Formal Verification for (Model-Based Diagnosis in) a Next-Generation Space Shuttle

Stacy Nelson (NelsonConsult / NASA Ames / NGC)

Charles Pecheur (RIACS / NASA Ames)

Overview

What does it take to put advanced software and formal methods into a typical space project?

- Overview of 2nd Gen RLV IVHM
- Current V&V Practice and Standards
- Formal Methods for IVHM V&V
- Ames V&V Tools for Livingstone
- Tool Maturation and Integration
- Conclusions and Perspectives

WARNING – EJECT!

√ AVIONICS

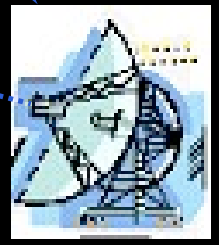
X AIRFRAME

X PROPULSION

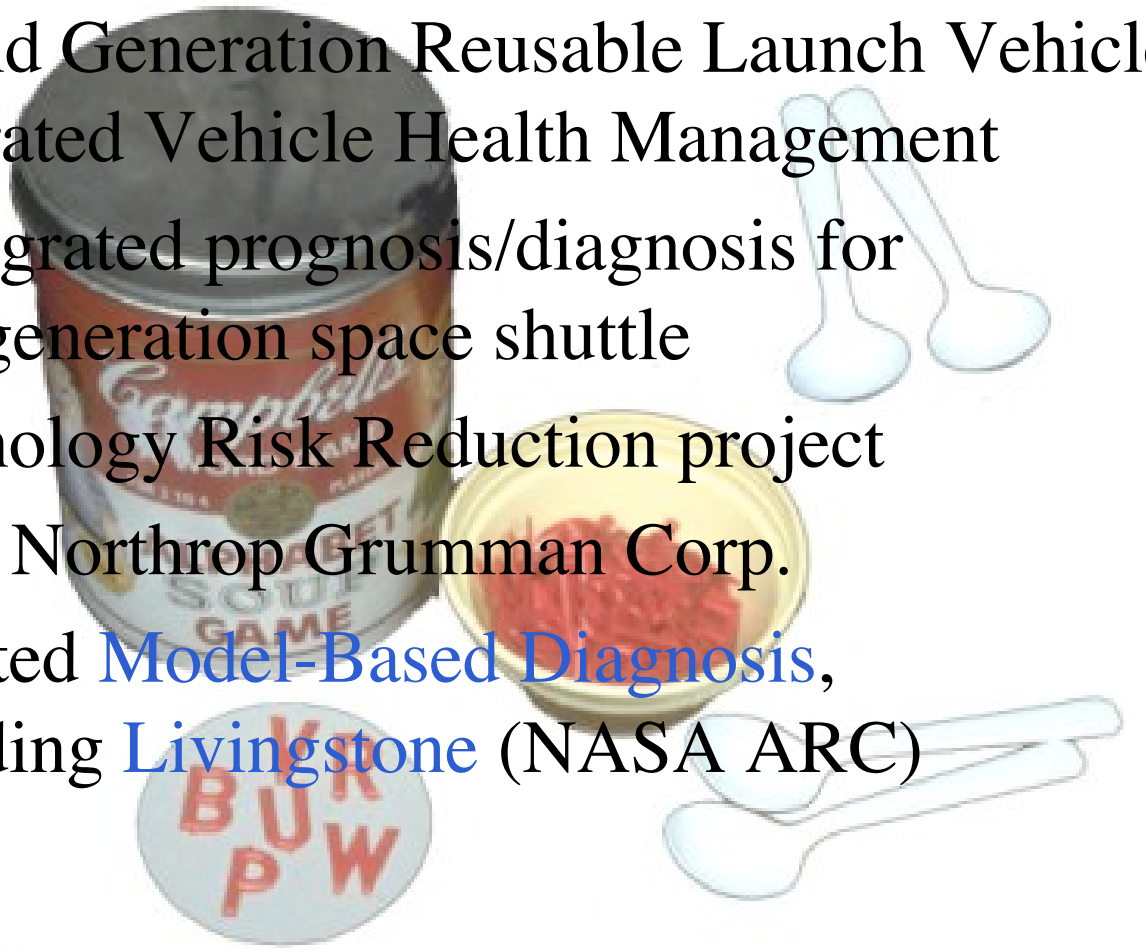
√ SUBSYSTEMS

Next-Gen Shuttle

Mission Control

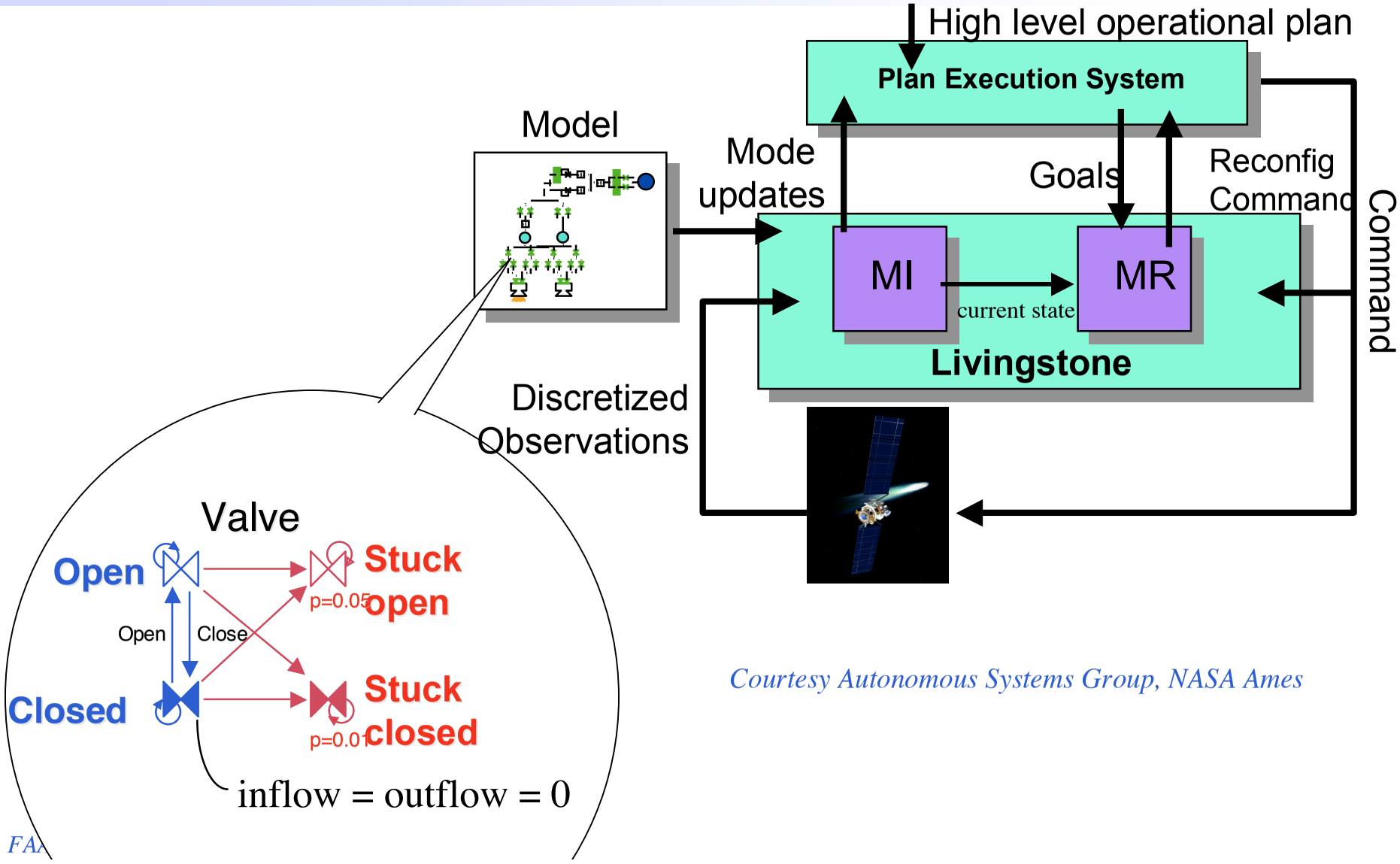


2nd Gen RLV IVHM



- Second Generation Reusable Launch Vehicle Integrated Vehicle Health Management
- = Integrated prognosis/diagnosis for next-generation space shuttle
- Technology Risk Reduction project
- Lead: Northrop Grumman Corp.
- Adopted Model-Based Diagnosis, including Livingstone (NASA ARC)

Livingstone



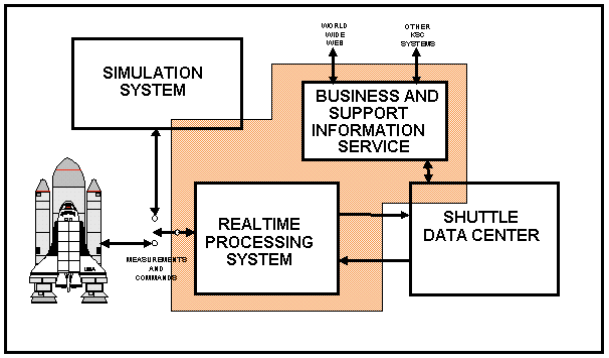
Courtesy Autonomous Systems Group, NASA Ames

Our Contributions

- First Phase: **Survey** (June 01 – March 02)
  - NASA Current V&V Practice
  - Applicable Formal Methods
  - Ames V&V Tools
  - Output: Three Reports (ase.arc.nasa.gov/vvivhm)
- Second Phase: **Tools** (April 02 – May 03)
  - Tool Extensions, GUI, Documentation, Integration
  - Output: Demonstrations (and Reports)

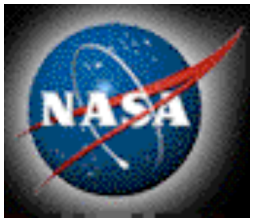
Survey of NASA V&V Processes/Methods

Deep Space One Remote Agent Experiment V&V



Checkout & Launch Control System (CLCS) V&V

X-37 IVHM Experiment V&V



NASA V&V Standards & DO-178B V&V Certification

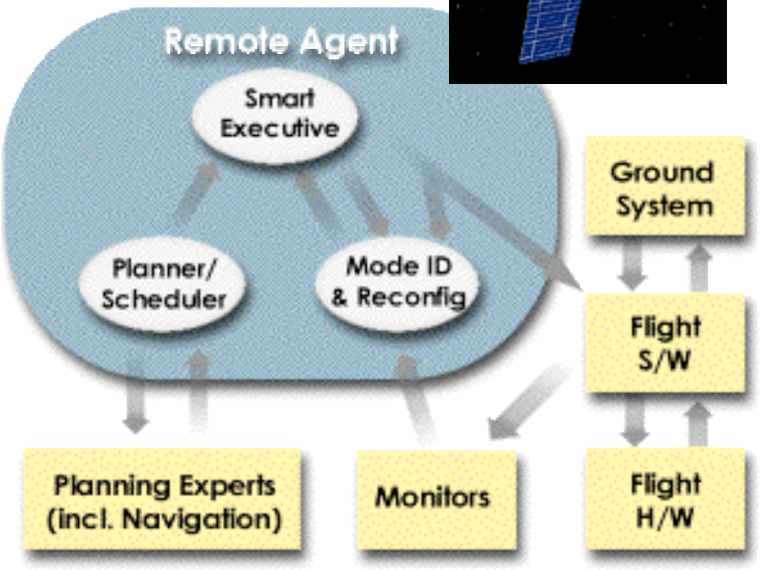
2nd Gen RLV IVHM V&V Requirements

DS-1 Remote Agent

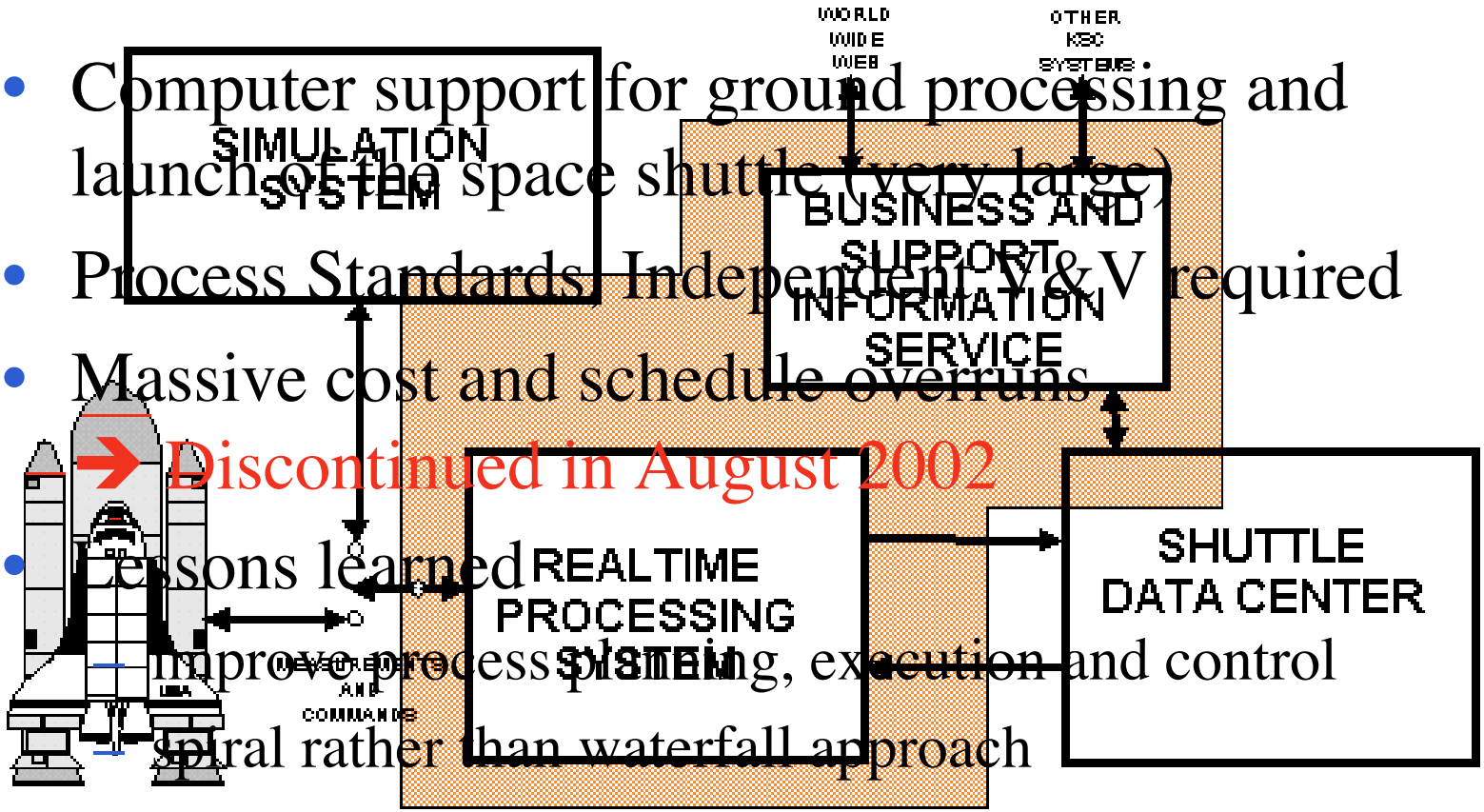
- Autonomous spacecraft controller
- 7 classes of testbeds

Testbed	Fidelity	Avail.	Speed
Unix ... DS-1	lowest ... highest	unlim. ... one	35:1 ... 1:1

- Change Control Board
  - As launch date gets closer, sometimes work around errors rather than fix them



Checkout & Launch Control System



- Computer support for ground processing and launch of the space shuttle (very large)
- Process Standards, Independent V&V required
- Massive cost and schedule overruns
  - → Discontinued in August 2002
- Lessons learned
  - Improve processing, execution and control
  - spiral rather than waterfall approach

X-37 IVHM Experiment

- Using Livingstone for IVHM of space vehicle
- Closest to 2nd Gen RLV IVHM
- Detailed V&V plan,
  follows NASA standards
- Early stage

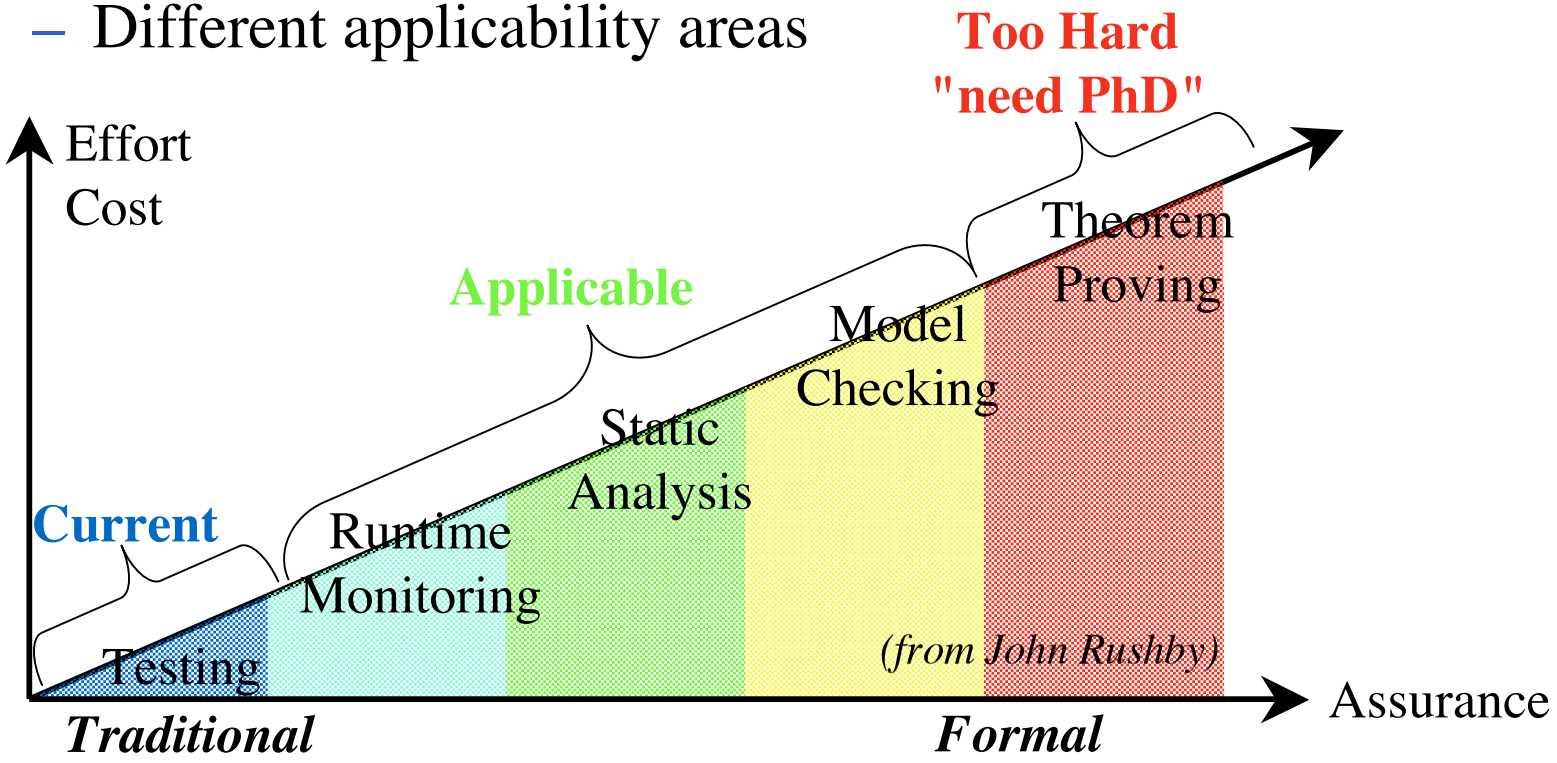
Software Process Standards

- NASA NPG 2820, based on IEEE/EIA 12207.{0,1,2}
  - Describes S/W lifecycle processes, support documents, implementation recommendations.
- NASA NPG 8730 (recently discontinued)
  - Covers Software Independent Verification and Validation (IV&V)
- Also relevant: RTCA DO-178B
  - Software for aviation, adopted by FAA

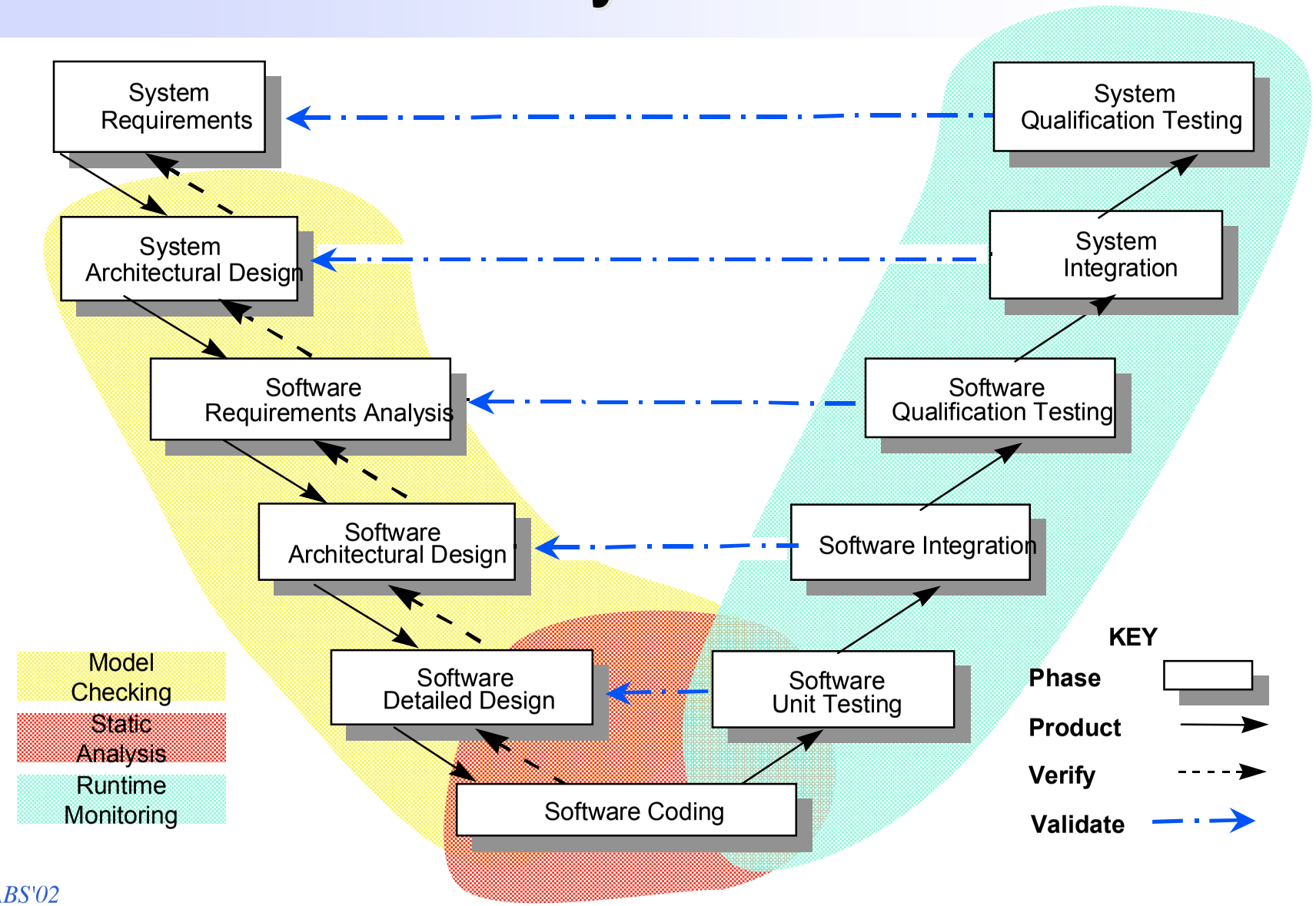
Prescribe precisely defined process with discrete phases and thoroughly documented work products at each phase

Formal Methods

- Different "formal" methods
  - Different strengths
  - Different applicability areas



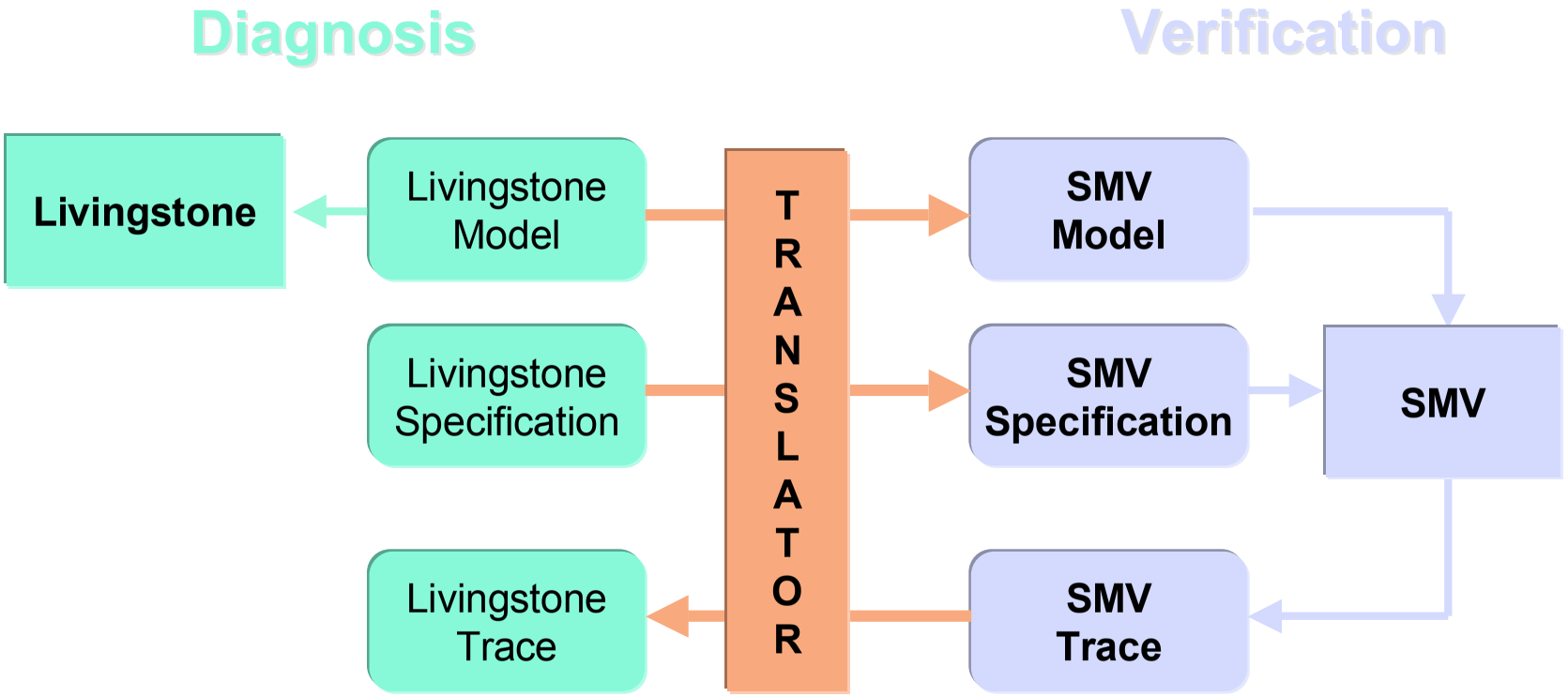
Formal Methods in the Software Lifecycle



NASA Examples

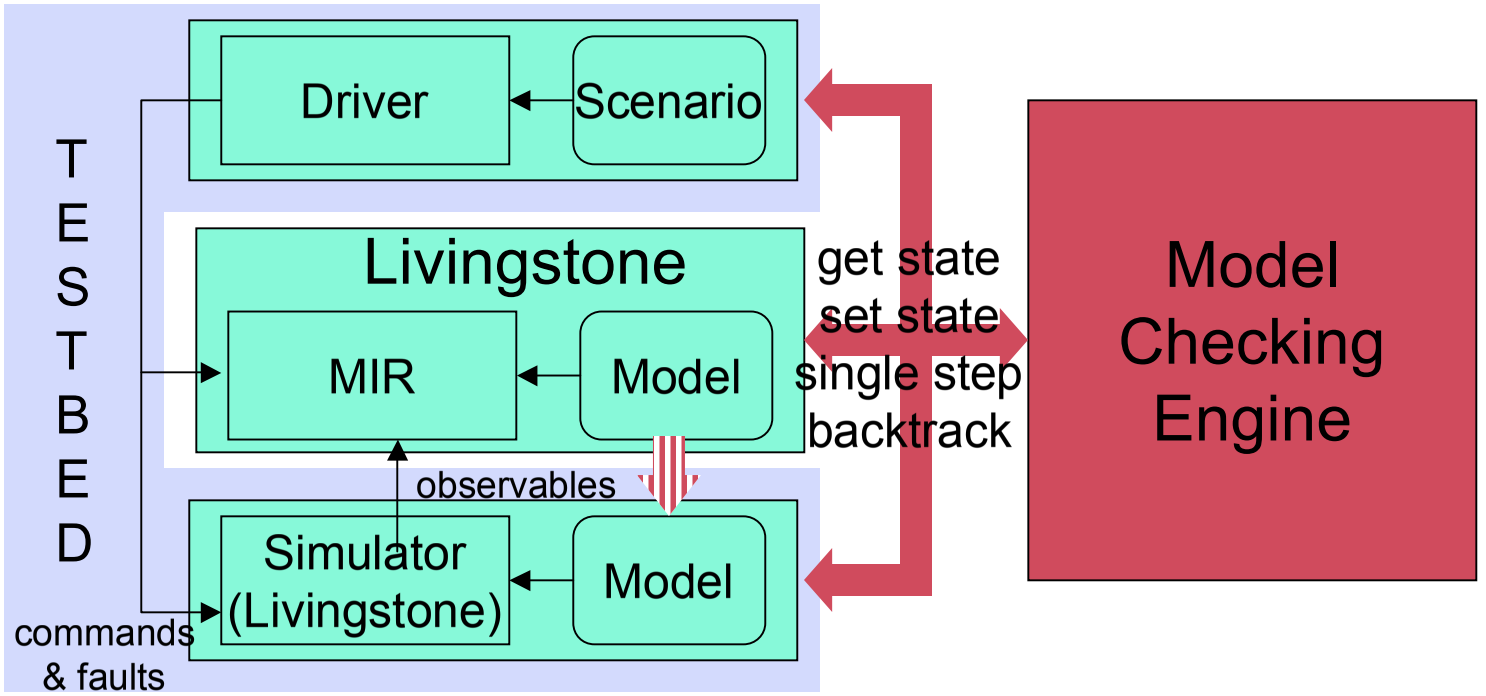
- Model Checking of Remote Agent [Havelund et.al.]
  - Detected errors similar to one that actually occurred in flight!
- Model Checking of Planning Models [Khatib et.al.]
  - Real-time models (uses UPPAAL)
- Lightweight FM for Remote Agent Exec [Feather et.al.]
  - Analyze execution traces a posteriori

Livingstone-to-SMV Translator



- SMV: symbolic model checker (BDD and SAT)
  can handle large state spaces, well suited for Livingstone
- Hide away SMV, offer a Livingstone model checker

Livingstone PathFinder (LPF)



- Start from Conventional Testing (the Real Program).
- Instrument the Code to be able to do Full Model Checking
  - or as close as possible

Correctness Criteria for Model-Based Diagnosis

1. **Engine Correctness**: the software is OK
   i.e. all that can be diagnosed is correctly diagnosed
2. **Model Correctness**: the model is OK
   i.e. the model is a valid abstraction of the plant
3. **Diagnosability**: the design is OK
   i.e. all that needs to be diagnosed can be diagnosed

In principle, 1+2+3 => diagnosis will be correct

In practice, compromises for efficiency purposes

- Model Verification can address **2** and **3**
- LPF can detect problems related to **1**, **2** or **3**

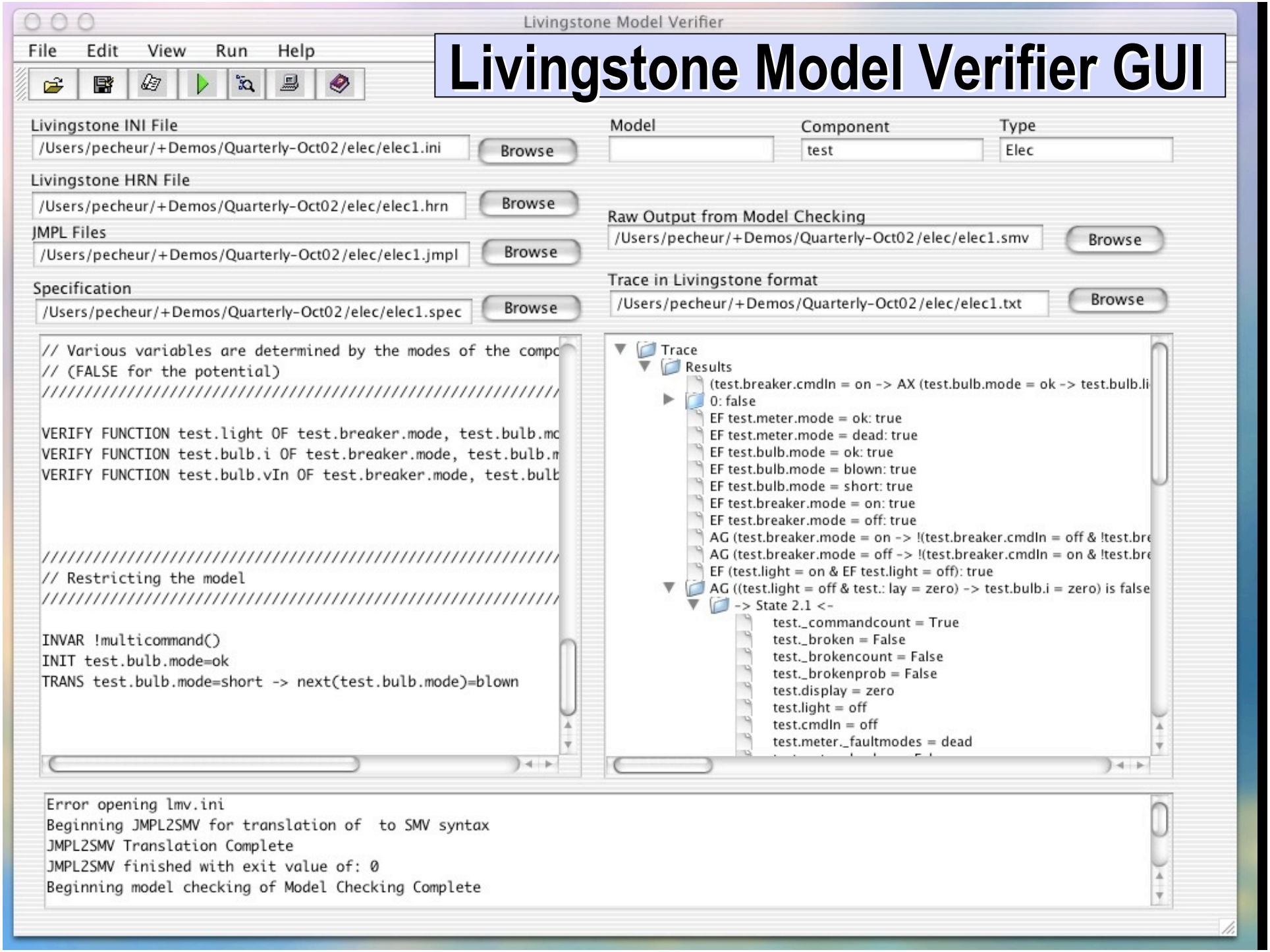
Current work:
Tool Maturation and Integration

Goal: Improve Usability of V&V Tools

- GUI (both translator and LPF)
- Translator: trace translation (SMV to Livingstone), more specification patterns
- V&V results tracking
- Documentation and Packaging

Also (other project): verification of **diagnosability**

- From observations, can some fault F always be detected?
- = model checking problem over twin model
- cf. MoChArt'02 paper (with A. Cimatti)


Livingstone Model Verifier
File Edit View Run Help
Livingstone Model Verifier GUI
Livingstone INI File
/Users/pecheur/+Demos/Quarterly-Oct02/elec/elec1.ini
Browse
Livingstone HRN File
/Users/pecheur/+Demos/Quarterly-Oct02/elec/elec1.hrn
Browse
JMPL Files
/Users/pecheur/+Demos/Quarterly-Oct02/elec/elec1.jmpl
Browse
Specification
/Users/pecheur/+Demos/Quarterly-Oct02/elec/elec1.spec
Browse
Model
Component
test
Type
Elec
Raw Output from Model Checking
/Users/pecheur/+Demos/Quarterly-Oct02/elec/elec1.smv
Browse
Trace in Livingstone format
/Users/pecheur/+Demos/Quarterly-Oct02/elec/elec1.txt
Browse
// Various variables are determined by the modes of the compo
// (FALSE for the potential)
VERIFY FUNCTION test.light OF test.breaker.mode, test.bulb.mo
VERIFY FUNCTION test.bulb.i OF test.breaker.mode, test.bulb.m
VERIFY FUNCTION test.bulb.vIn OF test.breaker.mode, test.bulb
// Restricting the model
INVAR !multicommand()
INIT test.bulb.mode=ok
TRANS test.bulb.mode=short -> next(test.bulb.mode)=blown
Trace
Results
(test.breaker.cmdIn = on -> AX (test.bulb.mode = ok -> test.bulb.li
0: false
EF test.meter.mode = ok: true
EF test.meter.mode = dead: true
EF test.bulb.mode = ok: true
EF test.bulb.mode = blown: true
EF test.bulb.mode = short: true
EF test.breaker.mode = on: true
EF test.breaker.mode = off: true
AG (test.breaker.mode = on -> !(test.breaker.cmdIn = off & !test.bre
AG (test.breaker.mode = off -> !(test.breaker.cmdIn = on & !test.bre
EF (test.light = on & EF test.light = off): true
AG ((test.light = off & test.: lay = zero) -> test.bulb.i = zero) is false
-> State 2.1 <-
test._commandcount = True
test._broken = False
test._brokencount = False
test._brokenprob = False
test.display = zero
test.light = off
test.cmdIn = off
test.meter._faultmodes = dead
Error opening lmv.ini
Beginning JMPL2SMV for translation of to SMV syntax
JMPL2SMV Translation Complete
JMPL2SMV finished with exit value of: 0
Beginning model checking of Model Checking Complete

Conclusions and Perspectives

- This is a limited effort
  - Few selected examples, but illustrative
  - Demonstration-level prototypes
- New space applications ask for advanced software
  Advanced software asks for advanced V&V
- Integrate into rigid, conservative practices
- Make methods and tools usable by practitioners
  - rather than the other way round...
- There is demand in the industry!