Verification and Validation of Integrated Vehicle Health Management

Charles Pecheur (RIACS)

with contributions from Stacy Nelson (Nelson Consulting)

Outline

- **V&V of Model-Based Diagnosis**
  - **Concepts, Approaches, Tools.**
- V&V of IVHM for Next-Gen. Shuttle
  - Highlights of work performed for SLI under the Northrop-Grumman contract.
- V&V Tool Demonstration
  - Description of example used and results.

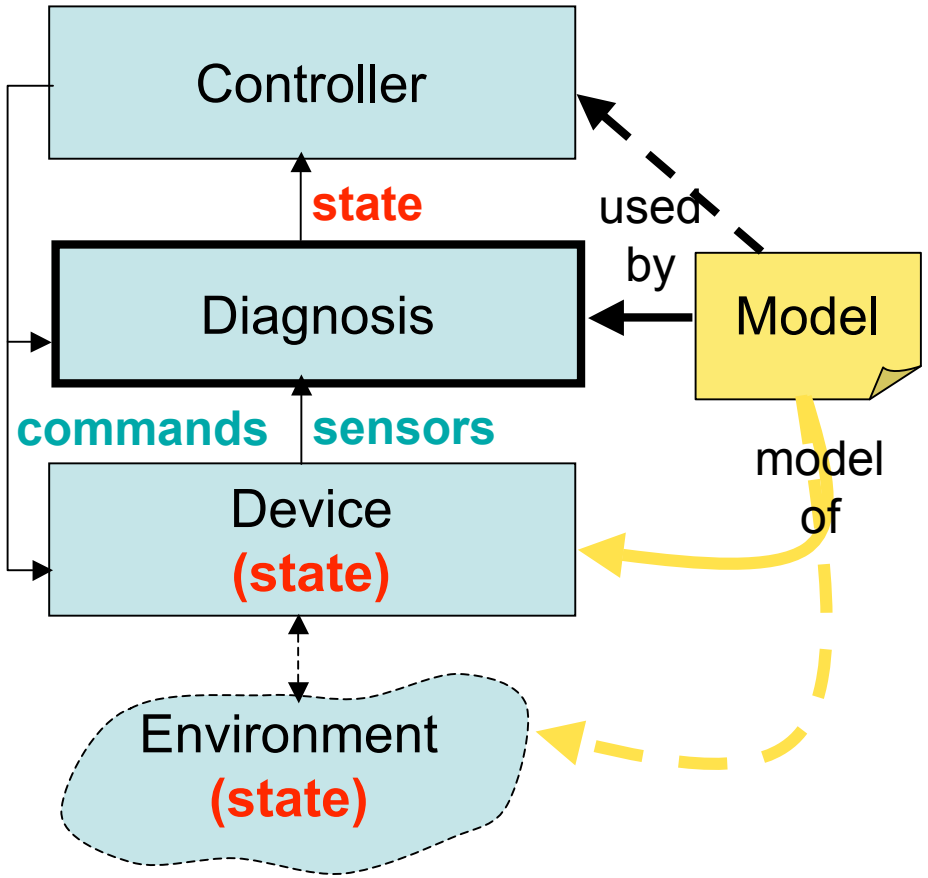
V&V of Advanced Diagnosis

- Future space missions need **extended diagnosis capabilities**
  - to extract and correlate information from a **larger array of components**
  - to be able to handle a **larger range of unpredictable scenarios**
- The **space** of possible situations **increases dramatically**
- **Extended V&V** capabilities are needed
  - Test **more cases**, **faster**, **automatically**
  - Analyze **coverage**, cover many cases with one test
  - **Design for V&V**, perform V&V **early**, take advantage of **high-level models**

Diagnosis

- Fault Protection = **F**ault
  - **D**etection
  - **I**dentification
  - **R**ecovery

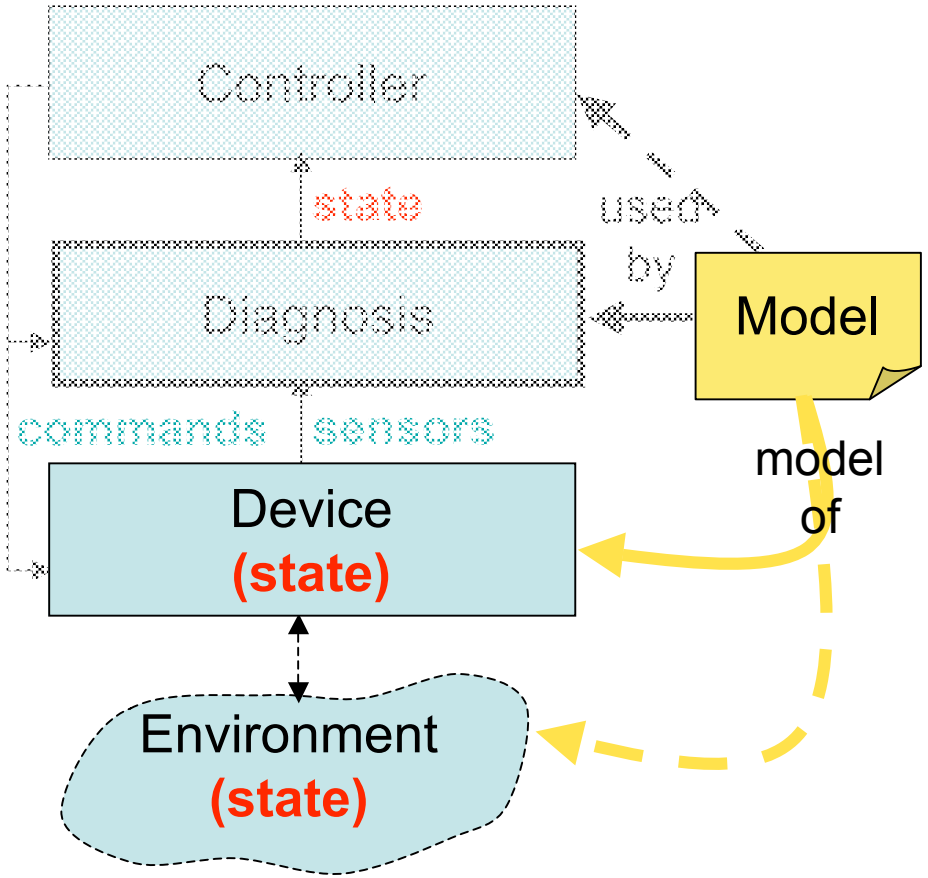
  } **Diagnosis**
- **Goal**: determine hidden state from visible commands and sensors
- **Model** used to build diagnosis, at design time and/or at run time
- Recovery is part of Controller



V&V Criteria for Diagnosis: Model Correctness

Is the model valid w.r.t. the physical device?

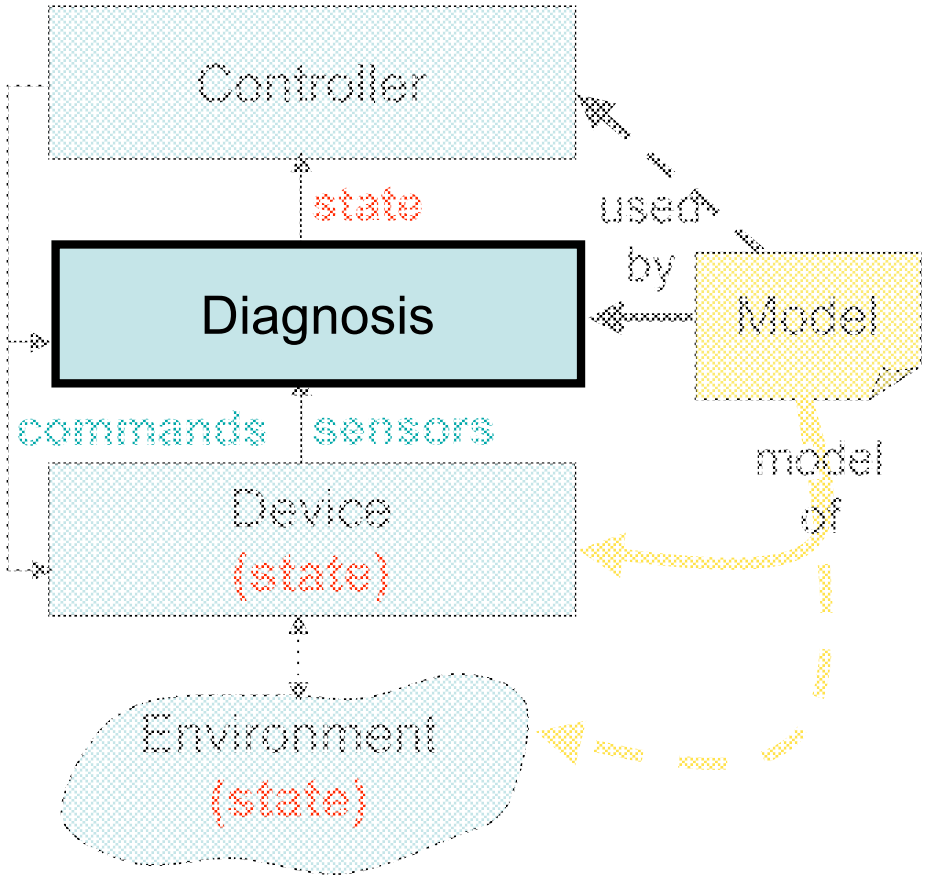
- Is it internally well-formed (complete, consistent, ...)?
- Does it correctly model the device specs?
- Do the specs correctly capture the physical device?



V&V Criteria for Diagnosis: Program Correctness

Does the actual program perform according to specifications?

- Is it free from programming defects (array bounds, pointers, etc)?
- Are the algorithms correct?
- Does the code correctly implement them?

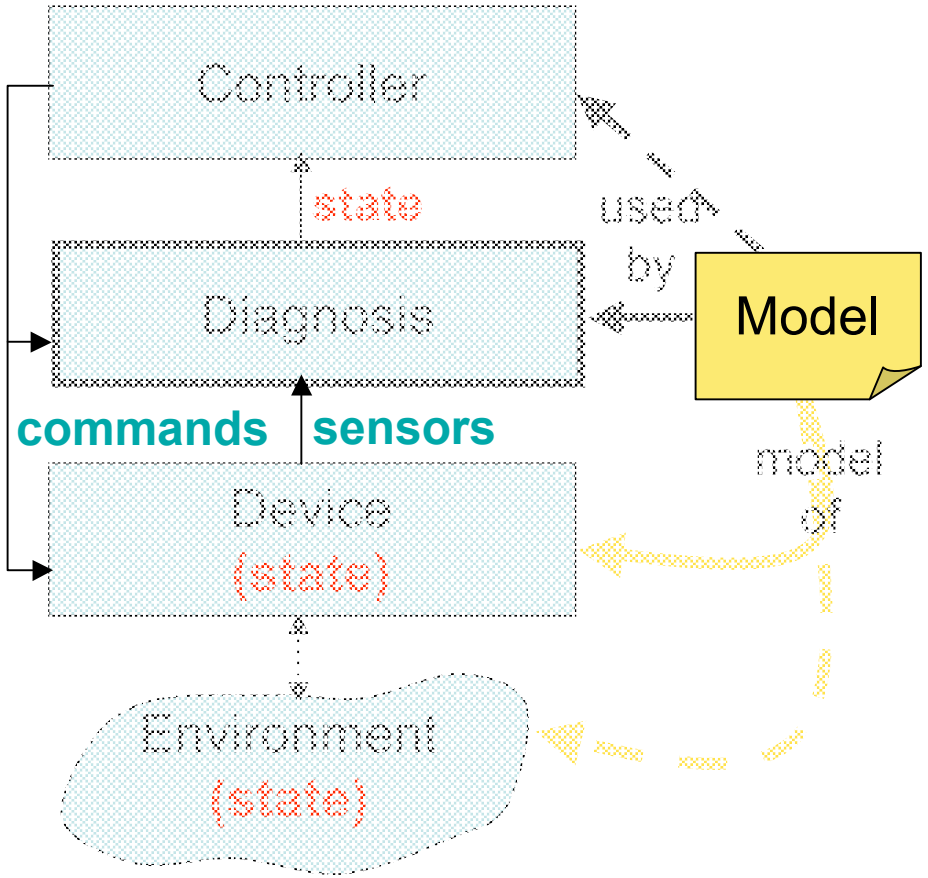


V&V Criteria for Diagnosis: Diagnosability

Is it possible to perform the required diagnosis, given the available data?

According to the model
(assuming model correctness),

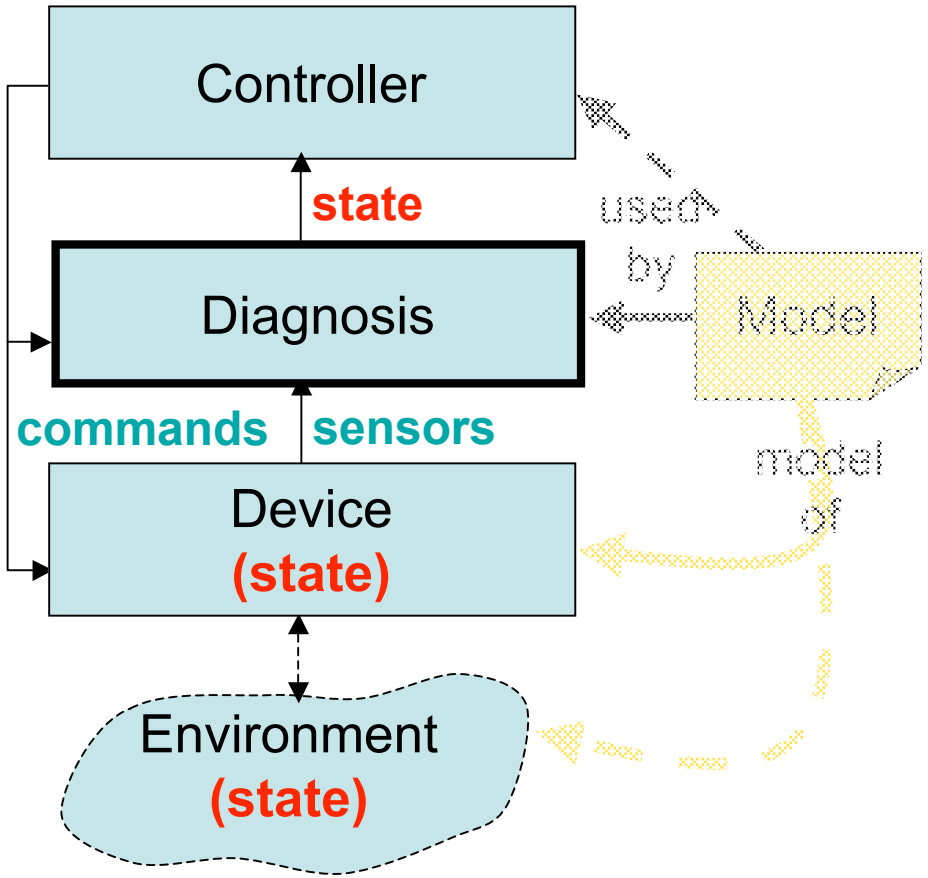
- ... can faults be detected as required?
- ... can fault groups be reduced as required?



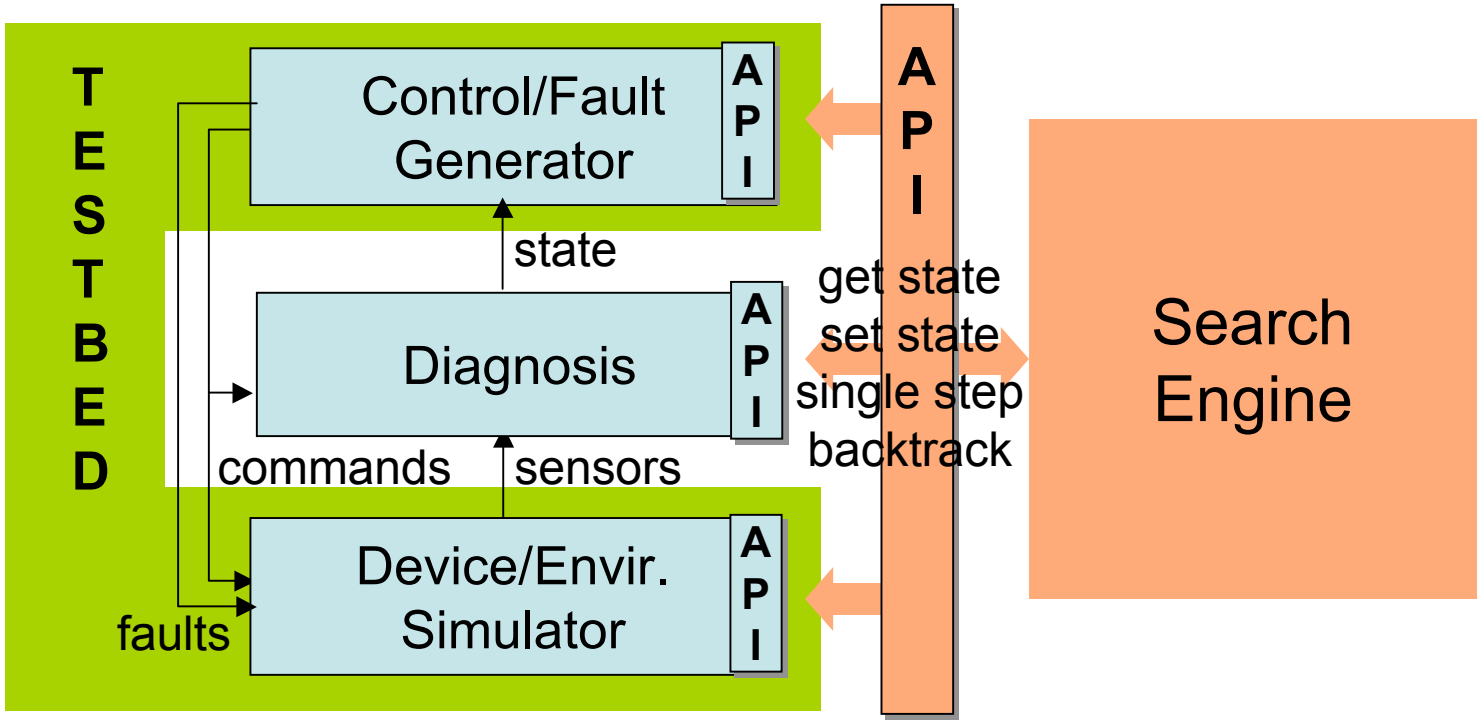
V&V Criteria for Diagnosis: Integration Correctness

Does the combination of the different parts work as expected?

- Does the operating framework properly supports the components and interactions?
- Is the provided diagnosis adequate w.r.t. the rest of the controller?
- Is the integrated system free of unwanted interferences?

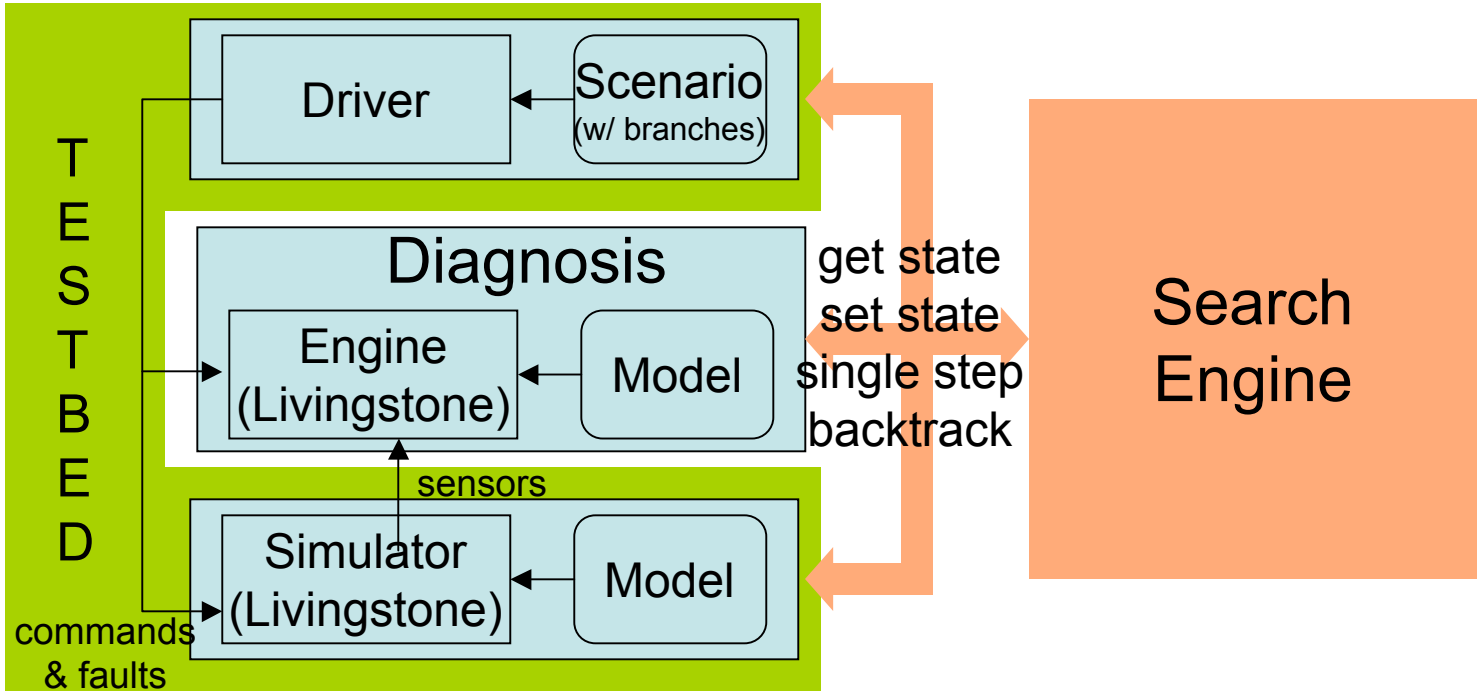


Simulation-Based V&V



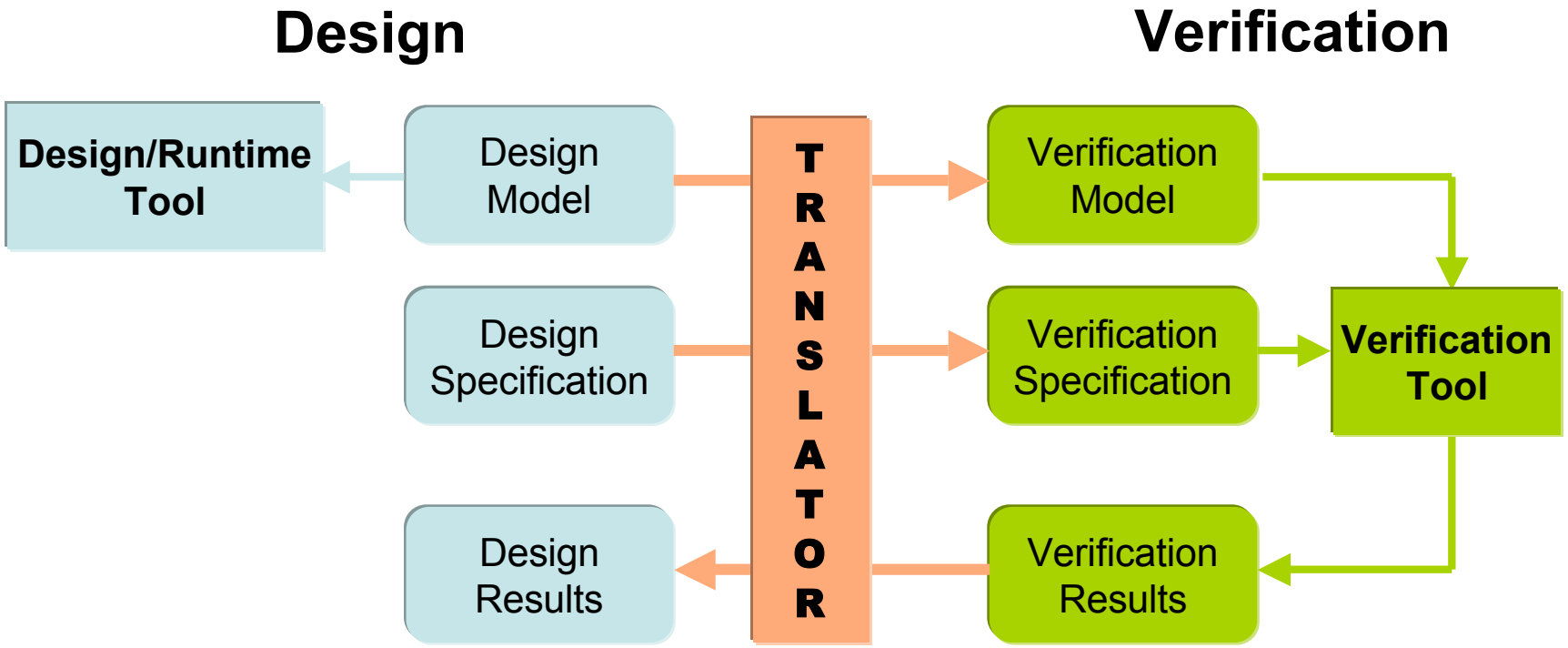
- Execute the **Real Program** in a simulated environment (testbed)
- **Instrument** the Code to be able to **backtrack** between alternate paths
- **Modular** architecture, allows different diagnosis, simulators, search algorithms
- Expands **conventional testing** with **model checking** concepts
  - Increased automation reduces test suite development costs
  - Optimized execution (backtracking) reduces test execution times
  - Modularity allows easy configuration to adjust fidelity, coverage, speed, focus, ...

Livingstone PathFinder (LPF)



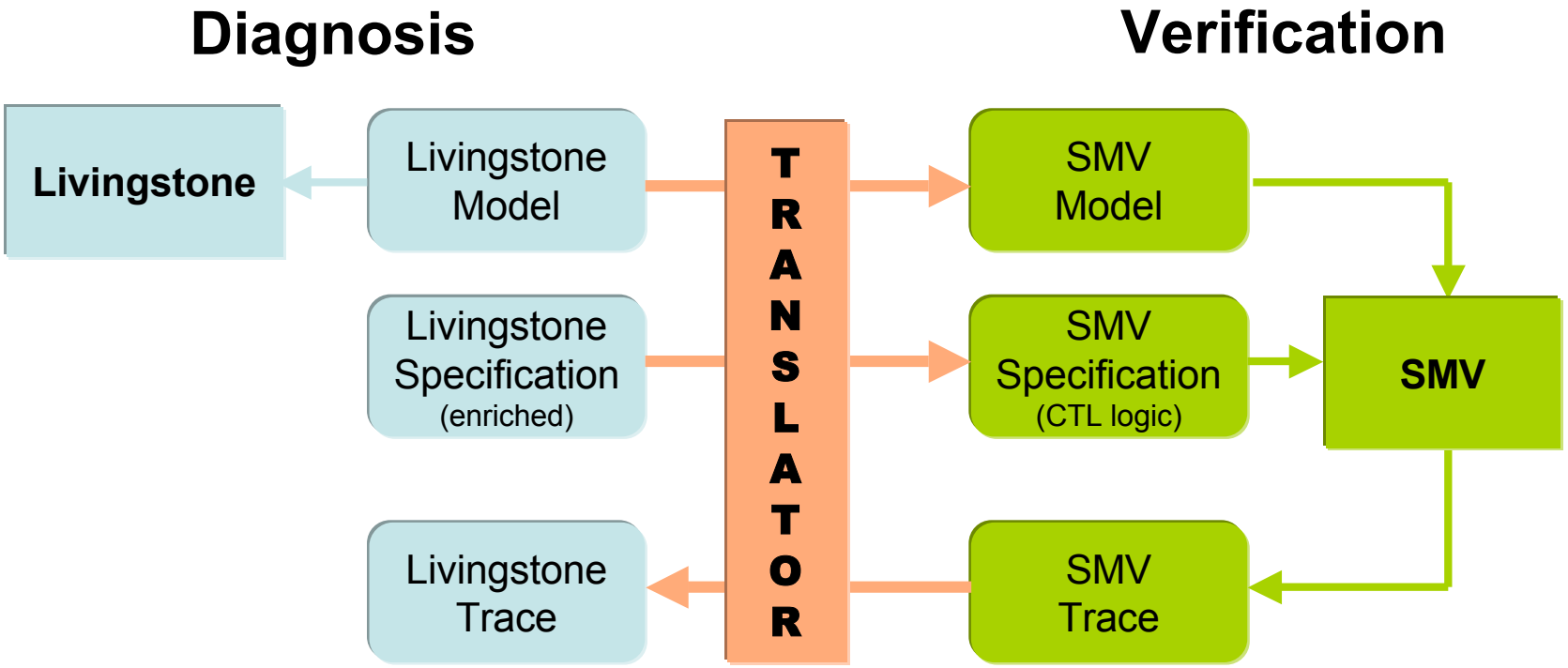
- Simulation-Based V&V for the Livingstone diagnosis system
- Uses Livingstone engine for simulator too
  - Other simulators can be substituted where available
- Scenario=non-deterministic program
  - Typically: a sequence of commands with one among a set of faults occurring anywhere

Model-Based V&V



- Apply verification tools to design models
- Translator hides away specificities of Verification Tool
- High-level models amenable to exhaustive analysis (e.g. model checking)
- Model-based diagnosis can use the same model!

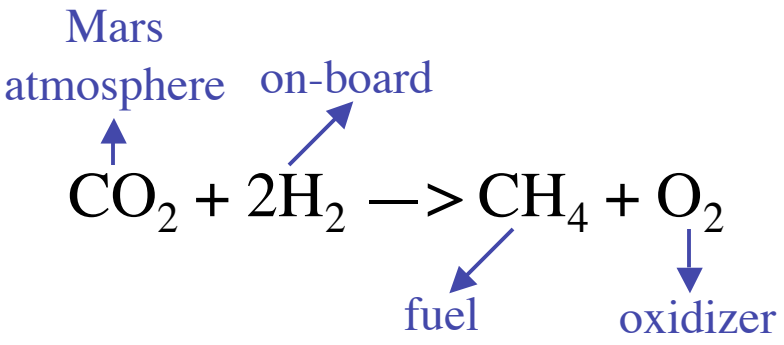
Livingstone-to-SMV Translator

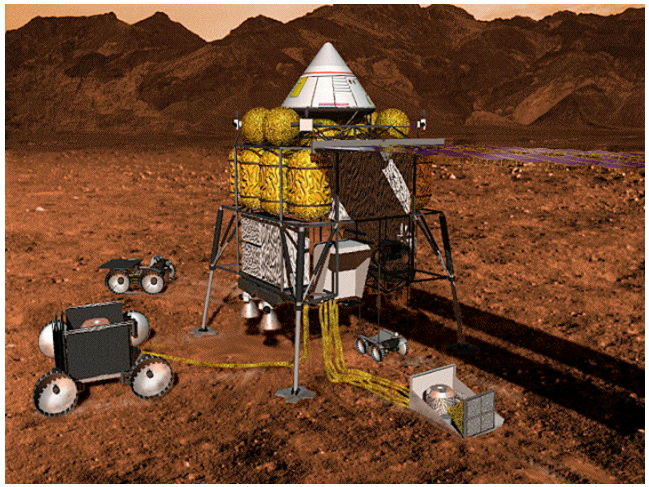


- Allows **exhaustive** analysis of Livingstone models (10^{50+} states)
- Uses **SMV**: symbolic model checker (BDD and SAT)
- Enriched spec syntax (vs. SMV's core temporal logic)
- Hide away SMV, offer a model checker for Livingstone
- Graphical interface, trace display

V&V of Models Example: In-Situ Propellant Production

- Use atmosphere from Mars to make fuel for return flight.
- Livingstone-based controller developed at NASA KSC.
- Largest model is 10^{55} states.
- Live experience of V&V methods used by non-specialists.
- SMV Exposed several modeling errors.





In-Situ Propellant Production Errors Found

- *"If the outlet was zero admittance, then there can be no flow in the z-flow module"*

  VERIFY INVARIANT (ispp.admittance.outlet=off ->
  ispp.z-flow-module.flow=off)

  Result shows a trace to a state where admittance is off and there is flow.

- *"The relative flow in the RWGS trap is a function of the input and output flows"*

  VERIFY FUNCTION rwgs.rwgs_trap.relative_flow
  OF rwgs.rwgs_trap.flow_in, rwgs.rwgs_trap.flow_out

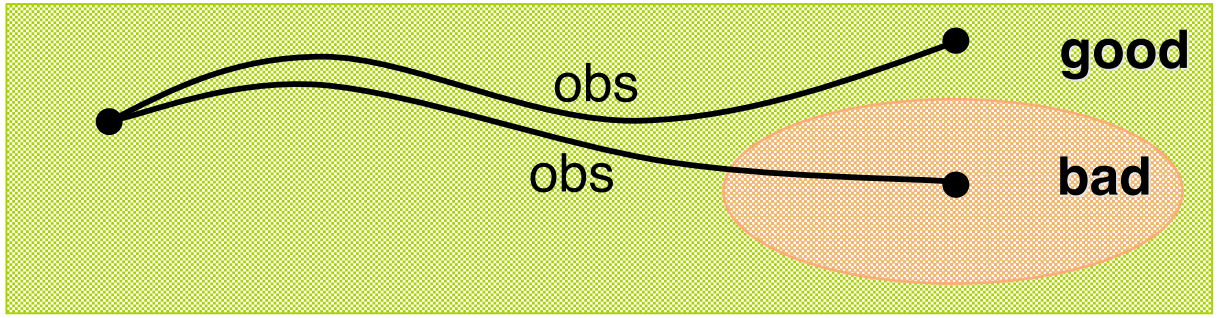
  Result shows two traces to states with the same flow_in and flow_out and different relative_flow.

Note: old data re-formatted using new tool features

Verification of Diagnosability

Q: From observations (input/output), can diagnosis always tell when plant comes to a bad state?

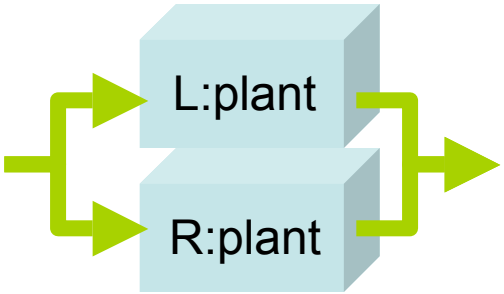
A: **YES unless** plant can go good or bad with the same observations (and therefore diagnosis cannot tell)



Verification using model checking (SMV)

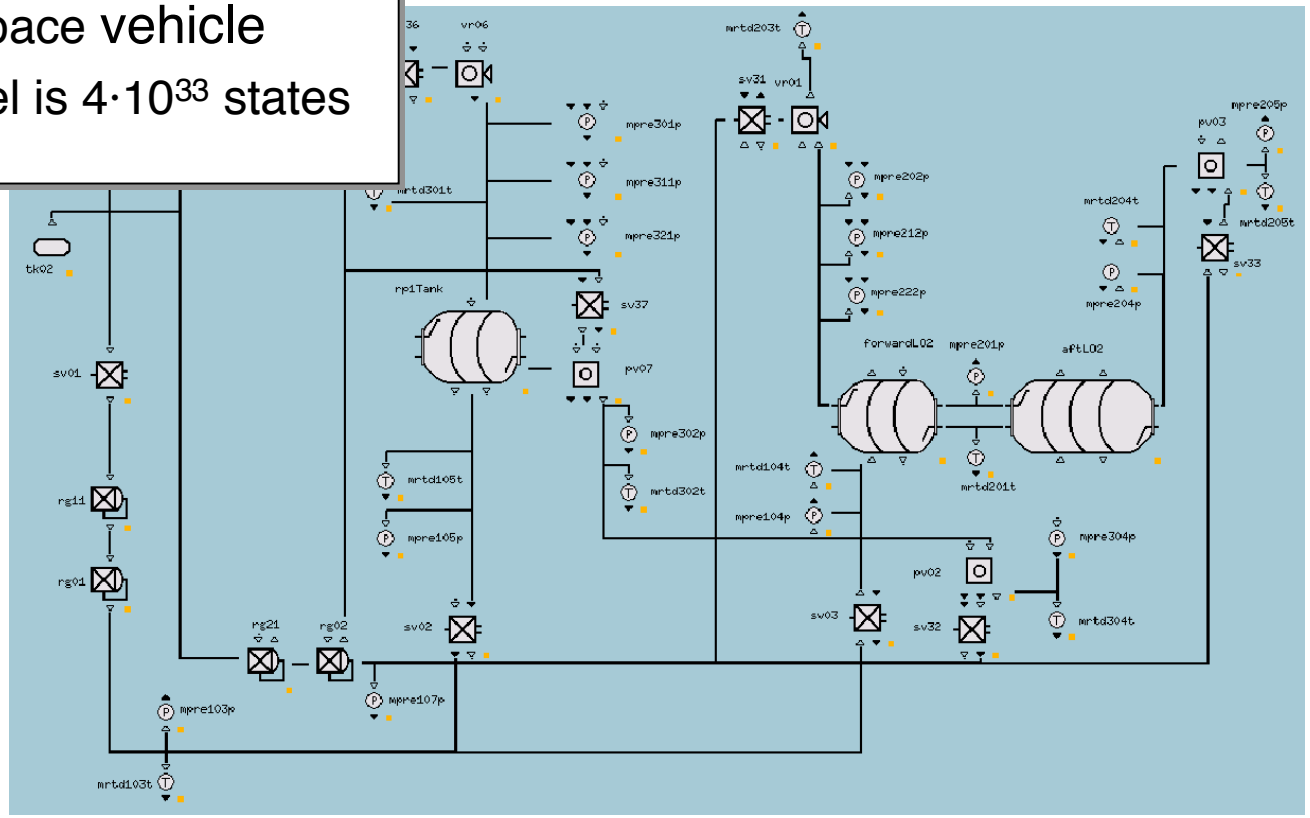
- Two "siamese twin" copies of the plant (L/R), with coupled observations
- verify that one cannot reach:

(L **in** good) **and** (R **in** bad)

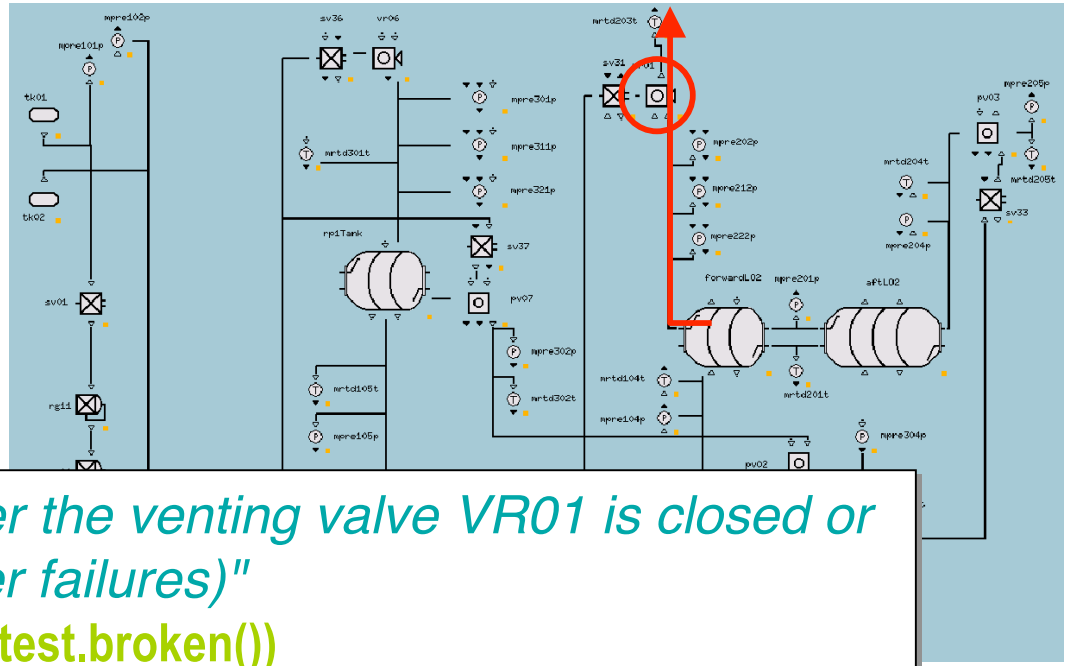


X-34 / PITEX

- Propulsion IVHM Technology Experiment (ARC, GRC)
- Livingstone applied to propulsion feed system of space vehicle
- Livingstone model is $4 \cdot 10^{33}$ states



PITEX Diagnosability – Error Found



- *"Diagnosis can decide whether the venting valve VR01 is closed or stuck open (assuming no other failures)"*
  INVAR !test.multibroken() & twin(!test.broken())
  VERIFY INVARIANT !(test.vr01.mode=stuckOpen &
  twin(test.vr01.valvePosition=closed))
  Results show a pair of traces with same observations, one leading to **VR01 stuck open**, the other to **VR01 closed**. Application specialists fixed their model.

V&V Solutions for Diagnosis

- **Model Correctness**
  - **Model-Based V&V** for (generic) well-formedness, (specific) documented properties of the device
  - **Testing**, **Simulation-Based V&V** for model-based diagnosis
  - **Compared Simulation** of high-level vs. high-fidelity models
- **Program Correctness**
  - **General Software V&V**: proofs of algorithms, static analysis for runtime errors, model checking for concurrency, ...
  - **Testing**, **Simulation-Based V&V**
  - For re-usable parts (inference engine), **one-time** V&V effort, then increased confidence from repeated use (cf. Java VM)

V&V Solutions for Diagnosis (cont'd)

- **Diagnosability**
  - **Model-Based V&V** using twin model approach or other
  - **Testing**, **Simulation-Based V&V**
  - This is a system design issue
- **Integration Correctness**
  - Mostly **Testing**, especially once hardware is included
  - **Simulation-Based V&V** for software-level integration, extended to include controller (and planner etc.)
  - **General Software V&V** on framework/support code
  - **Compositional reasoning**: assume/guarantee, program-by-contract

Related Work

- **DS1 Remote Agent** (Havelund-Lowry-Penix, ARC)
  - Focus on Executive
  - Parts model-checked at Ames in 1997, 5 errors found
  - Deadlock during flight in 1999, error similar to one of those found (but in a different part)
- **HSTS Planner Models** (Havelund-Pecheur-Penix, ARC)
  - Early experiment in model-based V&V at Ames
  - Compared 3 model checkers
- **Lightweight Formal Methods** (Feather-Smith, JPL)
  - Verify generated plans against flight rules
  - Use database: plans as data, properties as queries

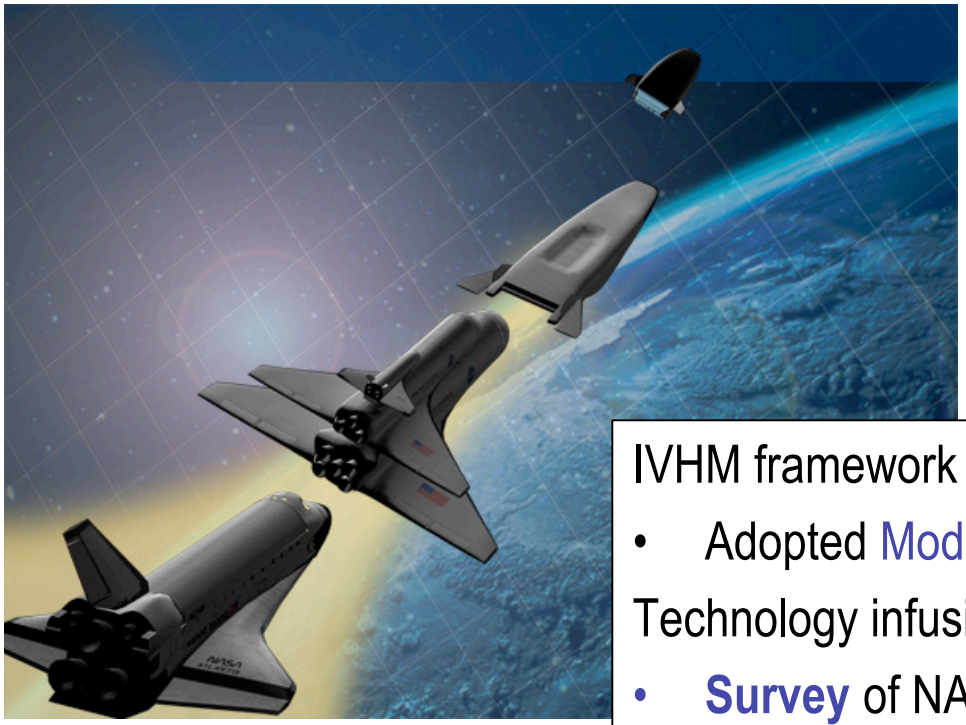
Conclusions

- Advanced diagnosis demands advanced V&V
- **Model-Based V&V**:
  - Not restricted to model-based diagnosis (but same model can be used for diagnosis and V&V)
  - High-level, formal model enables early and thorough analysis
- **Simulation-Based V&V**:
  - Extends testing to better speed, automation, coverage
  - On finished/refined product: less thorough but more accurate
- General software practices and processes still apply
- **ARC** can provide: – guidance on general issues,
  – tools for specific parts.

Outline

- V&V of Model-Based Diagnosis
  - Concepts, Approaches, Tools.
- **V&V of IVHM for Next-Gen. Shuttle**
  - **Highlights of work performed for SLI under the Northrop-Grumman contract.**
- V&V Tool Demonstration
  - Description of example used and results.

Verification of IVHM for Next-Gen Space Vehicle



IVHM framework developed by Northrop Grumman Corp.

- Adopted Model-Based Diagnosis, including Livingstone

Technology infusion project:

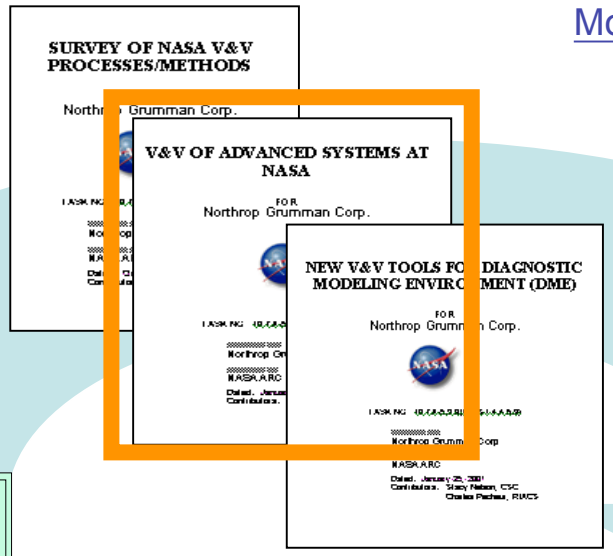
- **Survey** of NASA current V&V practice, applicable formal methods, our verification tools
  See ase.arc.nasa.gov/vvivhm
- **Maturation** of Livingstone verification tools (translator and LPF): tool extensions, GUI, improved documentation and packaging, integration with other IVHM tools

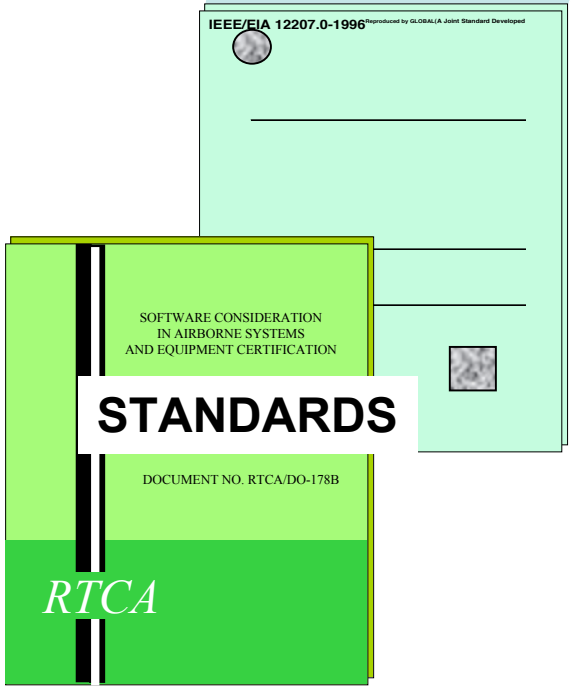
CASE STUDY: V&V of IVHM – Risk Reduction

More Info: http://ase.arc.nasa.gov/vvivhm

GOAL: **Formal verification of diagnostic systems based on NASA and FAA safety critical certification standards:**

IEEE 12207 and DO-178B





BENEFIT: Reduce risk for developing IVHM systems used on 2nd Gen RLV

KEY RESULTS:

Three reports, two improved tools

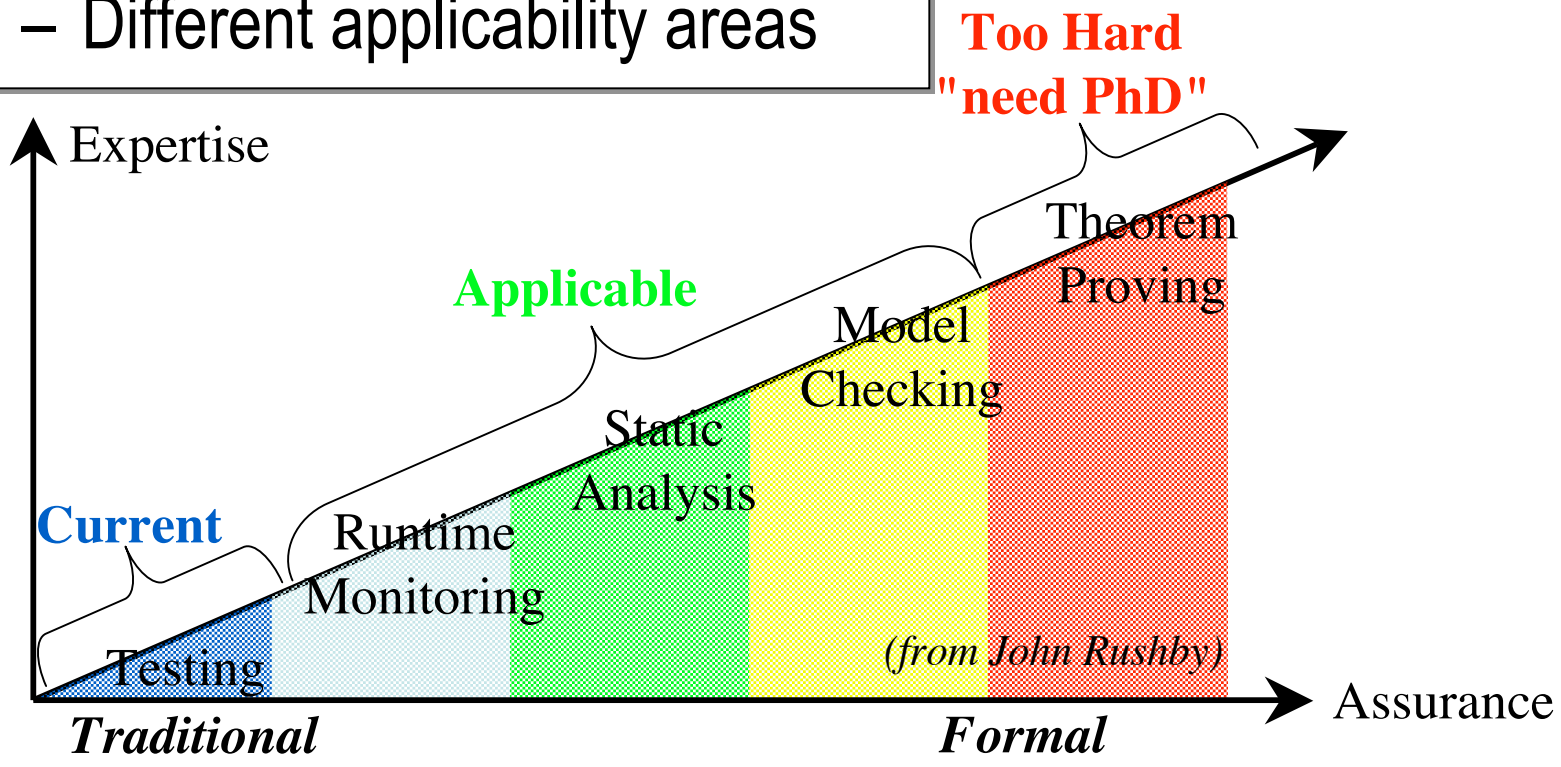
- NASA/CR-2002-211401 – Survey of NASA V&V Processes/Methods
- **NASA/CR-2002-211402 – V&V of Advanced Systems at NASA**
- NASA/CR-2002-211403 – New V&V tools for Diagnostic Modeling Environment (DME)

- Livingstone Model Verifier/JMPL2SMV tool (model checking)
- Livingstone PathFinder tool (simulator)

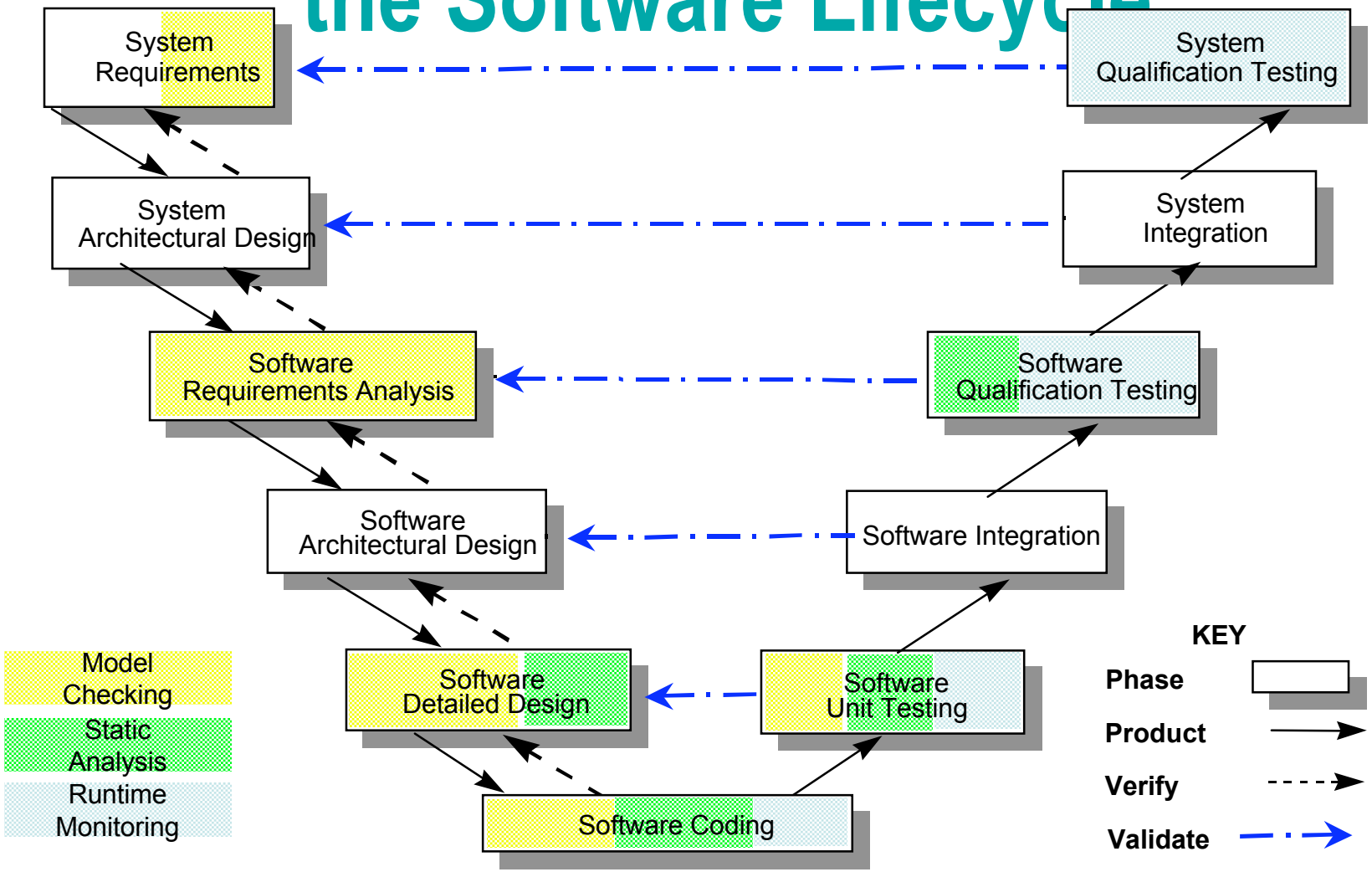
2nd Gen RLV

Formal Methods

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



Formal Methods in the Software Lifecycle



New V&V Processes

Formal Methods	Applicable SW Life Cycle Phase	Formal Verification Activities
Any	System Requirements Analysis SW Requirements Analysis	Perform a new development activity called "formalization" during which a new work product called a "formal specification" is created. This can be a separate product or an addition to an existing work product such as a requirements document. Documenting requirements reduces confusion later in the project and promotes customer approval of the software and system. Creating a formal specification enables the application of formal methods at later stages. It can also increase the accuracy of requirements and promote communication between developers and test engineers.
Model Checking (Theorem Proving)	System Requirements Analysis SW Requirements Analysis	Perform a new analysis activity called "proving assertions" to enhance the correctness of the formal specification and to understand the implications of the design captured in the requirements and specification.

New V&V Processes (cont'd)

Formal Methods	Applicable SW Life Cycle Phase	Formal Verification Activities
Static Analysis	SW & Model Detailed Design SW Coding SW & Model Unit Testing SW Qualification Testing	Use Static Analysis tools in addition to a compiler during code development. This can reduce the amount of traditional unit testing and even system-level qualification testing required while increasing the accuracy of the program. Static Analysis may also be applicable at the later stages of the Detailed Design phase.
Model Checking	SW Coding SW & Model Unit Testing	If available for the programming language and platform used, use model checkers in addition to standard debugging and test control tools. This can greatly improve the odds of detecting some errors, such as race conditions in concurrent programs.
Runtime Monitoring	SW Coding SW & Model Unit Testing SW Qualification Testing System Qualification Testing	Use Runtime Monitoring during simulation testing at each phase where program code gets executed. This can provide more information about potential errors.

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

V&V Tool Maturation

Goal: Improve Usability of Validation and Verification Tools

- LMV Trace Translation √
  - From SMV Back to Livingstone
- LMV New Specification Patterns √
  - Easier to Use than Temporal Logic
- LMV Control Center √
  - GUI for Setting Parameters, Running, Viewing Results
- LPF Control Center √
  - GUI for Setting Parameters, Running, Viewing Results
- Documentation and Packaging √
  - Extend Documentation, Simplify Installation

Future Work

- Continued development of current methods and tools
  - New target diagnosis systems, simulators, search algorithms
  - Case studies, Experiments
  - Maturation (user interface, documentation, integration in design environments, technology infusion)
- Address Fault Recovery
  - Include reactive control with fault remediation in Simulation-Based V&V
  - Apply Model-Based V&V to models that include control

To Probe Further

On-Line

- **Livingstone to SMV Translator:** ase.arc.nasa.gov/mpl2smv
- **Livingstone PathFinder:** ase.arc.nasa.gov/lpf
- **Verification of IVHM:** ase.arc.nasa.gov/vvivhm

Publications

- Stacy Nelson, Charles Pecheur. **Formal Verification of a Next-Generation Space Shuttle.** FAABS II, Greenbelt, MD, October 2002. To be published in LNCS.
- Charles Pecheur, Alessandro Cimatti. **Formal Verification of Diagnosability via Symbolic Model Checking**. MoChArt-2002, Lyon, France, July 2002.
- Steven Brown, Charles Pecheur. **Model-Based Verification of Diagnostic Systems.** Proceedings of JANNAF Joint Meeting, Destin, FL, April 8-12, 2002.
- Charles Pecheur, Reid Simmons. **From Livingstone to SMV: Formal Verification for Autonomous Spacecrafts.** FAABS I, April 2000. LNCS 1871, Springer Verlag.

Reports

- Stacy Nelson, Charles Pecheur. **NASA processes/methods applicable to IVHM V&V**. NASA/CR-2002-211401, April 2002.
- Stacy Nelson, Charles Pecheur. **Methods for V&V of IVHM intelligent systems**. NASA/CR-2002-211402, April 2002.
- Stacy Nelson, Charles Pecheur. **Diagnostic Model V&V Plan/Methods for DME.** NASA/CR-2002-211403, April 2002.
- Charles Pecheur. **Verification and Validation of Autonomy Software at NASA**. NASA/TM 2000-209602, August 2000.

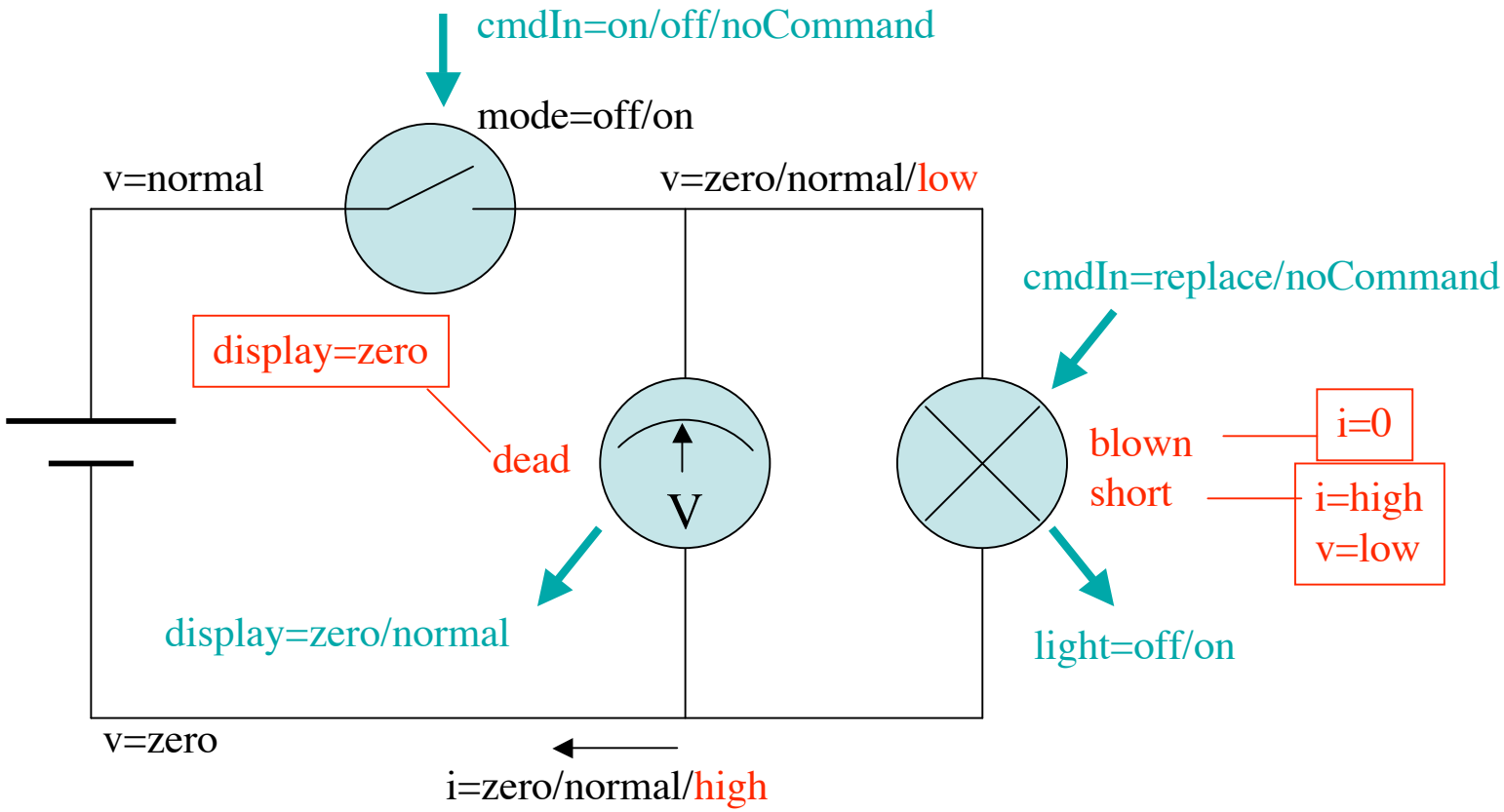
Publications and Reports available on-line at:

http://ase.arc.nasa.gov/pecheur/publi.html

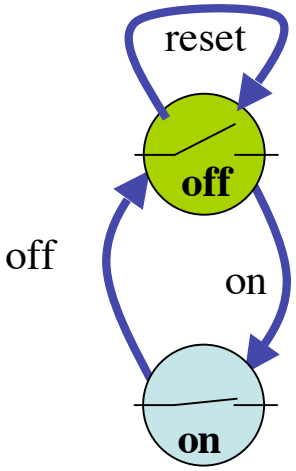
Outline

- V&V of Model-Based Diagnosis
  - Concepts, Approaches, Tools.
- V&V of IVHM for Next-Gen. Shuttle
  - Highlights of work performed for SLI under the Northrop-Grumman contract.
- **V&V Tool Demonstration**
  - **Description of example used and results.**

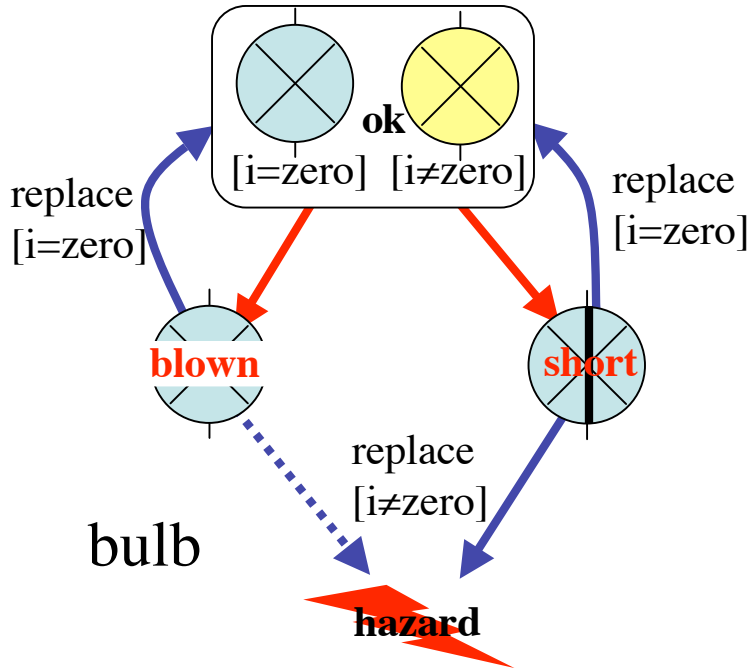
Demonstration
The Electric Model



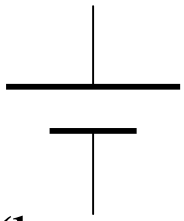
Electric Model Components



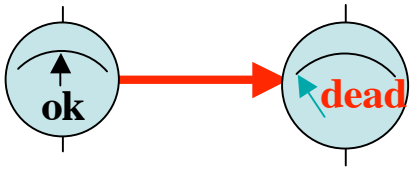
breaker



bulb



(battery)



meter

Demo: LMV and LPF on Elec

- Elec in Oliver
- LMV on Elec
- LPF on Elec
- Replay LPF Traces in Oliver

NB: **Oliver** (a.k.a. Stanley II) is the graphic development/simulation environment for Livingstone models.

LMV on PITEX

- 1-month experiment in Oct-Nov 02
  - by Roberto Cavada (IRST, NuSMV developer)
- Focus on diagnosability
- Goals
  - Evaluate scalability
  - Refine wrt. application needs
- Compared NuSMV variants
  - BDD vs. SAT, found SAT much better
- Found application-relevant anomaly in PITEX model
- See report: RIACS TR 03.03

LPF on PITEX

- By Tony Lindsey (QSS / NASA ARC)
  - Supported by ECS project
- Two scenarios considered:
  - Random: auto-generated scenario (10K states)
  - PITEX: combining PITEX test scenarios (90 states)
- Explores 50-100 states / min
  - Too long for live demonstration
- First rounds (early 2002, early 2003)
  - Found errors in LPF and Livingstone (checkpointing)

LPF on PITEX (cont'd)

- Types of diagnosis properties verified
  - "some diagnosis matches the true faults":
    reports many errors, mostly spurious/benign (hidden faults).
  - "some diagnosis subsumes the true faults":
    only 5 errors with Random scenario (10K states),
    considered useful by PITEX modelers at ARC.
  - Further refinements will likely need domain knowledge: when is a fault relevant/critical?