galois

Reflections on Industrial Use of Frama-C

Joseph R. Kiniry and Daniel M. Zimmerman

Sound Static Analysis for Security Workshop

NIST — Gaithersburg, Maryland — 28 June 2018

About Galois

- established in 1999 to apply functional programming and formal methods to the problem of information assurance
- over 40 active projects for numerous clients, both U.S. government (NSA, DARPA, IARPA, Homeland Security, Air Force Research Lab) and commercial (Amazon, LG, others)
- over the years, we have substantially broadened our scope to *high assurance everything*

Galois and Frama-C

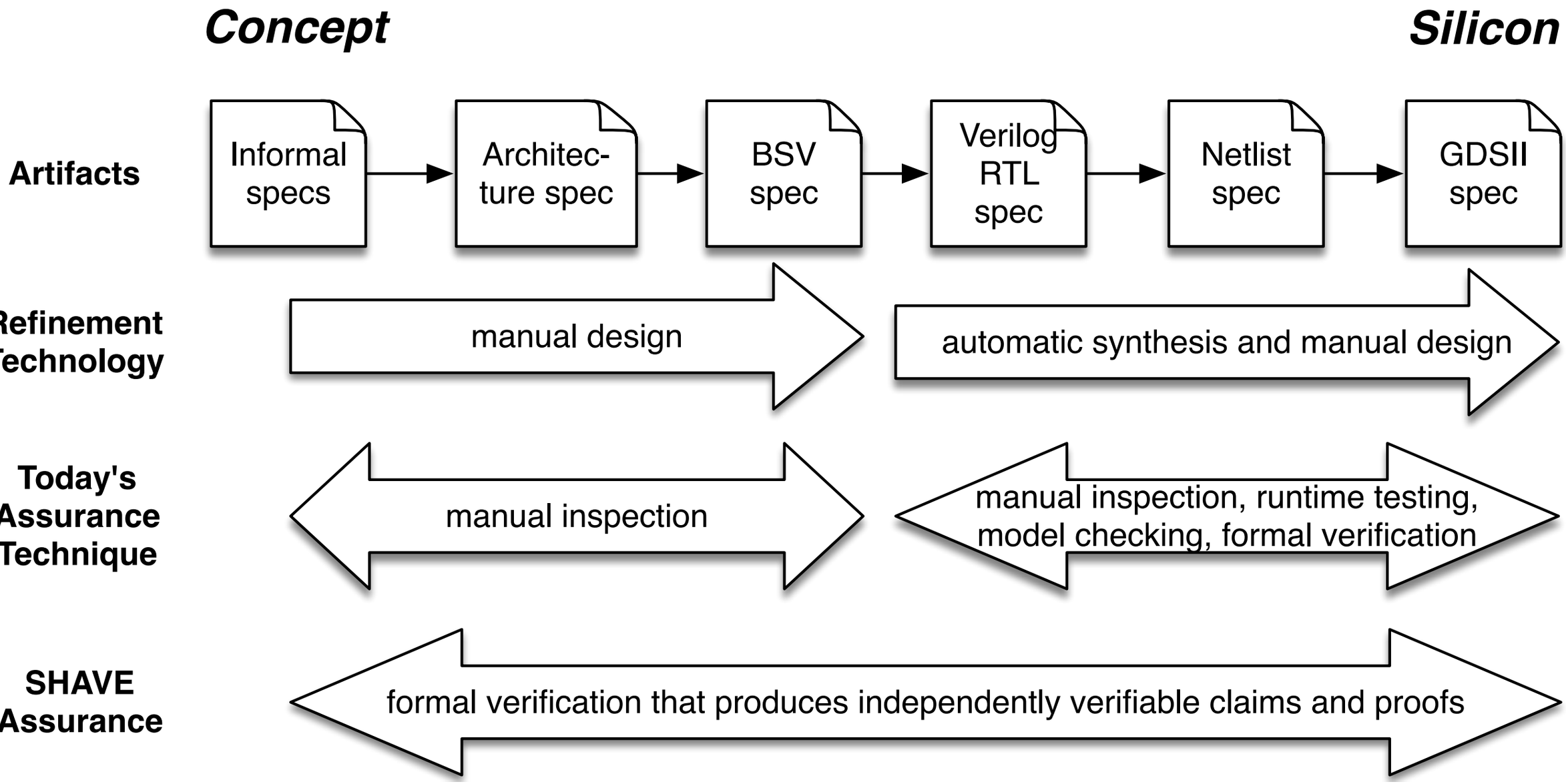
- Galois has used Frama-C on some projects
- DARPA Crowdsourced Formal Verification
  - WP plugin to generate verification conditions
  - custom plugins to generate schematic assertions and program traces
- DARPA SHAVE

- Our clients rarely ask for formal verification in Frama-C's "sweet spot"

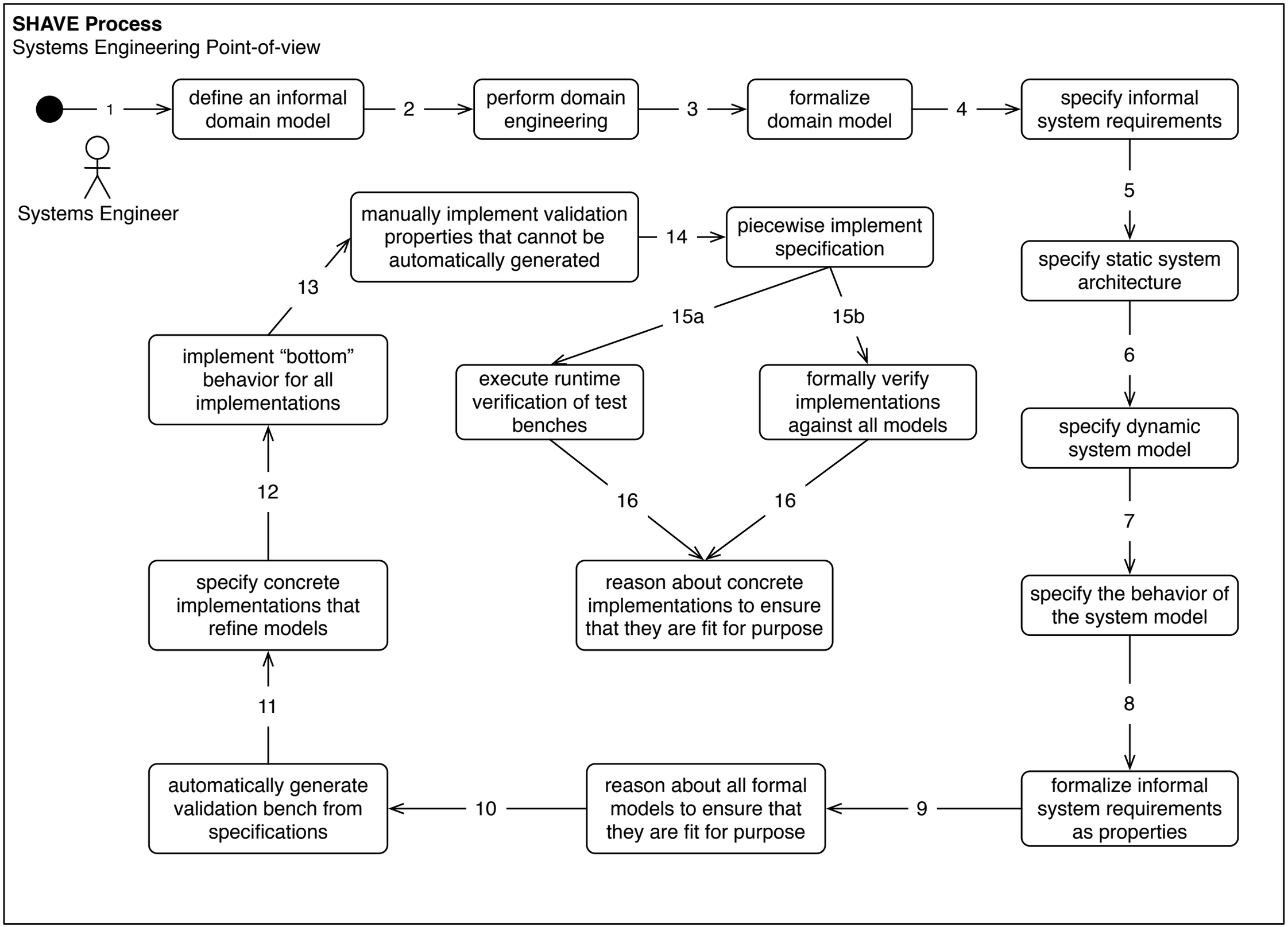
SHAVE: Software/Hardware Assurance Verified End-to-End

- DARPA MTO seedling for the SSITH program
- a bump-in-wire encryption device
- single, one-time AES key provisioning
- encryption or decryption mutual exclusion
- open hardware, firmware, and software
- crypto realized as a MMIO RISC-V extension
- custom development of a verification system for Bluespec hardware description language

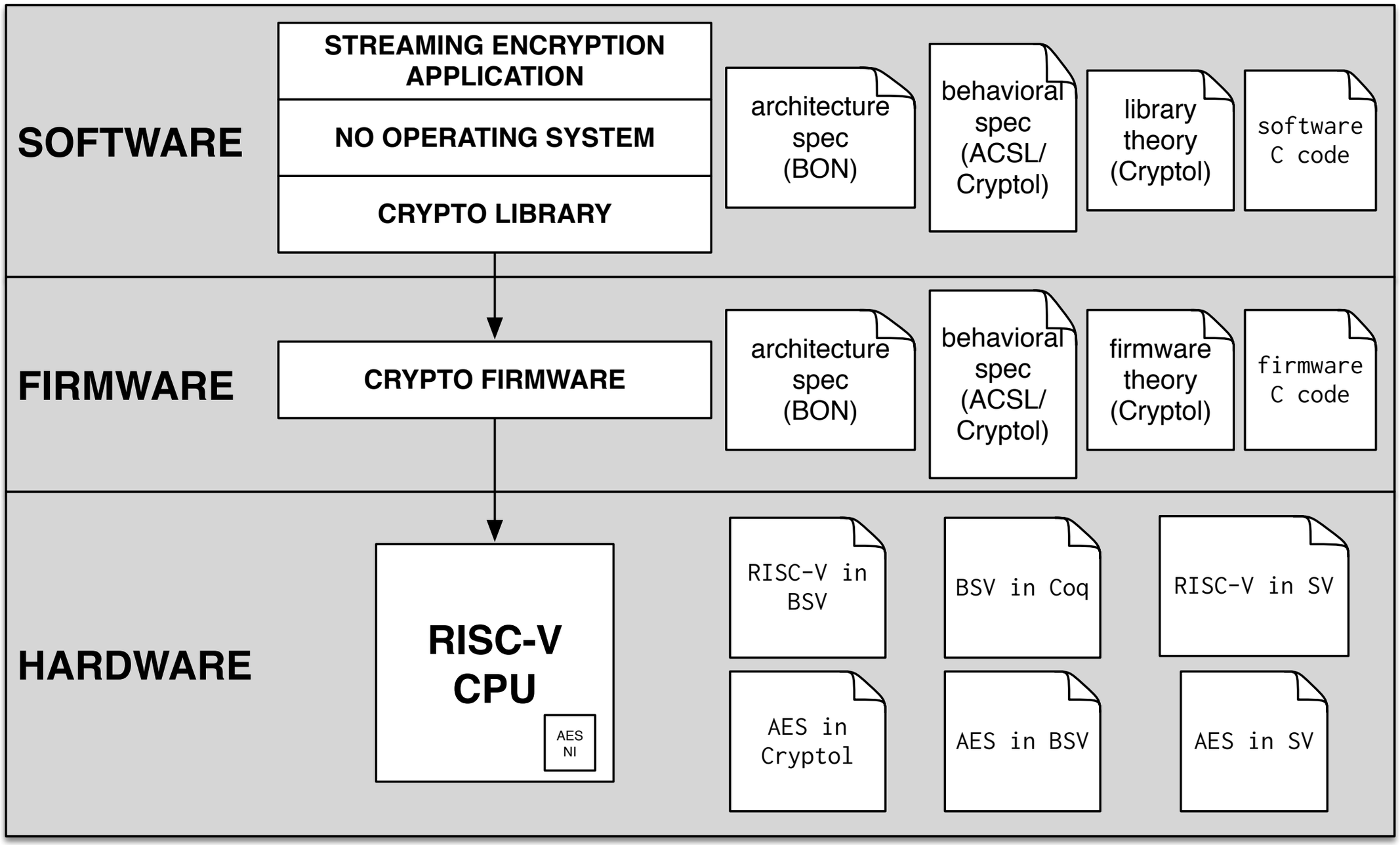
SHAVE Assurance



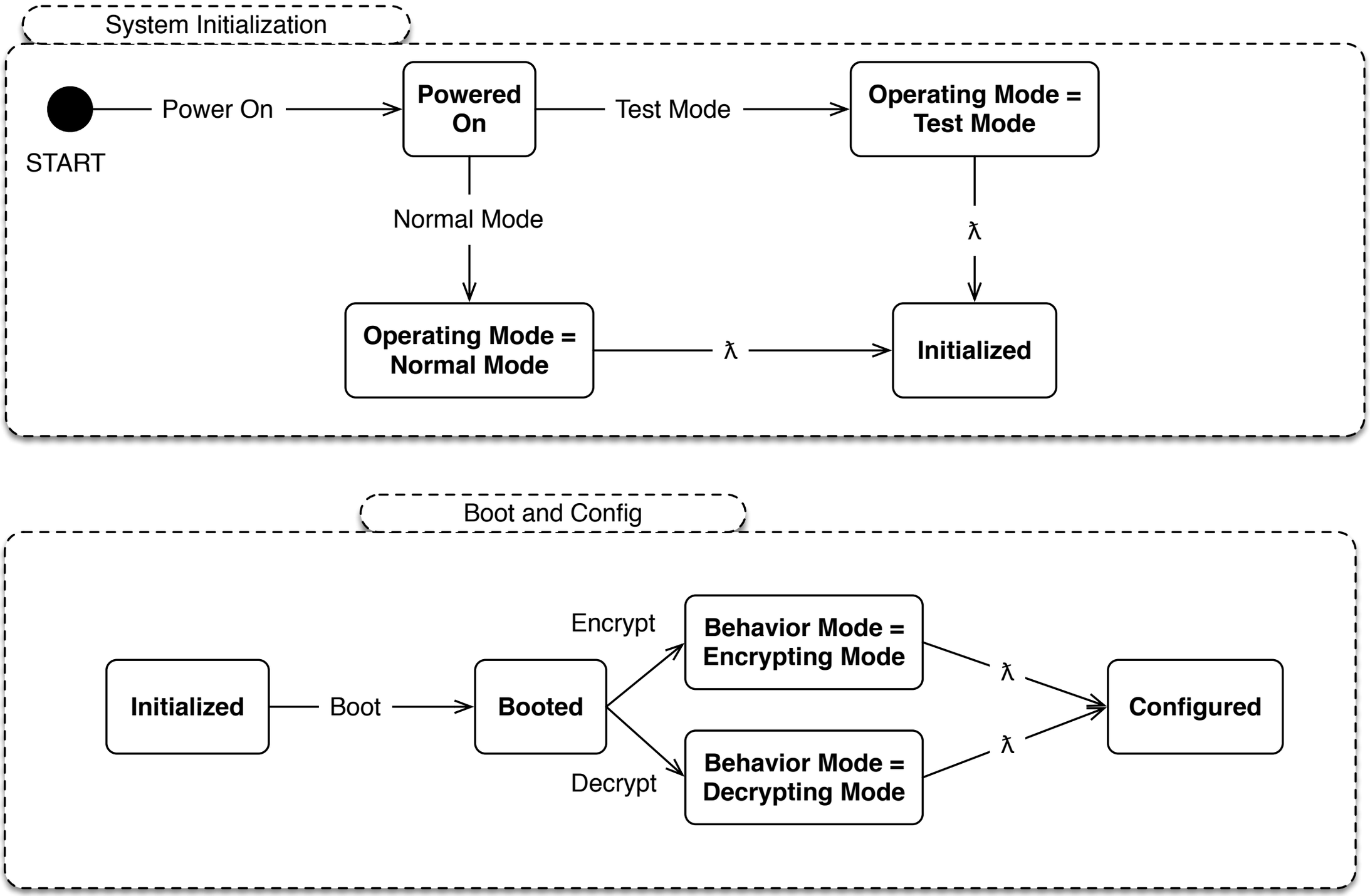
The SHAVE Process



The SHAVE Architecture



SHAVE Abstract State Machines



SHAVE Assurance Case

- assurance case via layered rely/guarantee
- entire system formally specified and assured except for (a) implementation of RISC-V ISA, and (b) behavior below RTL
- entire assurance case hangs on realization of system ASM composing ASMs for soft/firmware
- security properties include reset predicate, write-once key, no key leakage, crypto correctness, and guarantee that all bits are always encrypt/decrypted
- also includes formally verified trusted boot for RISC-V

Assurance Technologies and Compositionality

- userland software and firmware specified in BON, PVS, ACSL, Cryptol, & SAW and verified using multiple Frama-C plugins, PVS, Cryptol, and SAW
- hardware and state machine assurance via Cryptol and SAW
  - including a new frontend on SAW for reasoning about Bluespec SystemVerilog
- Cryptol is the compositional formal model that spans formalisms and tools

Composed Assurance Case

- ad hoc
- human reviewed to ensure specs written in different concrete languages are consistent
- complex!

- need a SAW-like *assurance language* that understands evidence
- we're working on that for SSITH for hardware (and firmware) security, and some day…

Comparison of Experience with other BISL Technologies

- we have ~20 years of experience using CodeContracts, SPARK, Eiffel, and JML
- we have written several formal verification and rigorous validation tools on these foundations
- our statements of joy and disappointment with respect to ACSL and Frama-C come from this background, with love

Frama-C Tools and Techniques Used

- both mainstream & experimental plugins used
- metrics, callback, pdg, and from analysis to drive verification process
- ASM reasoning with Aorai
- rtegen for combined reasoning a la Julien's talk this morning on combining RTE+E-ACSL
- value analysis for unexpected behavior
- wp reasoning about functional correctness

Our Experiences

- tool documentation is very good
- tool behavior is not as reliable as compilers
- the “fine print” is hard to find and understand even for a formal methods expert
- understanding the dependencies between, and order in which, different plugins should/can be used is complex
- experimental aspects of ACSL and reasoning tools are what we need most for scaling (advanced logic specifications, sets and lists, model programs, memory model subtleties, etc.)

Constructive Next Steps

- we continue to use Frama-C at Galois as one of the tools in our toolbox
- Frama-C complements our reasoning capabilities (embodied in Cryptol and SAW)
- we see opportunities for writing new (possibly open source) plugins that relate to our work on hardware security and firmware reasoning