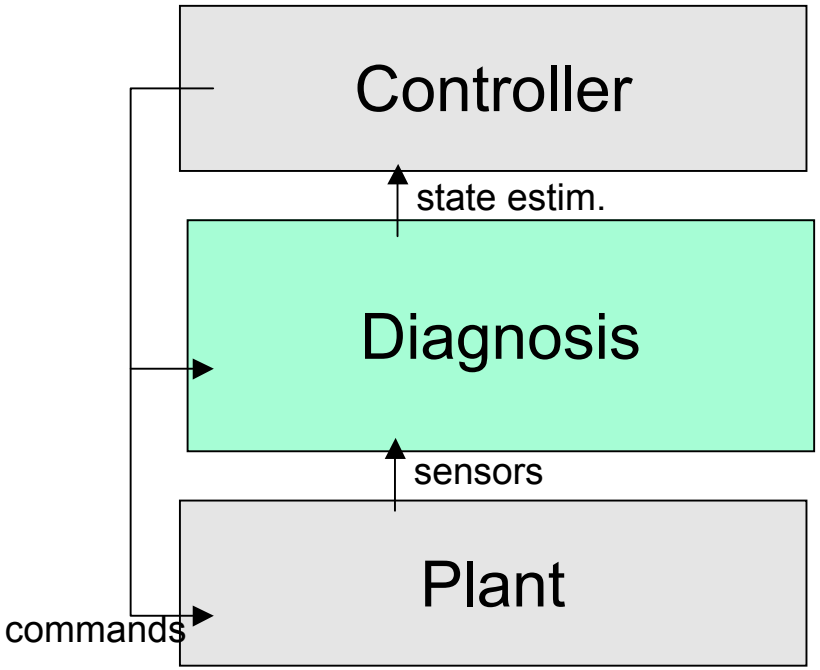
Ames Research Center

Simulation-Based Verification of Autonomous Controllers via Livingstone PathFinder

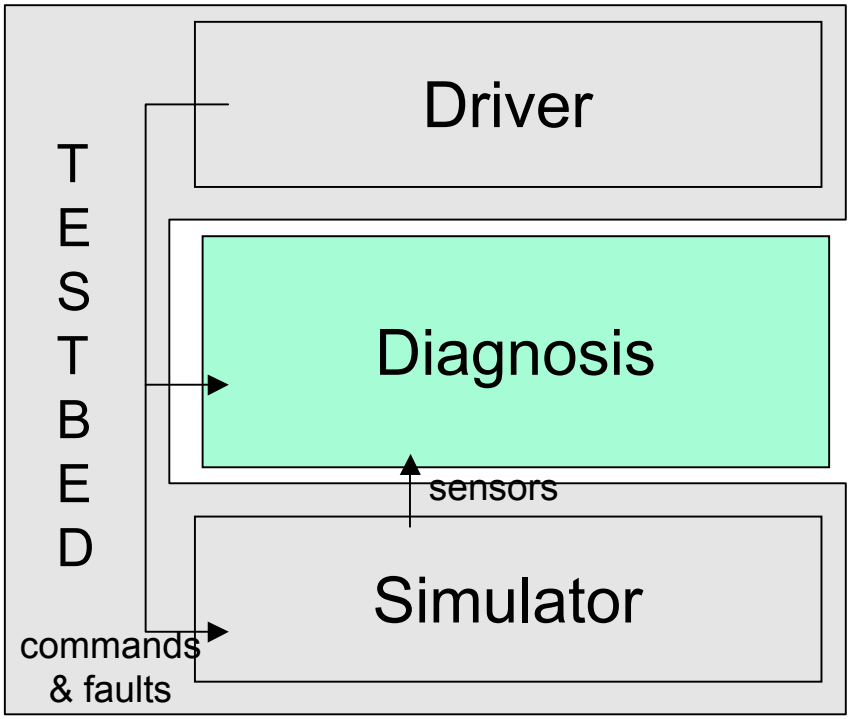
Tony Lindsey (QSS Group, NASA Ames)

Charles Pecheur (RIACS, NASA Ames)

Diagnosis

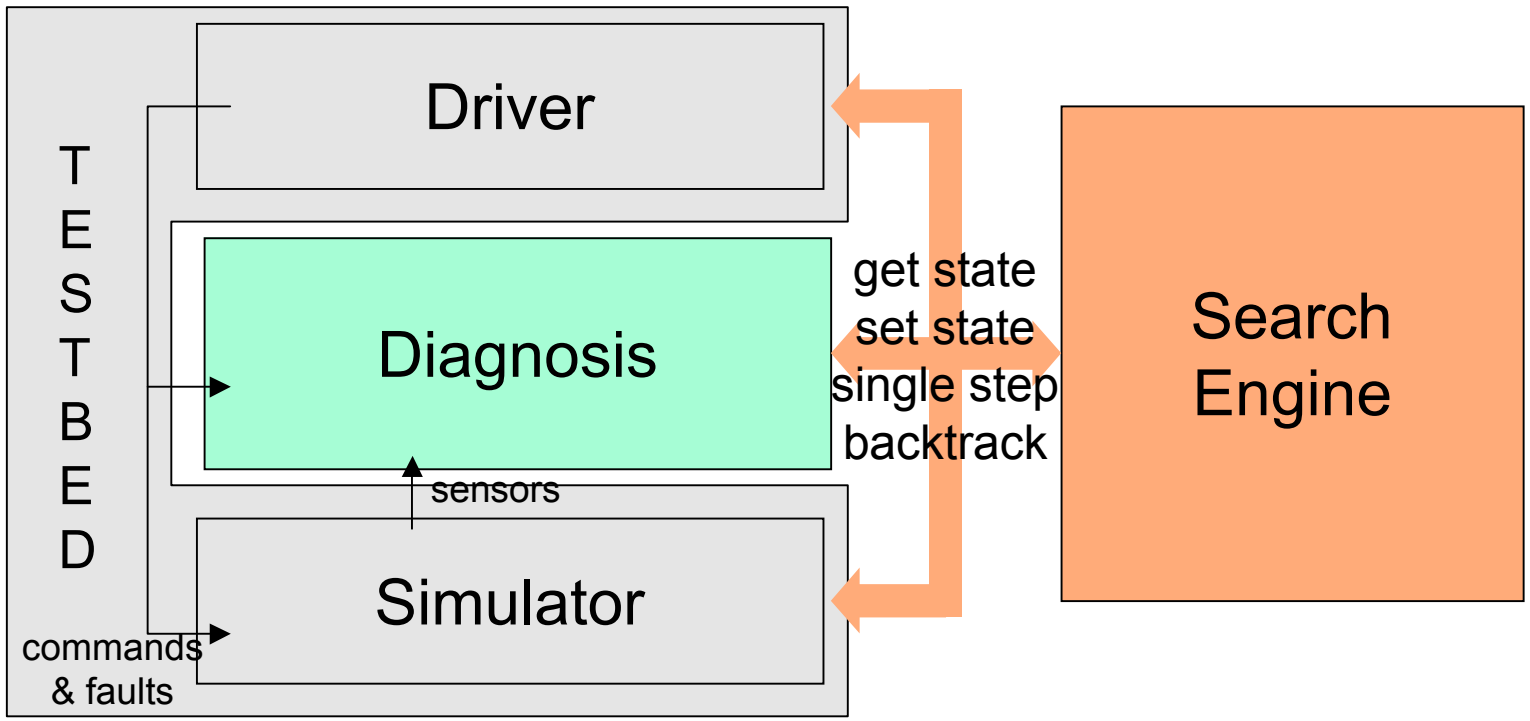


Diagnosis + Testbed



Diagnosis + Testbed + Search

Ames Research Center

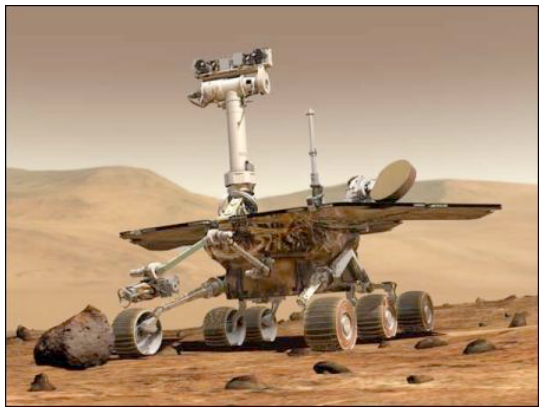


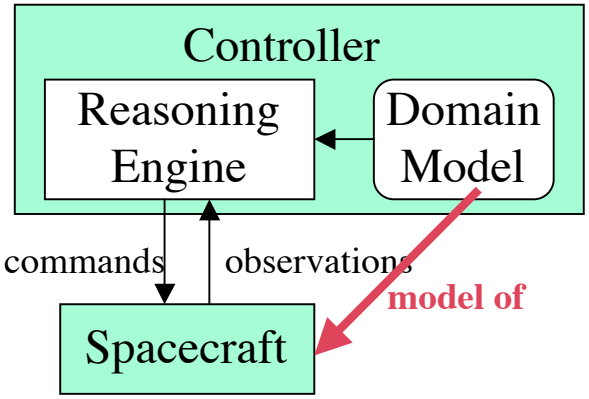
Autonomy at NASA

Ames Research Center

Autonomous spacecraft = on-board intelligence (AI)

- **Goal:** Unattended operation in an unpredictable environment
- **Approach:** model-based reasoning
- **Pros**: smaller mission control crews, no communication delays/blackouts
- **Cons:** **Verification and Validation ???** Much more complex, huge state space
- Better verification is critical for adoption

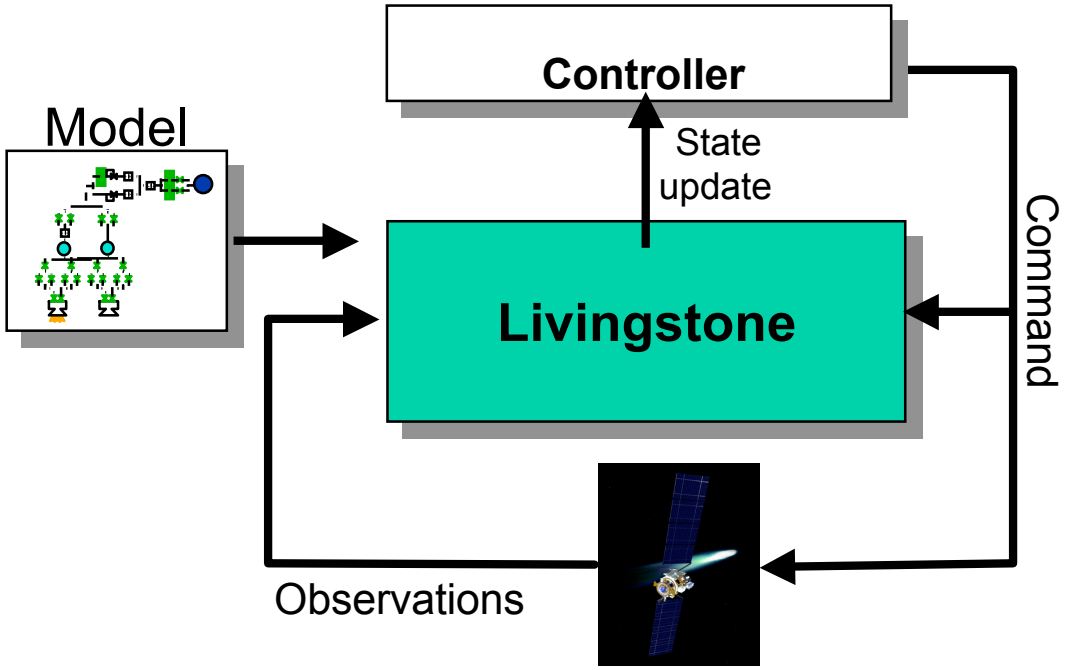




Model-Based Diagnosis

Ames Research Center

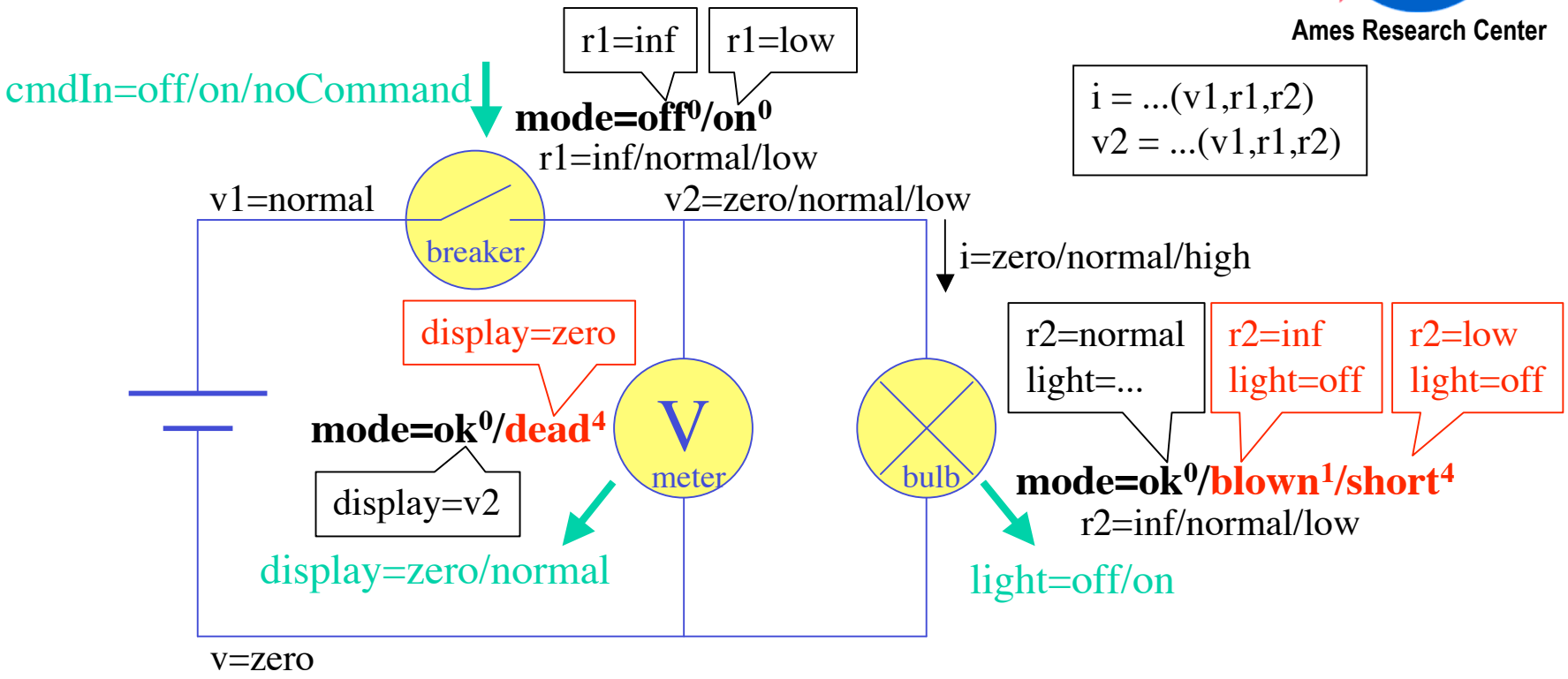
- Focus on **Livingstone** system from NASA Ames.
- Uses a discrete, qualitative model to reason about faults



Courtesy Autonomous Systems Group, NASA Ames

A Simple Diagnosis Model

Ames Research Center



Goal: determine **modes** from observations
Generates and tracks *candidates*

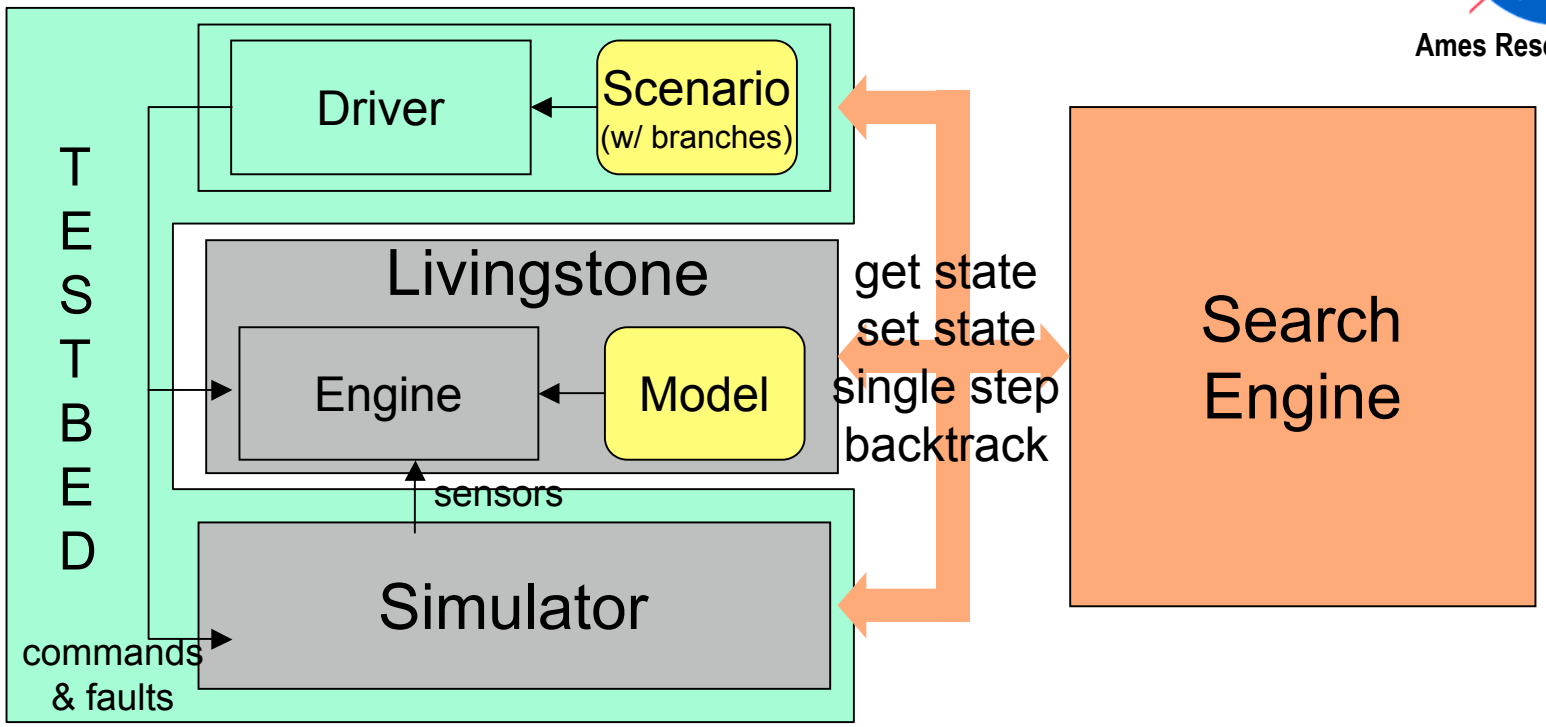
breaker	bulb	meter	rank
off⁰	**ok⁰**	**ok⁰**	**0**
off⁰	**ok⁰**	**blown¹**	**1**
on⁰	**dead⁴**	**short⁴**	**8**

Faults vs. Errors

Ames Research Center

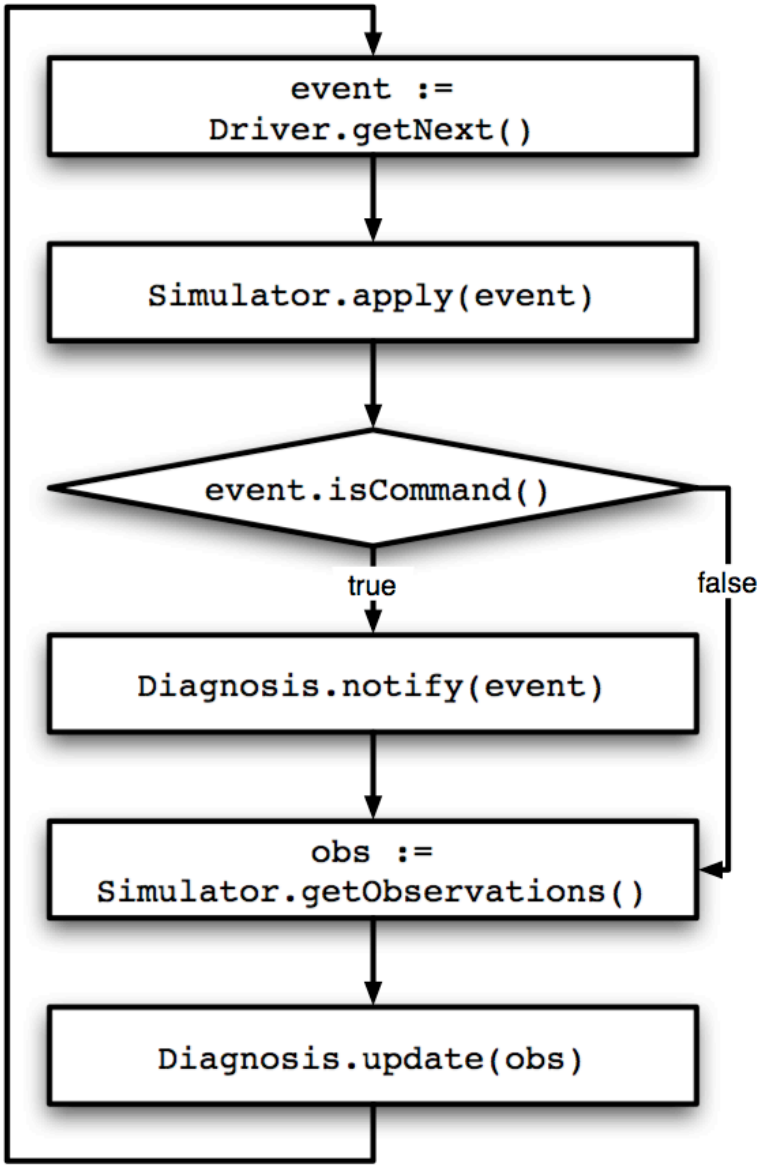
Faults	Errors
Ex: valve is stuck	Ex: fault not detected
in Plant/Simulator	in Diagnosis/Design
Spontaneous physical event	Human design flaw
To de detected by Diagnosis	**To be detected by Verification**

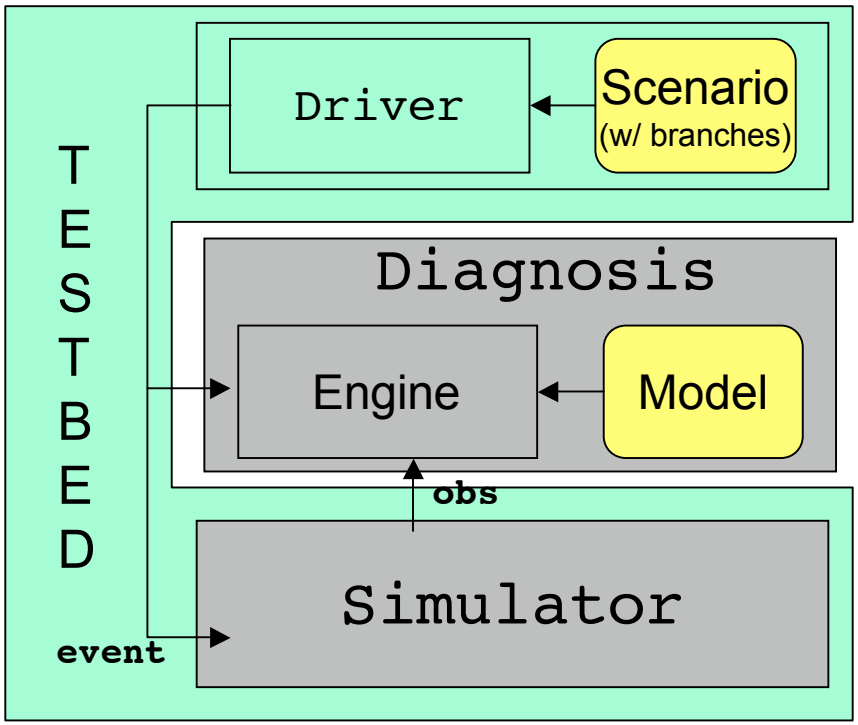
Livingstone PathFinder (LPF)



- Similar to VeriSoft[Godefroid 97]
- Uses checkpointing implemented in Livingstone
- In Java, accesses Livingstone (C++) through JNI

One Diagnosis Step



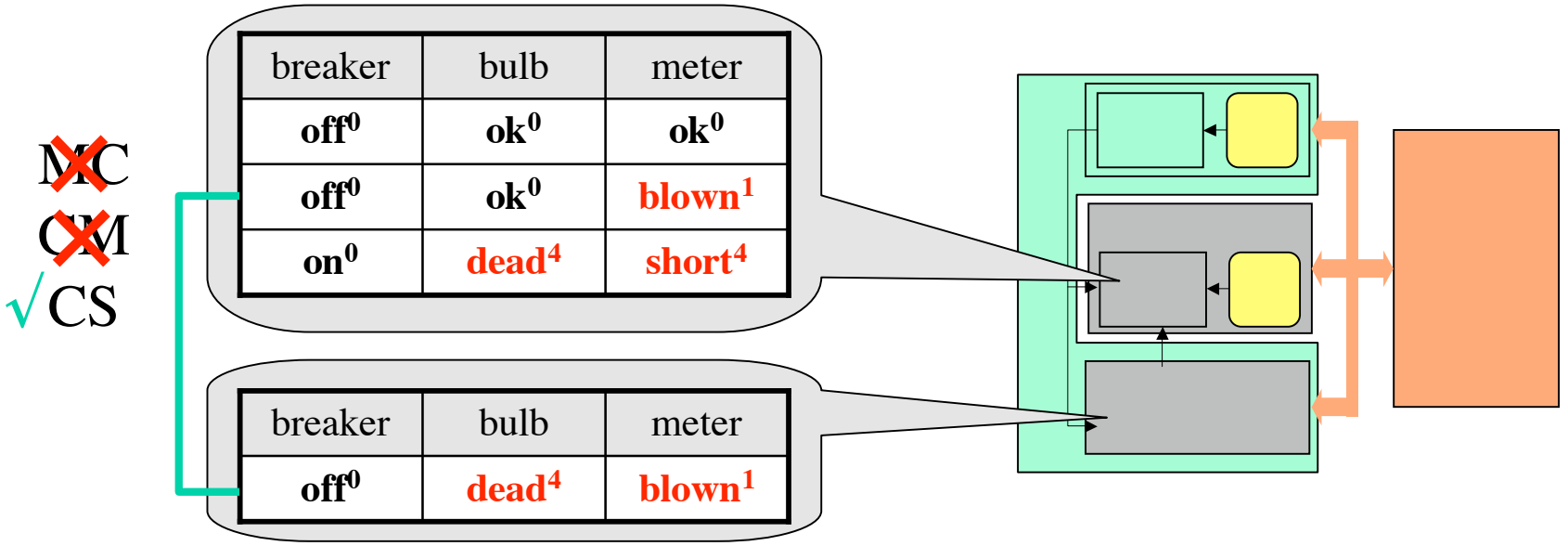


Ames Research Center

LPF Error Conditions

Ames Research Center

- Diagnosis candidates are "correct" w.r.t. Simulator modes
  - Mode Comparison (MC): first candidate is correct
  - Candidate Matching (CM): some candidate is correct
  - Candidate Subsumption (CS): some candidate's faults are included
- CS may miss errors but works best in practice

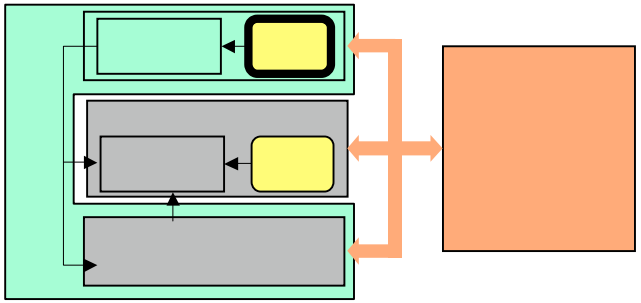


LPF Simulation Scenarios

Ames Research Center

- Defines the tree of executions to be explored
- Described as a non-deterministic program using a simple scripting language

```
stmt  ::= " event "  ;          single event
        | { stmt* }             sequence
        | mix stmt (and stmt)*  interleaving
        | choose stmt (or stmt)*  choice
```

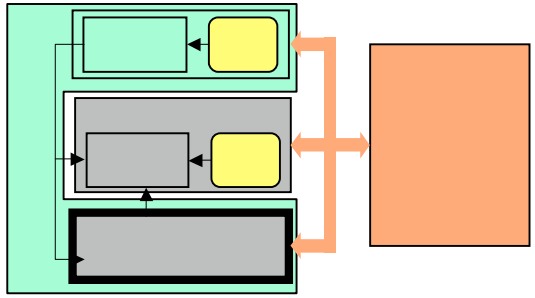


- Implemented as a hierarchy of automata objects matching the scenario script structure

LPF Simulators

Ames Research Center

- Framework allows to use any (suitably instrumented) simulation software
  - Trade-off: higher-fidelity simulators may restrict instrumentation
- Current implementation uses second Livingstone engine as simulator
  - Same or different model
  - Different mode of operation:
    Diagnosis : cmds, obs –> modes
    Simulator : cmds, modes –> obs
  - Simulator comes "for free"
  - Rationale: verify diagnosis assuming the model is correct

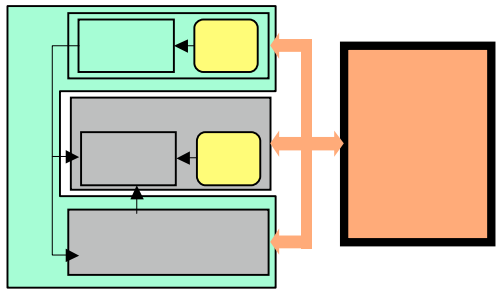


- Also considered: CONFIG (hybrid, NASA JSC)

LPF Search

Ames Research Center

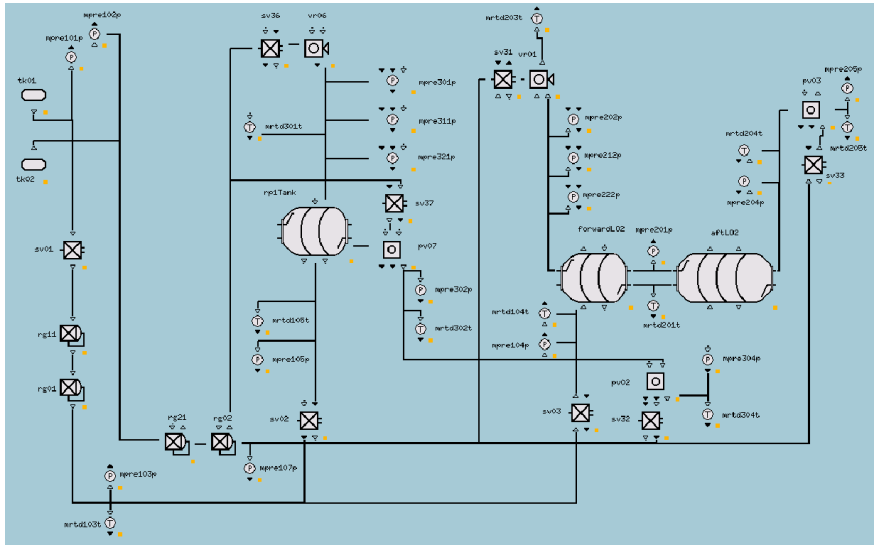
- The whole testbed is seen as a transition system
- API to enumerate transitions, backtrack, get/set state
  - Shared with Java PathFinder[Visser et al. 00]
  - Principle inspired from OPEN/CAESAR[Garavel 98]
- Search engine fixes exploration strategy
  - Depth-First: simplest, most efficient
  - Heuristic: valuation function on states (e.g. number of diagnosis candidates)
  - Breadth-First
  - Others are possible (random, pattern-based, interactive)



Application: PITEX

Ames Research Center

- Propulsion feed system of space vehicle
- Livingstone model: 2300 lines, 823 vars, $\approx 10^{33}$ states (SMV)
- Two scenarios:
  - Random Scenario (10216 states):
    sequence of commands || choice of faults
  - PITEX Scenario (89 states):
    combines 29 test cases used by application team



LPF on PITEX: Results

Ames Research Center

scenario	strategy	search	condition	errors	non-trivial	states	states/min
baseline	DFS	all	CM	27	4	89	44
baseline	DFS	all	CS	0	0	89	67
random	DFS	all	CM	9621	137	10216	51
random	DFS	all	CS	5	5	10216	52

scenario	strategy	search	condition	max. depth	states	states/min
random	DFS	one	CS	16	8648	49
random	BFS	one	CS	3	154	38
random	CC	one	CS	5	154	38

DFS=depth-first, BFS=breadth-first, CC=candidate-count
all=all errors, one=first error, min=shortest trace
CM=candidate matching, CS=candidate subsumption

trivial error=no fault reported

Perspectives

Ames Research Center

- Extend search options
  - More heuristics (including application-specific)
  - New search strategies (randomized, coverage-based)
- Improve usability
  - GUI, post-process and display results
- Generalize to reactive control
  - From fault detection to fault recovery
  - In progress: adapt LPF to Titan (MIT)
- Other approach: apply SMV (and BMC) to Livingstone models, verify diagnosability[Cimatti et al. 03]
  - using Livingstone-to-SMV translator[Pecheur et al. 00]

Extra Slides

Ames Research Center

Verification of Diagnosis systems

Ames Research Center

Verify what?

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

Here we look at 3 only!

PITEX Scenarios

Ames Research Center

```
mix {
  "command test.sv02.valveCmdIn=close";
  "command test.sv02.valveCmdIn=open";
  ...
} and {
  choose
    "fault test.forwardLO2.mode=unknownFault"; or
    "fault test.mpre101p.mode=faulty"; or
    ...
}
```

```
{
  choose { "fault test.mpre202p.mode=biased"; }
  or { "fault test.mpre212p.mode=biased"; }
  or {
    "command test.sv31.valveCmdIn=open";
    choose {
      "fault test.sv31.sv.mode=stuckOpen";
      "command test.sv31.valveCmdIn=close";
    } or {
      "command test.sv31.valveCmdIn=close";
       ...
} } }
```