Formal Verification of Autonomous Systems

Charles Pecheur

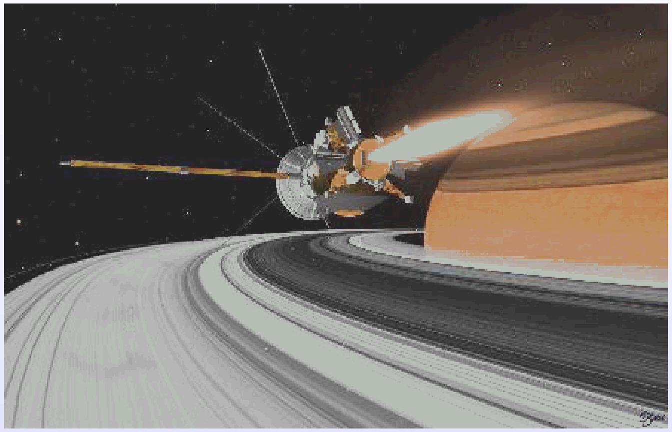
RIACS / ASE Group, NASA Ames

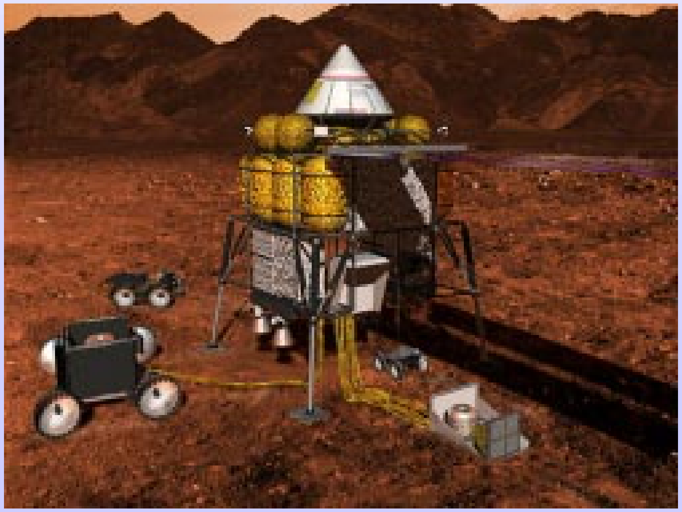
Autonomous Systems

Autonomous space explorers

"Faster, better, cheaper"

- Reduced human supervision
  => reduced cost
- Local reactions
  => no com delays/blackouts
- From self-diagnosis
  to on-board science.





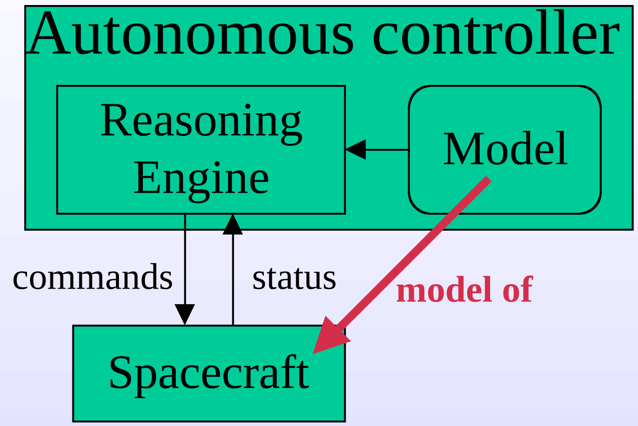
Model-Based Autonomy

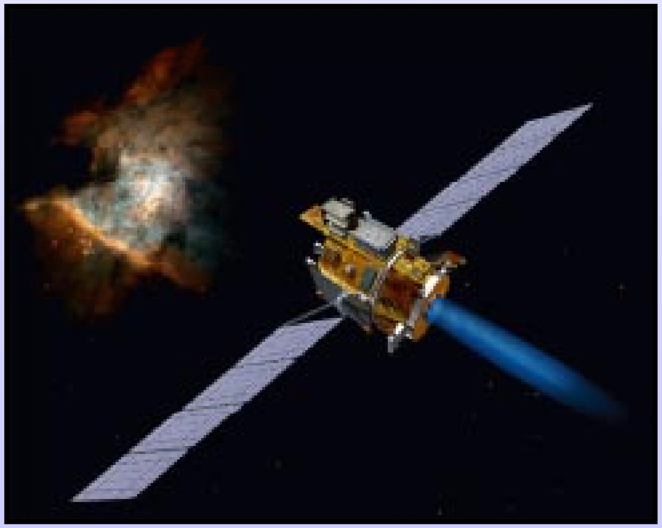
- Based on AI technology
- General reasoning engine + application-specific model
- Use model to respond to unanticipated situations
- Example: Remote Agent
  - Model-based planner/scheduler
  - AI-based executive
  - Model-based fault recovery

First run on Deep Space One:

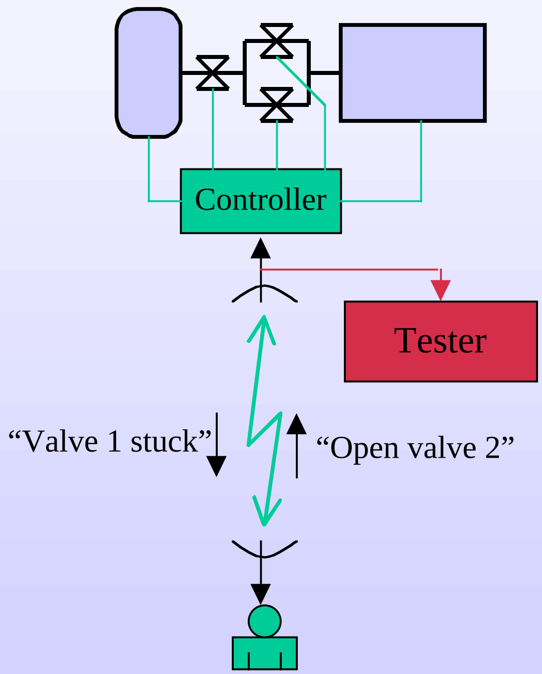
May 17, 1999

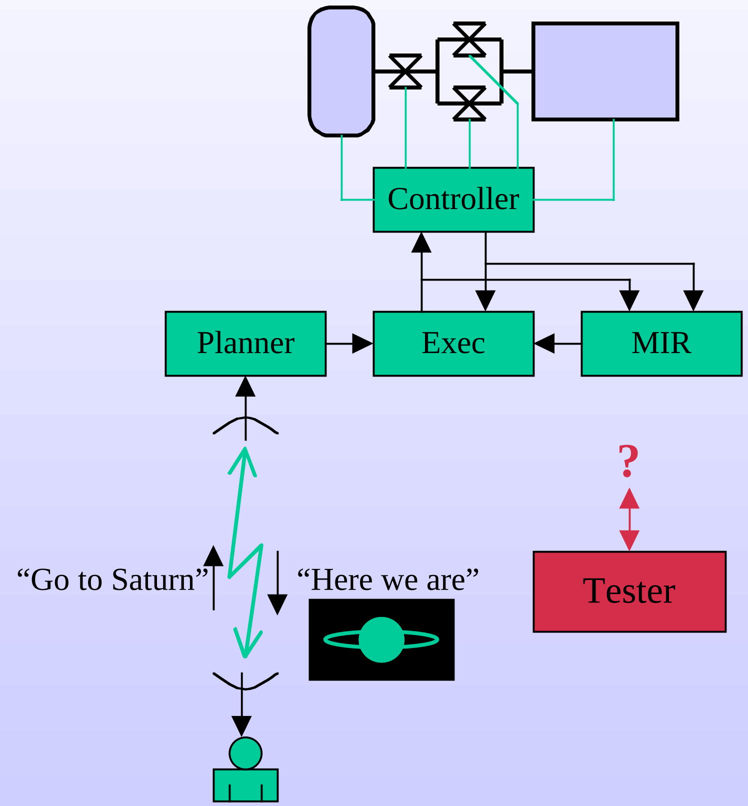
(1st A.I. program in space!)





Controlled vs. Autonomous





The Challenge

V&V of autonomous systems ?

- Critical for NASA to keep risk low.
- Huge state space and branching factor:
  - complex algorithms and data structures
  - internal decisions (no open control loop)
  - agent-based, knowledge-based, adaptive

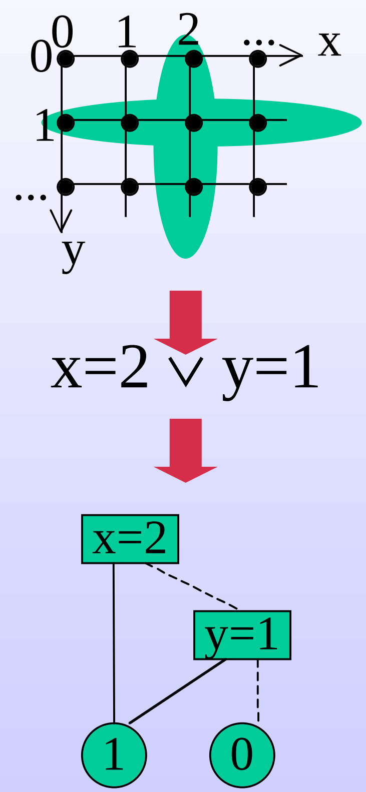
=> Conventional testing methods yield a very poor coverage.

Model Checking

- Checks whether S satisfies P, where:
  - S = model of the system, as a finite-state machine
  - P = property to verify, in temporal logic
- By exhaustive exploration
  - \+ Full coverage (incl. non-determinism)
  - – Limited by state space explosion
- At early stage => less costly
- Widely used in hardware, coming in software
- e.g. Spin (Bell Labs), Murphi (Stanford)

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.)



Verification of Remote Agent Executive

(Lowry, Havelund and Penix)

- Smart executive system with AI features
- Modeled (1.5 month) and
  Model-checked with Spin (less than a week)

  NB: costly modeling phase
  => need automated translation
- 5 concurrency bugs found, that would not have been found through traditional testing

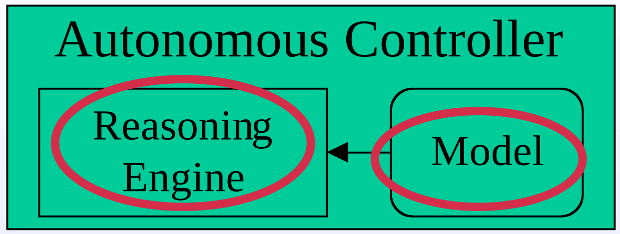
Hunting the RAX Bug

(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:
  - Front-end group selects suspect sections of the code
  - Back-end group does modeling (in Java) and verification (using Java Path Finder + Spin)
- => A: V&V found it... two years ago!

  Same as one of the 5 concurrency bugs found before
- 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 ???

The Planner/Scheduler

- High-level mission planning in DS-1, model-based.
- Produces a plan for achieving a given high-level goal (e.g. take snapshot of asteroid)
- Models = declarations of components (OO) + temporal constraints on values of variables

Example:

```
((Robot.Task=Fix) starts_before (10 20)
  (Hole.Status = Fixed))
```

Verification of Planner/Scheduler models

(Penix, Pecheur and Havelund)

- Compare 3 model checkers: Spin, Murphi, SMV
- Small sample model
- Translation by hand but systematic
  => can be automated
- General translation rules for a subset of the modeling language – Full language is for further study (non-local constraints, quantitative time)
- SMV gives easier translation and faster verification (≈0.05s vs. ≈30s for Spin or Murphi)

Planner/Scheduler Models (encore)

(Khatib) **anc. CS Faculty at FIT!**

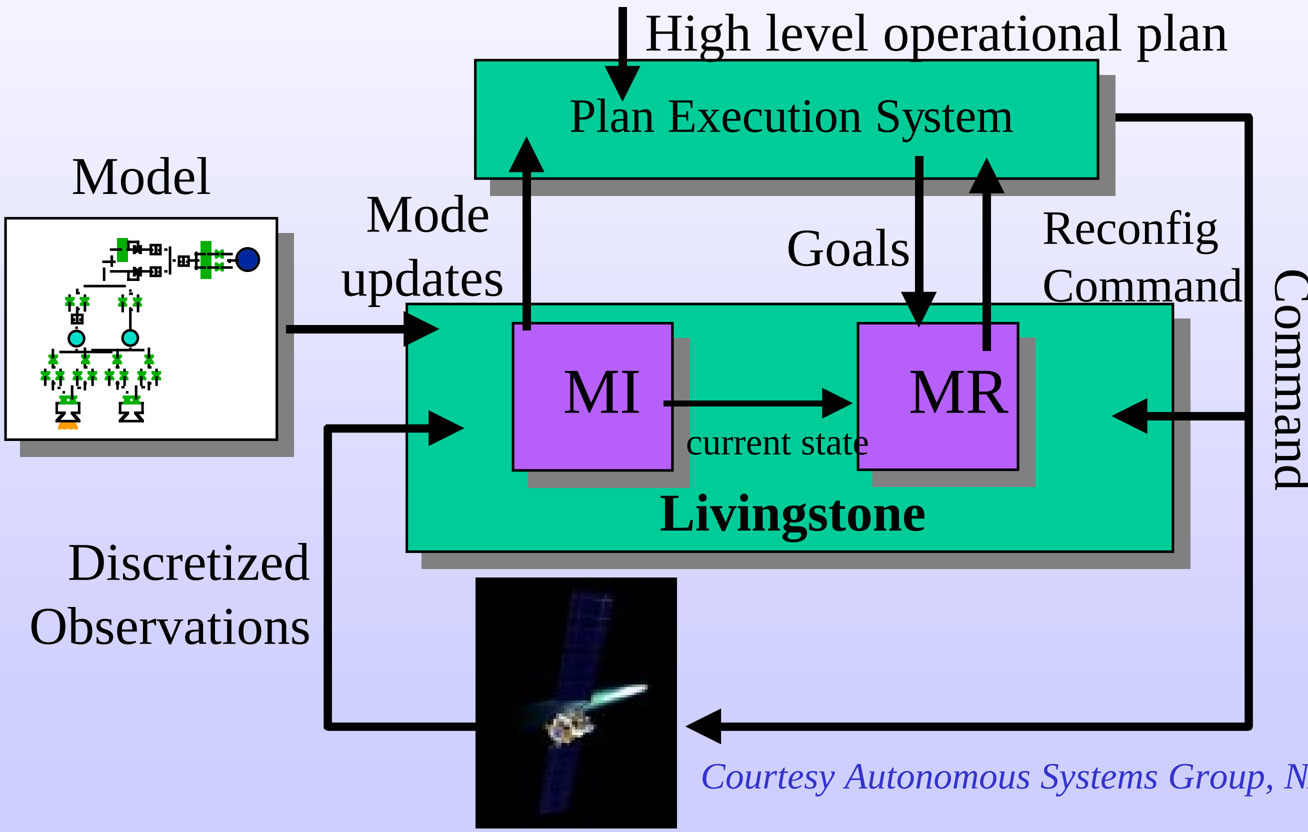
- Need for handling quantitative specifications:

  distances, durations, ...
- Timed automata : UPPAAL (UPPsala & AALborg)

  Modeling, simulation and verification of real-time systems.
- Translate planner models in UPPAAL
- Questions:
  - Consistency
  - Bounded Liveness
  - Mutexes

The Livingstone MIR

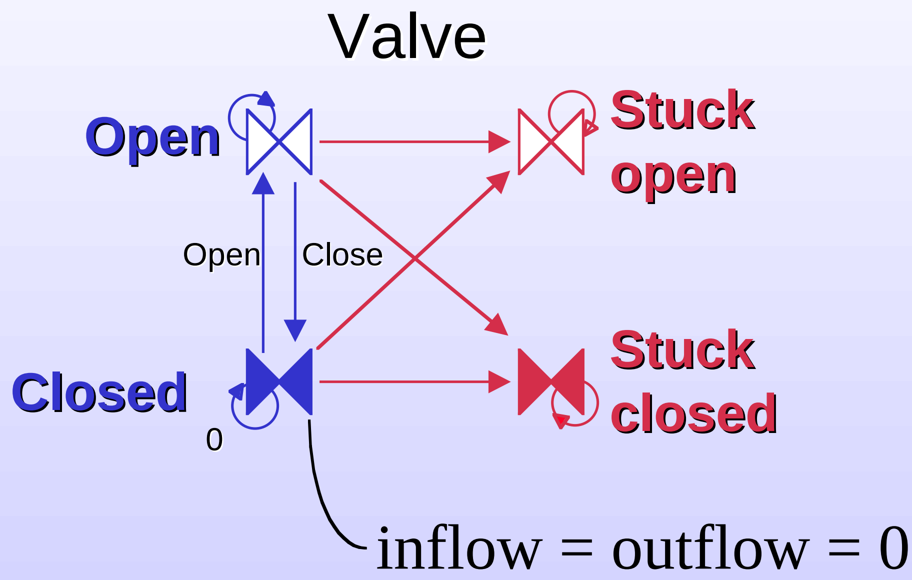
Remote Agent's model-based fault recovery sub-system



Courtesy Autonomous Systems Group, NASA Ames

Livingstone Models

- Models = concurrent transition systems
- qualitative values => finite state
- nominal/fault modes

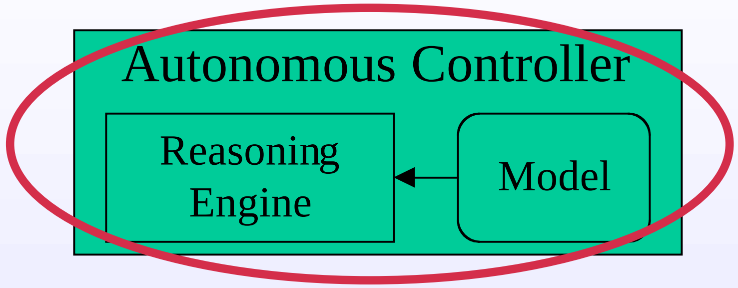


Courtesy Autonomous Systems Group, NASA Ames

From Livingstone to SMV

- Translate Livingstone models to SMV models similar languages => translation is easy
- Add property specifications
  - In temporal logic (CTL)
  - Using application-level extensions
- Initial work from CMU (Reid Simmons)
- Application: ISPP autonomous controller (KSC)
- Improvements in progress:
  - Correctness (=> formalize Livingstone)
  - Ease of use (more application-level extensions)

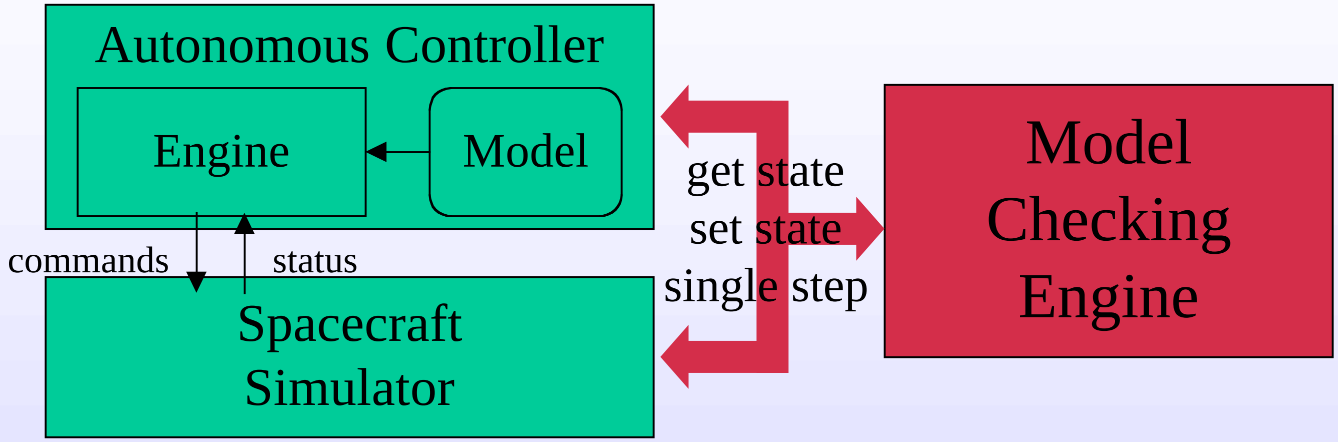
Verification of Model-Based Systems



- Model-based system = engine + model
- correct engine + correct plan ≠> good system !
  e.g. can fail to properly recognize a fault
- Model check? Very hard!

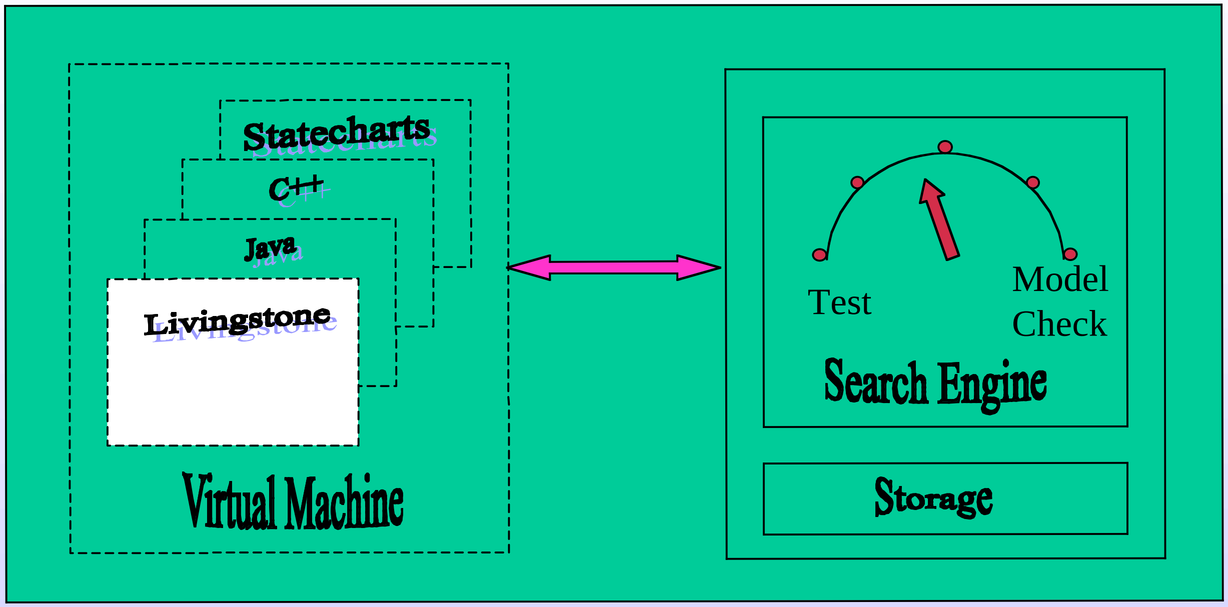
  Need (abstract) model of reasoning engine + model
  => complex, error-prone, huge state space

Analytic Testing



- Testing the real system => accuracy.
- Model-checking approach => exhaustive exploration.
- Restricted scenarios in simulator (otherwise too big).
- Completes, not supersedes, Model V&V (later stage).

Generic Verification Environment



- Principle: uncouple V&V subject from V&V algo.
- Common denominator of several projects in ASE.
- Hooks already present in Livingstone.

Conclusions

- Autonomy needs advanced V&V techniques
- Model checking for autonomous systems based on automated reasoning over discrete models (need to scale up)
- Translators to bridge the gap between design and V&V
- System-level V&V => Analytic testing
- For further study:
  - Continuous models (real-time, hybrid, neural nets)
    New mathematics required
  - Learning/adaptive systems *after* training
  - Learning/adaptive systems *including* training capabilities