems will act in ways that are intended and not in ways that are unintended."*
(DARPA Guide for Formal Methods)

# World-leading Expertise

We are internationally recognized experts who are actively shaping the field through cutting-edge research and development

- Fault-tolerant computing (Byzantine Agreement)
- Separation kernel and proof of separability
- Interactive theorem proving
- Rigorous assurance cases
- Formal verification of critical HW/SW building blocks
- Formal methods for human factors
- Automated reasoning  techniques
- Safe AI and formal methods

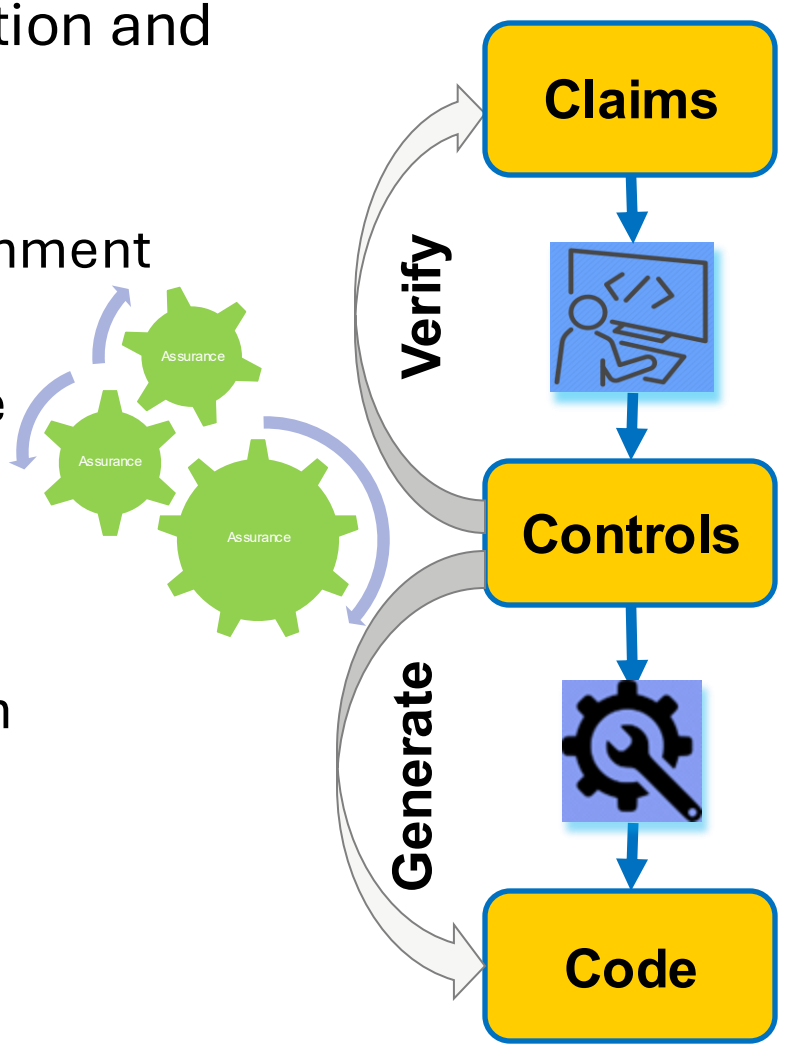

Screenshot from the film The Martian (2015)  of code from the PVS  NASA library

# Comprehensive Toolset

We provision a suite  of cutting-edge formal specification and verification tools

- **PVS:** Integrated specification and verification environment
- **Yices:** Leading SMT solver
- **Sally:** IC3-based model checker with MOXI language
- **ETB:** Workflow-based curation of evidence
- **PCE:** Automating Markov logic networks
- **RADL:** Multi-rate CPS architecture
- **PVS2C:** Generating memory-safe efficient code from specifications




Downloads at https://github.com/SRI-CSL/,

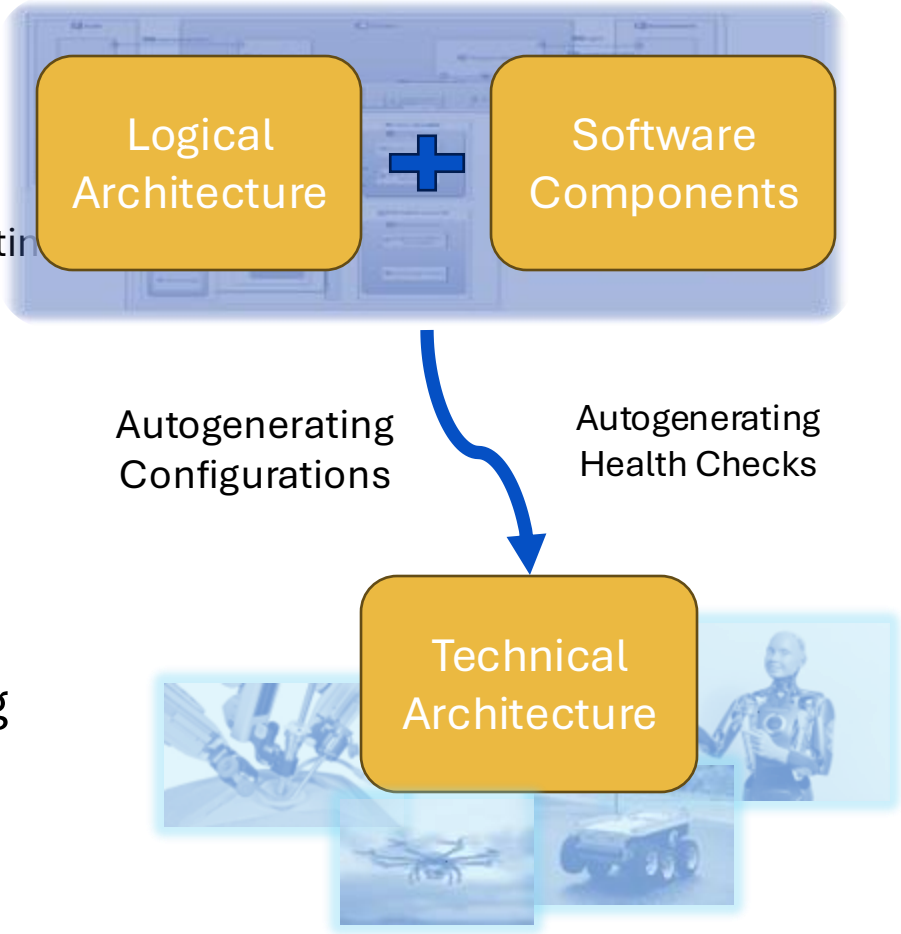
Generally open-sourced (under  GPL 3.0)

# Decades of Experience

We have a history of successful cooperations on the sustainable practice of formal methods in the aerospace, defense, healthcare, and finance industries

- Development of the first flight-control computer
- Formal verification of HW/SW embedded systems
  - microprocessors, hypervisors, arithmetic circuits, real-time OSs, ti
    triggered architecture, FADEC
- Verification of distributed, fault-tolerant, real-time systems
- Automated curation of evidence and assurance cases
- Formal analysis of requirements/specification documents
- Autoformalizing and analyzing data and protocol standards
- Rigorous methods and tools for constructing and assessing assurance cases



# Formal Methods Team

John Rushby

Natarajan Shankar

Sam Owre

Harald Ruess

Hassen Saidi

Huascar Sanchez

Stephane Graham-Lengrand

Ahmed Irfan

Karthik Nukala

# Thank you!

fm@csl.sri.com

# A Short History of Formal Methods

## 1970s:

**SIFT:** State-machine replication, modern fault tolerance

**Byzantine Agreement:** Tolerating faults with no assumptions on behavior; later, basis of blockchain

**PSOS:** Capability-based security, father of CHERI, ARM Morello

**Early FM:** JOVIAL Verifier, Boyer-Moore, SMT

**Information Flow Analyzer:** Pre-noninterference semantics

**HDM:** Hierarchical development of secure software


## 1980s:

**Separation Kernel:** Later evolved to MILS, also partitioning/safety

**EHDM:** Clock synchronization proofs

**Noninterference** and its intransitive form

**Algebraic Semantics** and rewriting

**OBJ3:** Modular and equational programming

**IDES:** Intrusion detection, evolved to network intrusion detection

**Institutions:** Abstract model theory for specification and programming

## 1990s:

**Interrogator**: Cryptographic protocol verification

**State of the Art FM:** PVS, RRM, Maude

**PVS:** Interactive specification and verification

**Model-check the brain:** human factors verification

**Lazy compositional verification**

**RRM: p**erformant rewrite rule machine

**Maude:** model-checking concurrent systems

**Reflective Logic** for meta-programming

**Reconfiguration** from first principles

**AAMP5:** Microprocessor verification

**Bitvector** decision procedures

**Predicate abstraction**

**Floating-point verification**

## 2000s:

**ICS:** First advanced SMT solver

**Cyberlogic:** Logic of evidential transactions

**SAL:** Combining finite and infinite-state model checking

**CAPSL:** Crypto-protocol analysis

**Pathway logic:** Analyzing biological pathways

**Provenance:** Theory and tools

**State of the Art FM**: PVS, Maude continue

**Calendar automata:** Verification of fault-tolerant distributed real-time algorithms

**WMC:** Witness-producing model checker

**Parikh automata** that count

**Relational abstraction:** generalizes qualitative physics

**PCE**: analyzing Markov logic networks

## 2010s:

**DimSim/SimCheck**: Simulink analyzer

**OCCAM:** Debloater

**CHERI** architecture

**Yices2** SMT solver

**State of the Art FM:** PVS, Maude continue

**ARSENAL:** Semantic parsing

**RADL:** Resilient multirate architectures

**Parsley:** Verified parsing/unparsing

**Sherlock:** Neural net analyzer

**Kernel of Truth:** Verification of proof checkers against trusted kernel

**PVS2C: :** Autogenerating efficient code from specifications

**Reverse engineering** of hardware

**ETB**: Evidential toolbus, assurance workflow

**OGIS:** Oracle-guided inductive synthesis

**SeaHorn**: static analyzer

## 2020s:

**Assurance 2.0:** Rigorous assurance cases

**State of the Art FM**: PVS, Maude continue

**Ontic types:** Meaningful typing

**CoProver:** LLM-assisted interactive theorem proving

**Stitches:** Seamless system of systems integration

**Zero-Knowledge:** Verification of proof systems and code

**DPRIVE:** Hardware design for fully homomorphic encryption

**CDSAT:** Conflict-Driven Reasoning

**Semantic Browser:** Validated informal specifications

...