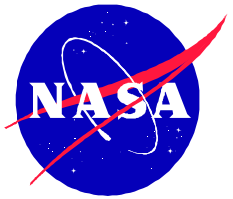
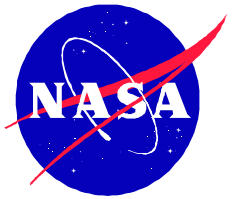


Verification of Intelligent Software

Charles Pecheur (RIACS / NASA Ames)

©Charles Pecheur, RIACS / NASA Ames

Contents



Model Checking for Intelligent Software

- Why?

  Intelligent software, how to verify it?

- What?

  A bird's-eye view of model checking

- How?

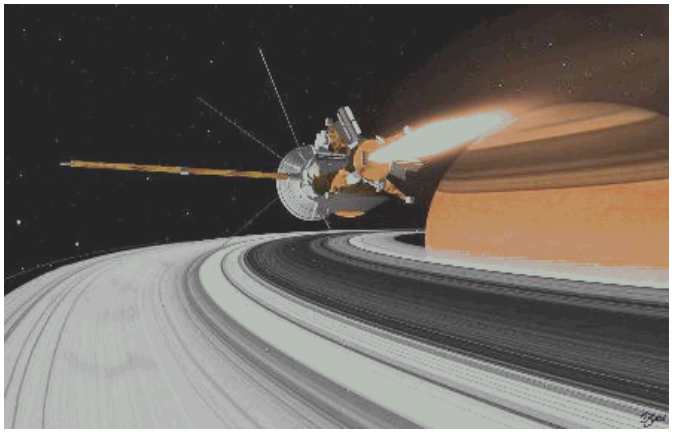
  Experiences in the ASE Group

Autonomous Systems

"Faster, better, cheaper" spacecrafts

=> add on-board intelligence

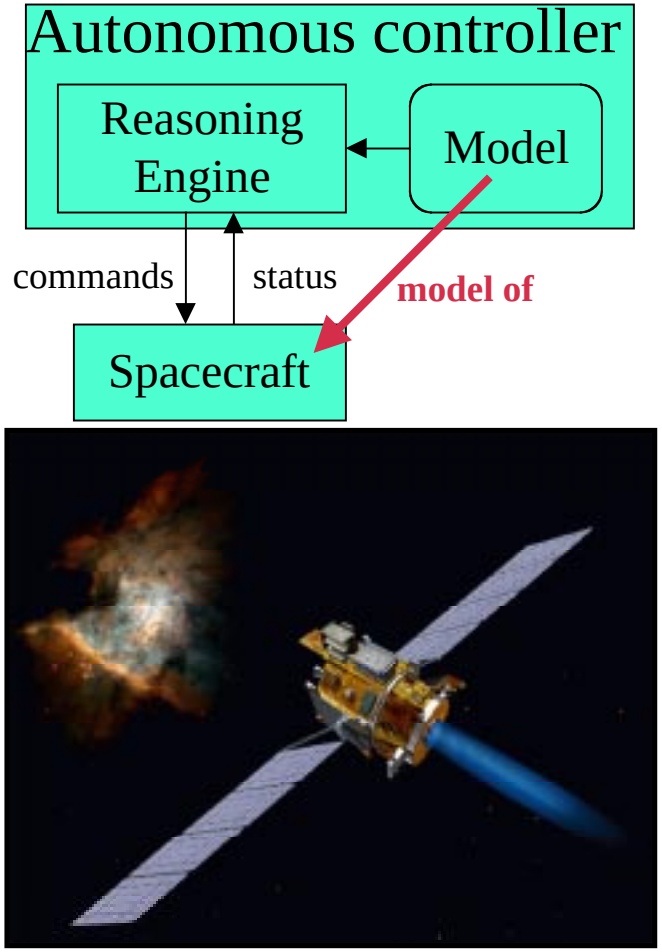
- From self-diagnosis
  to on-board science.
- Smaller mission control crews
  => reduced cost
- Less reliance on control link
  => OK for deep space



©Charles Pecheur, RIACS / NASA Ames

Model-Based Autonomy

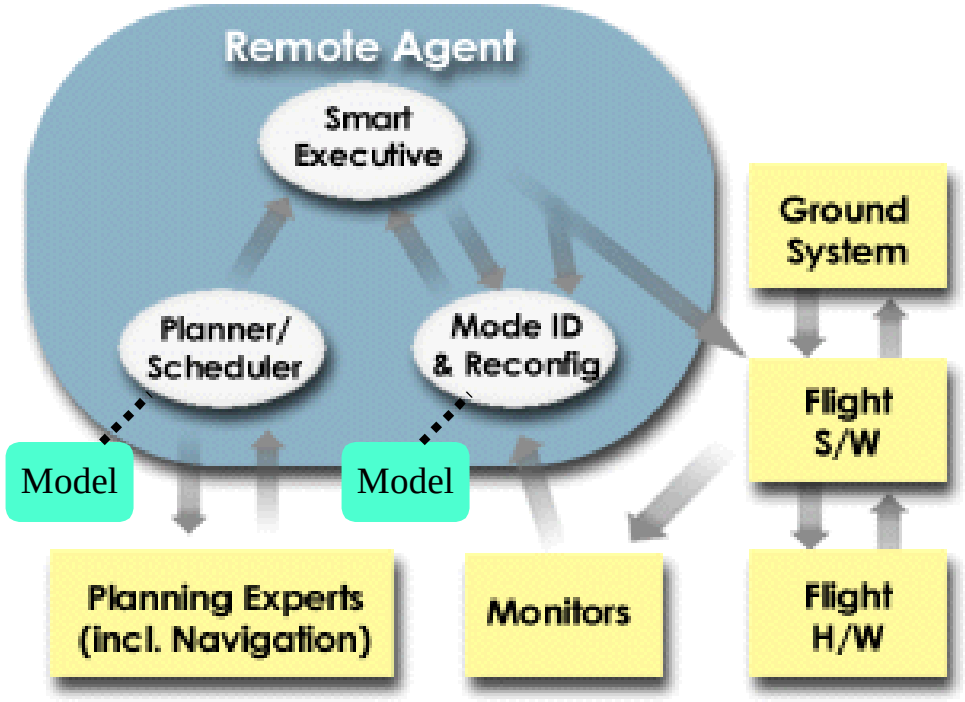
- Based on AI technology
- General reasoning engine + application-specific model
- Use model to respond to unanticipated situations



©Charles Pecheur, RIACS / NASA Ames

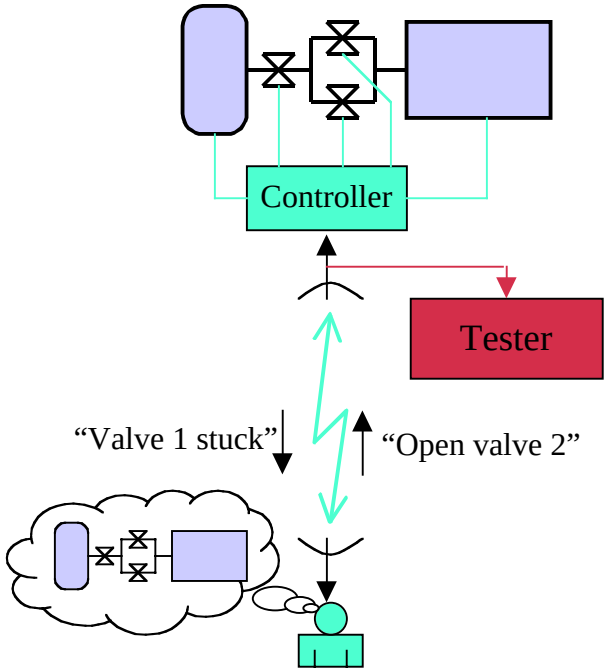
Example: Remote Agent

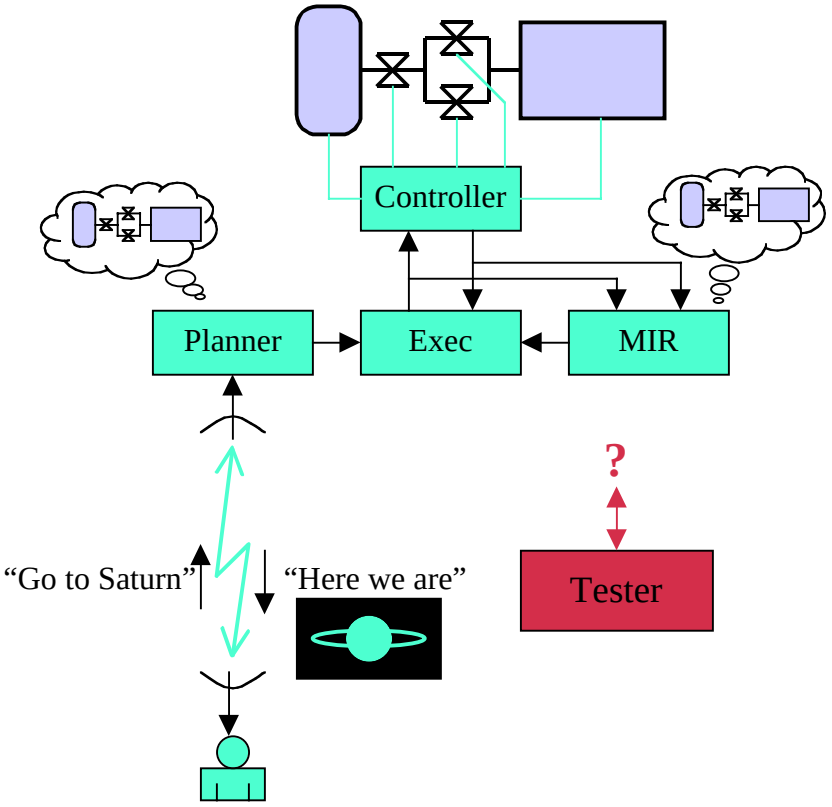
- From Ames ARA Group (+ JPL)
- On Deep Space One in May 1999 (1st AI in space!)



©Charles Pecheur, RIACS / NASA Ames

Controlled vs. Autonomous





©Charles Pecheur, RIACS / NASA Ames

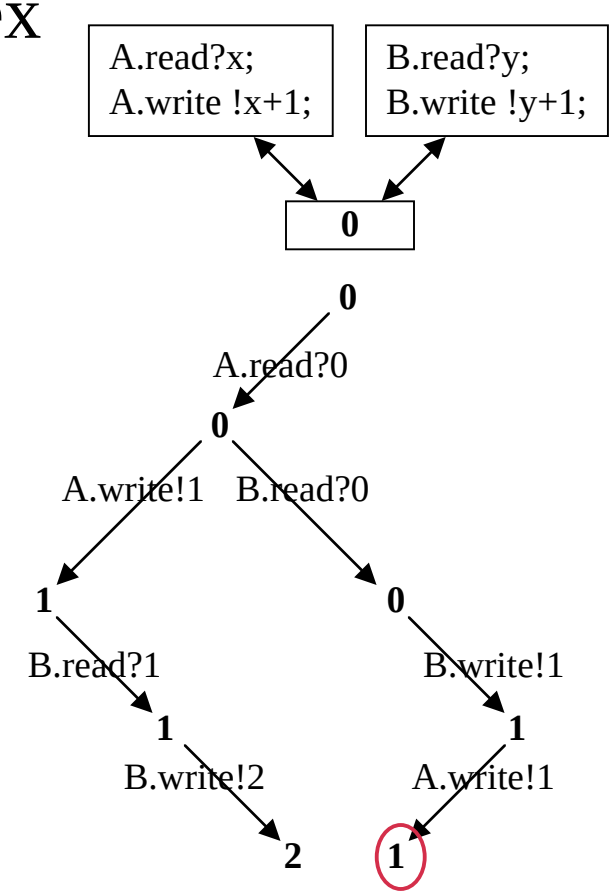
Testing intelligent software?

- Programs are much more complex
- Many more scenarios

  => testing gives low coverage
- Concurrency!

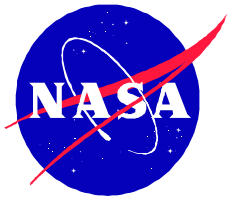
  Due to scheduling,
  the same inputs (test) can give
  different outputs (results)

  => test results are not reliable



©Charles Pecheur, RIACS / NASA Ames

Contents



Model Checking for Intelligent Software

- Why?

  Intelligent software, how to verify it?

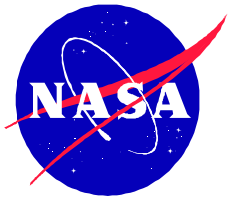
- What?

  A bird's-eye view of model checking

- How?

  Experiences in the ASE Group

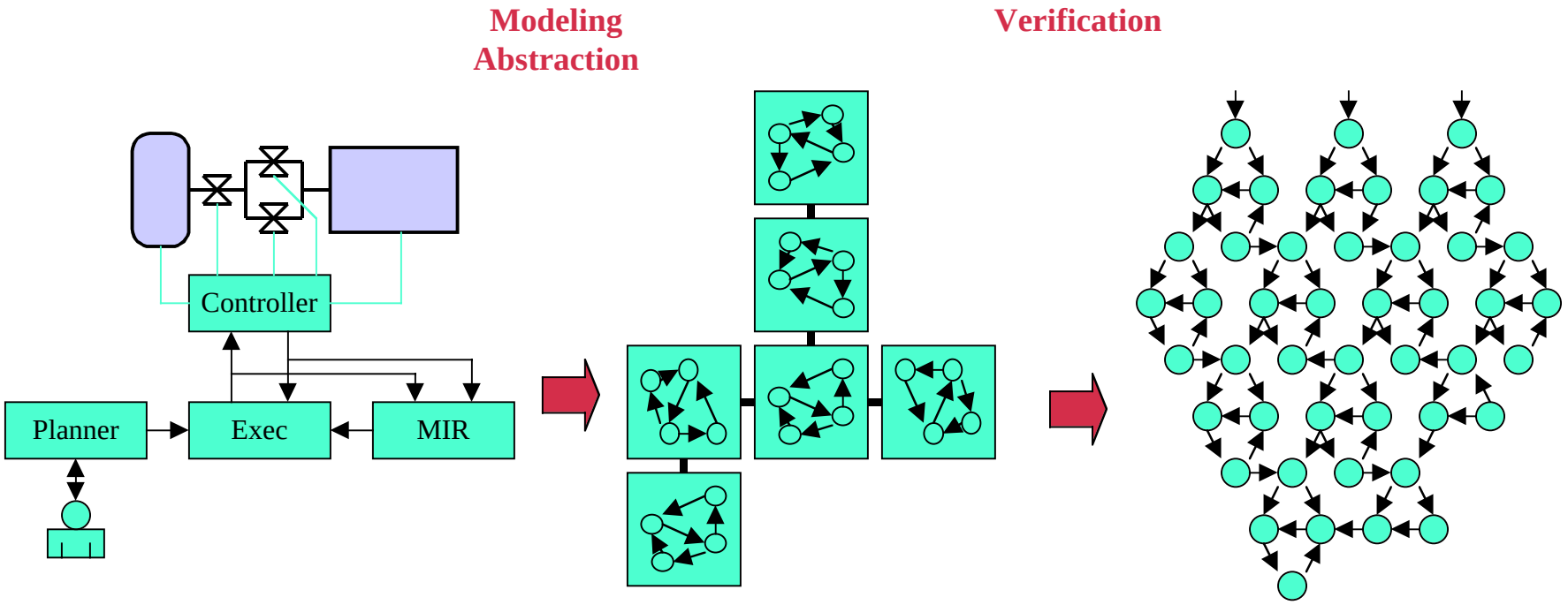
Model Checking



Check whether a system S satisfies a property P
by exhaustive exploration of all executions of S

- Controls scheduling => better coverage
- Can be done at early stage => less costly
- Widely used in hardware, coming in software
- Examples: Spin (Bell Labs), Murphi (Stanford)

Model ...

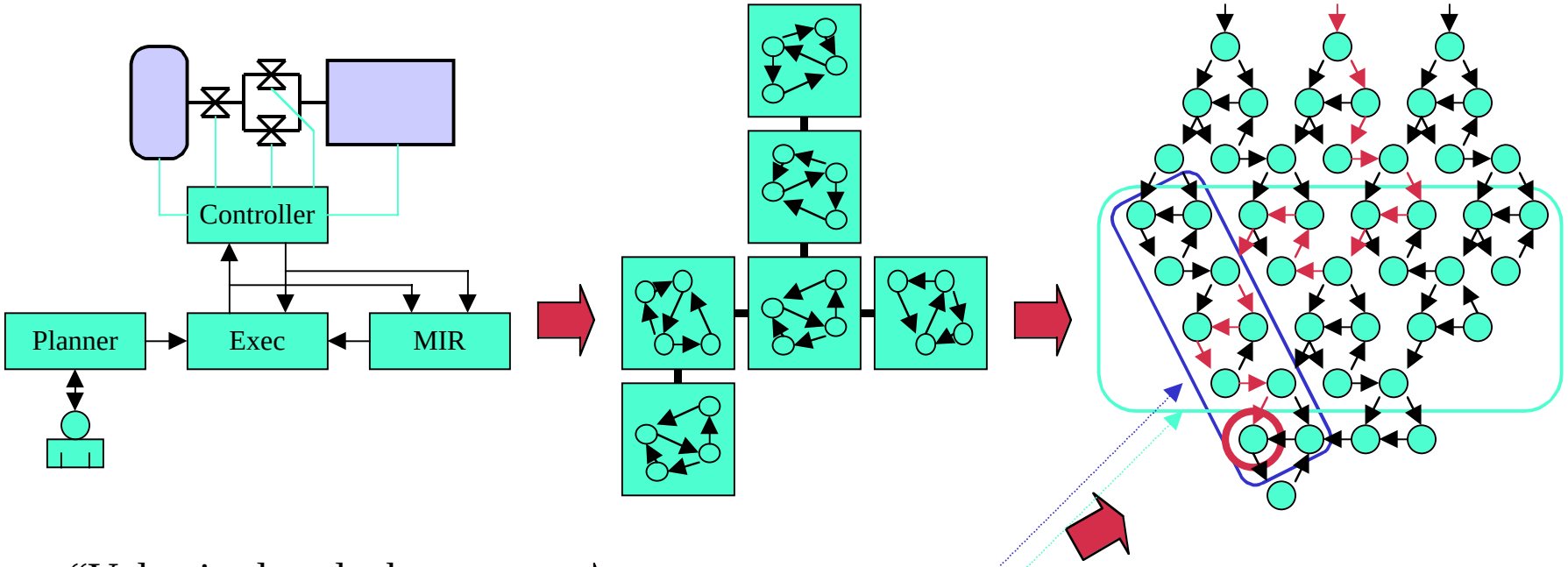


©Charles Pecheur, RIACS / NASA Ames

Model Checking

Modeling Abstraction

Verification



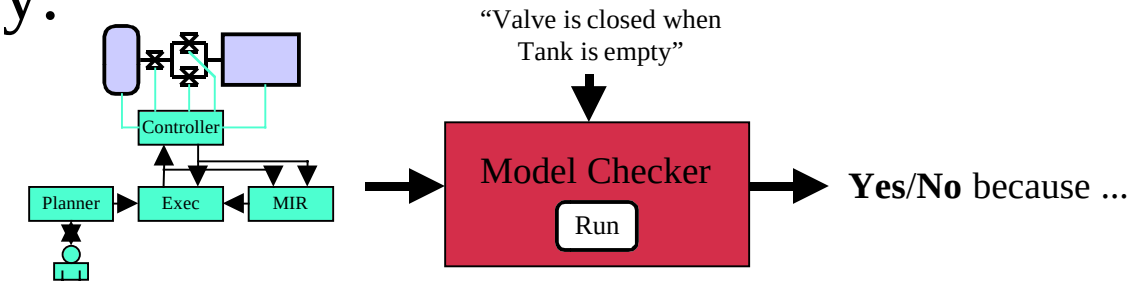
"Valve is closed when Tank is empty"

AG (tank=empty => valve=closed)

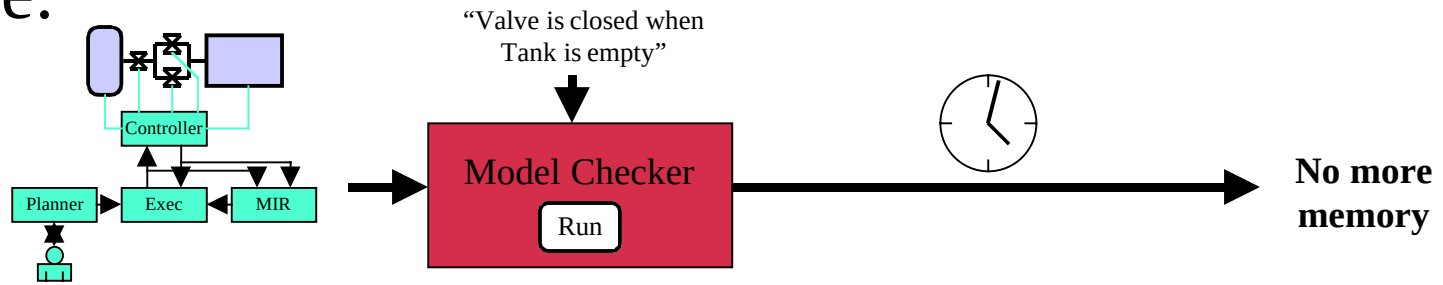
State Space Explosion

K processes with N local states ≤ N^K global states

Theory:

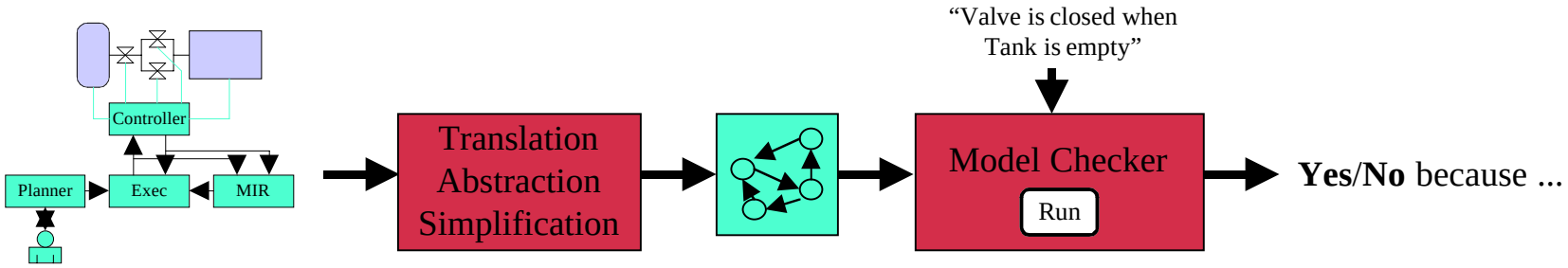


Practice:



©Charles Pecheur, RIACS / NASA Ames

Modeling



This is the tough job!

- **Translation**: to model checker's syntax

  e.g. C —> Promela (Spin)

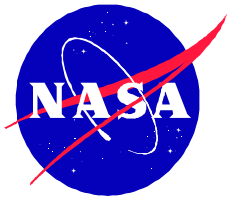
- **Abstraction**: ignore irrelevant parts

  e.g. contents of messages

- **Simplification**: downsize relevant parts

  e.g. number of processes, size of buffers

Temporal Logic



- Propositional logic + quantifiers over executions
- Example: "every request gets a response"

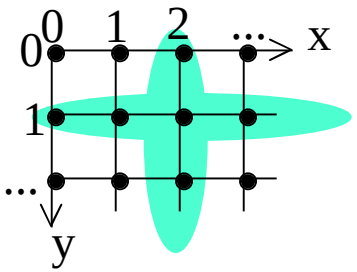
  AG (Req => **AF** Resp)

  Always Globally, if Req then **Always Finally** Resp
- Branching (CTL) vs. linear (LTL)
  - different verification techniques
  - neither is more general than the other
- Model checking without TL
  - Assertions, invariants
  - Compare systems, observers

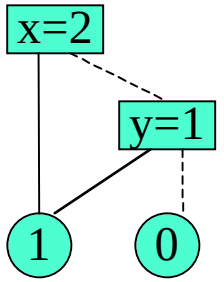
©Charles Pecheur, RIACS / NASA Ames

Symbolic Model Checking

- Manipulates sets of states,
  Represented as boolean formulas,
  Encoded as binary decision diagrams.
- Can handle larger state spaces (10^{50} and up).
- BDD computations:
  - Good in average but exponential in worst case.
  - Computation time depends on BDD size
    => number of variables, complexity of formulas,
    but not directly state space size.
- Example: SMV (Carnegie Mellon U.)



$x=2 \vee y=1$



©Charles Pecheur, RIACS / NASA Ames

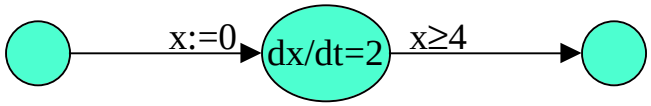
Real-Time and Hybrid

- "Classic" model checking: finite state, un-timed
- Real-time model checking: add clocks

  e.g. Khronos (Verimag), Uppaal (Uppsala/Aalborg)

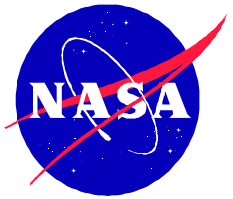
- Hybrid model checking: add derivatives

  e.g. Hytech (Berkeley)



More complex problems & less mature tools

Contents



Model Checking for intelligent software

- Why?

  intelligent software, how to verify it?

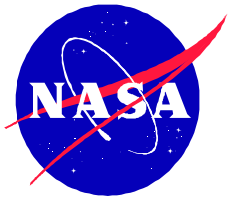
- What?

  A bird's-eye view of model checking

- How?

  Experiences in the ASE Group

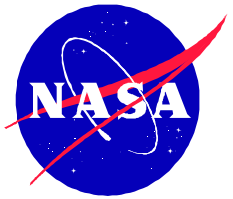
Verification of Remote Agent Executive



(Lowry, Havelund and Penix)

- Smart executive system with AI features (Lisp)
- Modeled (1.5 month) and
  Model-checked with Spin (less than a week)
- 5 concurrency bugs found, that would have been hard to find through traditional testing

Hunting the RAX Bug

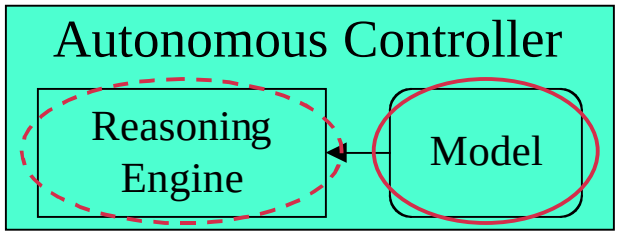


(Lowry, White, Havelund, Pecheur, ...)

- 18 May 1999: Remote Agent Experiment suspended following a deadlock in RA EXEC
  => Q: could V&V have found it?
- Over-the-week-end "clean room" experiment
- => A: V&V found it... two years ago!

  Similar to one of the 5 bugs found before (elsewhere)
  - Highly unlikely to occur
  - Never occurred during thorough testing
  - Occurred in flight!
- Morale: Testing not enough for concurrency bugs!

Verification of Model-Based Autonomy



Reasoning Engine

- Relatively small, generic algorithm => use prover
- Requires V&V expert level but once and for all
- At application level, assume correctness (cf. compiler)

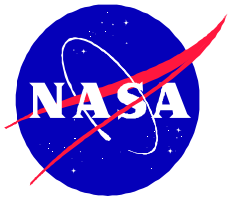
Model

- Complex assembly of interacting components => model checking
- Avoid V&V experts => automated translation
  Not too hard because models are abstract

Reasoning Engine + Model ???

©Charles Pecheur, RIACS / NASA Ames

Verification of Planner/Scheduler Models



(Penix, Pecheur and Havelund)

- Model-based planner from Remote Agent

  Models: constraint style, real-time

- Small sample model translated by hand

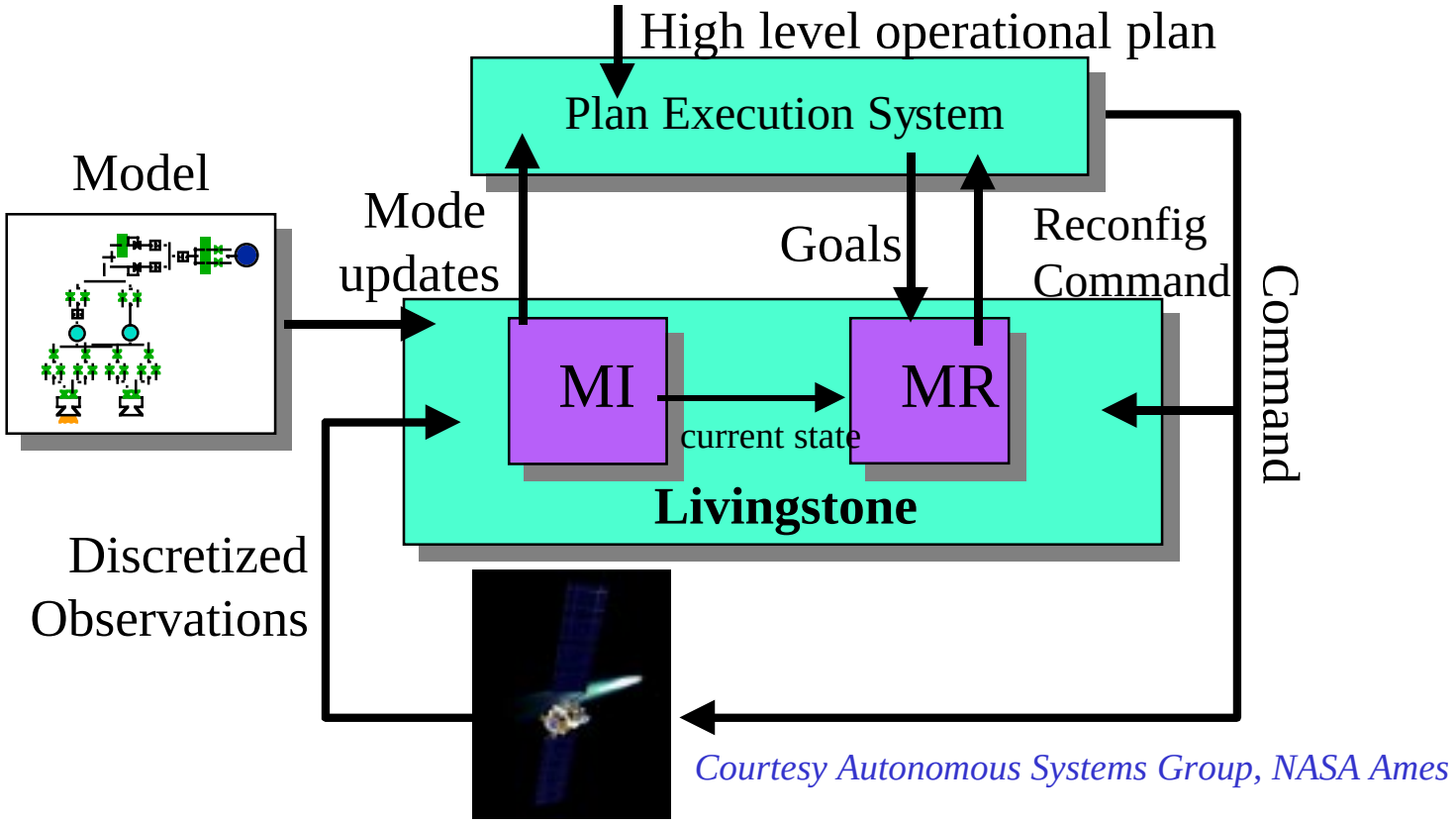
  Subset of the full modeling language, untimed

- Compare 3 model checkers: Spin, Murphi, SMV
  => SMV much easier and faster (≈0.05s vs. ≈30s)

- Continuation *(Khatib)*: handle timed properties using real-time model checker (Uppaal)

The Livingstone MIR

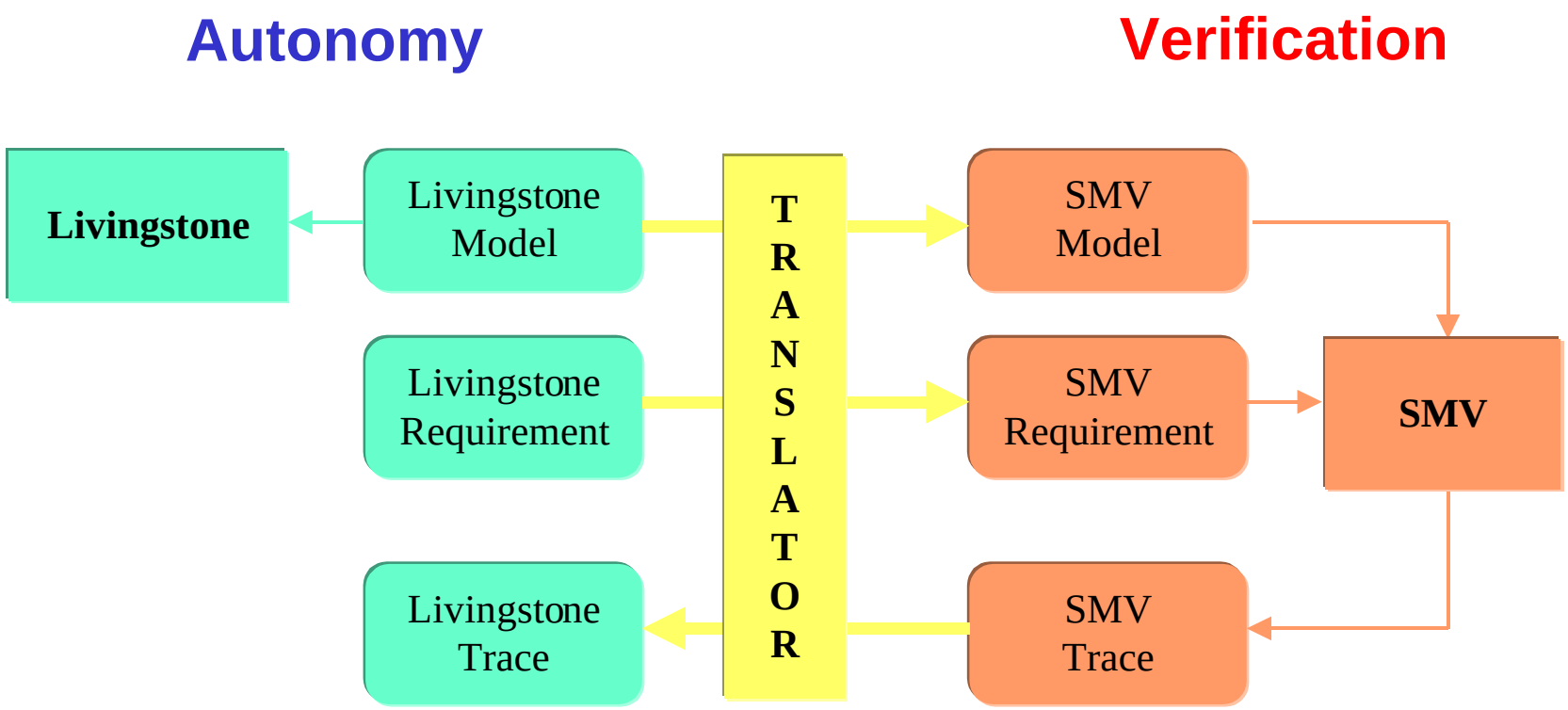
Remote Agent's model-based fault recovery sub-system



Courtesy Autonomous Systems Group, NASA Ames

©Charles Pecheur, RIACS / NASA Ames

Verification of Livingstone Models



©Charles Pecheur, RIACS / NASA Ames

Livingstone to SMV Translation

Information Sciences & Technology
Ames Research Center

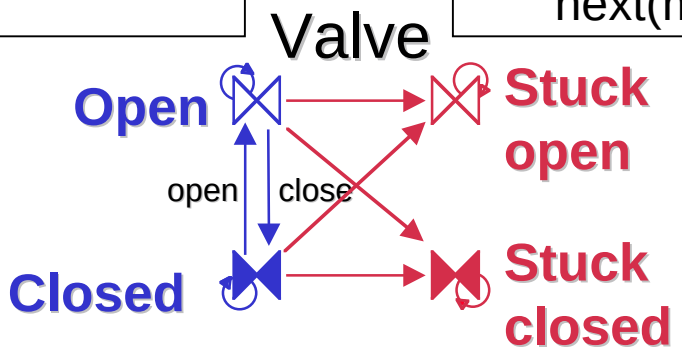
NASA

Livingstone Model

```
(defcomponent valve ()
  (:inputs (cmd :type valve-cmd))
 ...
  (Closed :type ok-mode
    :transitions
      ((do-open :when (open cmd)
        :next Open) ...))
  (StuckC :type :fault-mode ...)
  ...)
```

SMV Model

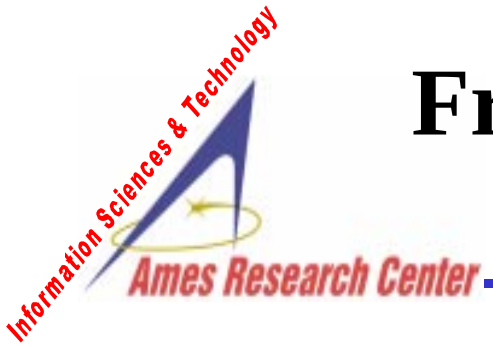
```
MODULE valve
VAR      mode: {Open,Closed,
                StuckO,StuckC};
         cmd: {open,close};
DEFINE faults:={StuckO,StuckC};
TRANS
  (mode=Closed & cmd=open) ->
    (next(mode)=Open |
      next(mode) in faults)
```



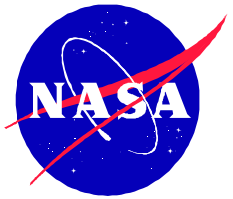
Livingstone
Autonomous Controller

SMV
Symbolic Model Checker

©Charles Pecheur, RIACS / NASA Ames



From Livingstone Models to SMV Models



(Simmons, Pecheur)

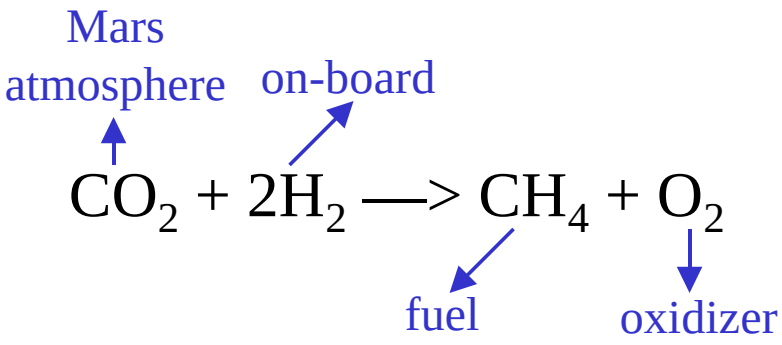
Translation program developed by CMU and Ames

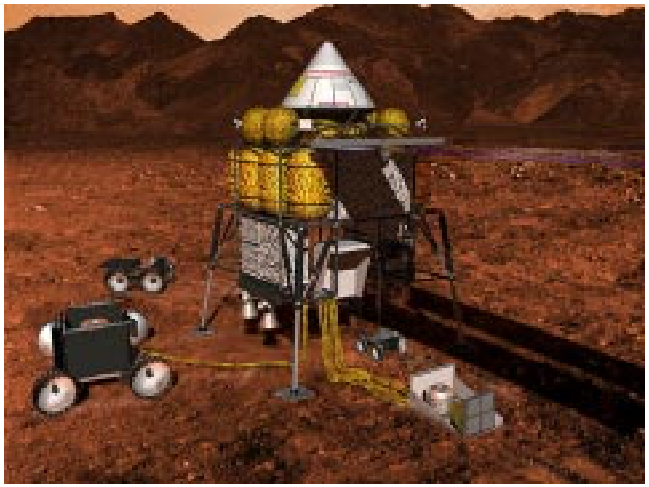
- 4K lines of Lisp
- Similar nature => translation is easy
- Properties in temporal logic + pre-defined patterns
- In progress:
  - more property patterns
  - translate results back to Livingstone

©Charles Pecheur, RIACS / NASA Ames

Application
In-Situ Propellant Production

- Use atmosphere from Mars to make fuel for return flight.
- Livingstone controller developed at NASA KSC.
- Components are tanks, reactors, valves, sensors...
- Exposed improper flow modeling.
- Latest model is 10^{50} states.





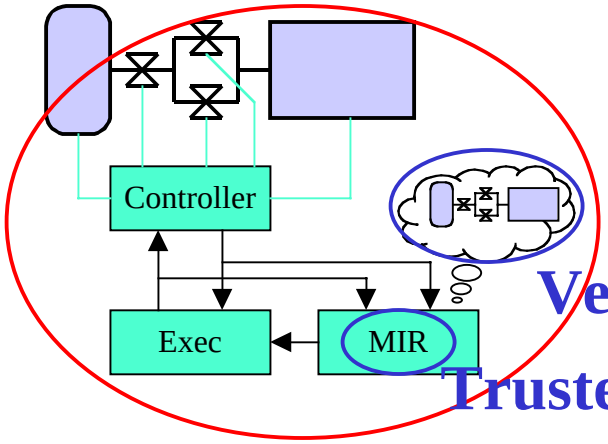
©Charles Pecheur, RIACS / NASA Ames

Beyond Model-Based Verification

- correct engine + correct model
  ≠> correct control !
  - heuristic search strategies
  - enough sensors/actuators?
  - model approximations
- Model check everything?
  Very hard!

  Need (abstract) V&V model of
  engine + model + spacecraft + ...

  => complex, error-prone, huge state space

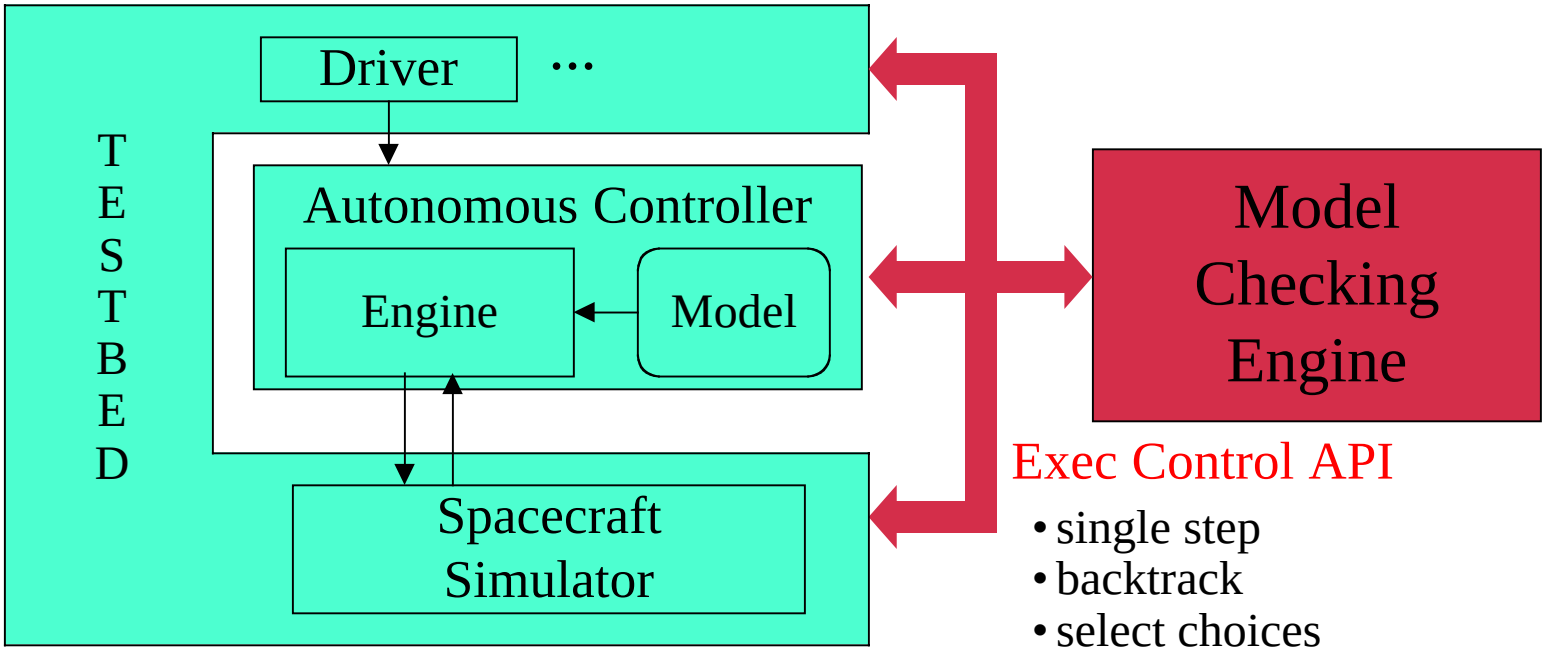


Verified

Trusted

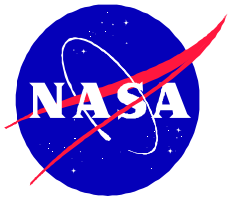
=> Reliable???

Closed-Loop Verification



- Real system => accuracy.
- More control => more coverage.
- For any discrete-event controller (not only model-based).

Model Checking Java
Java PathFinder

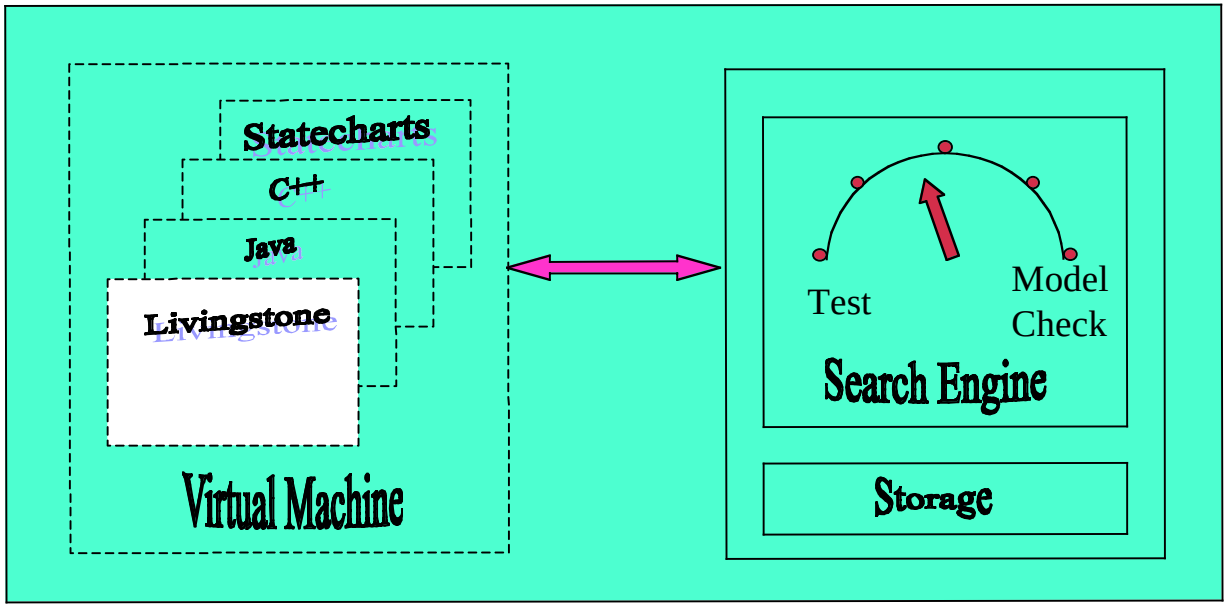


(Visser, Havelund)

- Java PathFinder 1
  - Translates from Java to Promela (Spin)

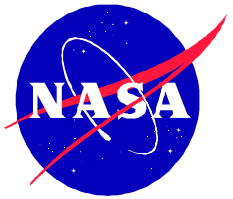
- Java PathFinder 2
  - Explicit-state model checking.
  - Works with bytecodes => handle all of Java.
  - Based on custom Java Virtual Machine
    - Written in Java (rapid prototyping).
    - Emphasis on memory management not speed.
  - Efficient encoding of states (heap, GC).

Generic Verification Environment



- Principle: uncouple V&V subject from V&V algo.
- Common denominator of several V&V projects.
- Current VMs: Java, Livingstone.

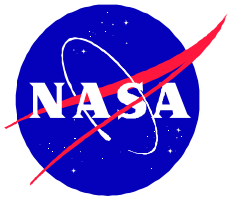
Conclusions



Model checking:

- Autonomy needs it – testing is not enough
- General pros&cons apply:
  - exhaustive... if model is small enough
  - automatic verification... but tough modeling
- Works nicely on autonomy models
- Solutions inbetween testing and model checking
- Not short of tough problems:
  - Real-time, hybrid, AI
  - Learning/adaptive systems: *after* training/*including* training

Pointers



- My home page

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

- JavaPathFinder

  http://ase.arc.nasa.gov/jpf

- Model-Based Verification of Intelligence

  AAAI Spring Symposium, Stanford, March 2001

  http://ase.arc.nasa.gov/mvi