HESSD '99: Liege

Using Model Checking to Help Discover Mode Confusions and Other Automation Surprises

John Rushby

Computer Science Laboratory
SRI International
Menlo Park, CA

Postulates

- Operators use mental models to guide their interaction with automated systems
- Automation surprises arise when the operator's mental model does not accurately reflect the behavior of the actual system
- Mode confusion is just a special case: the mental model is not an accurate reflection of the actual mode structure
    - Or is accurate but gets out of sync
- Mental models can be explicitly formulated as state machines
    - And we can "capture" them through observation, interviews, and introspection
    - Or by studying training manuals
      (which are intended to induce specific models)

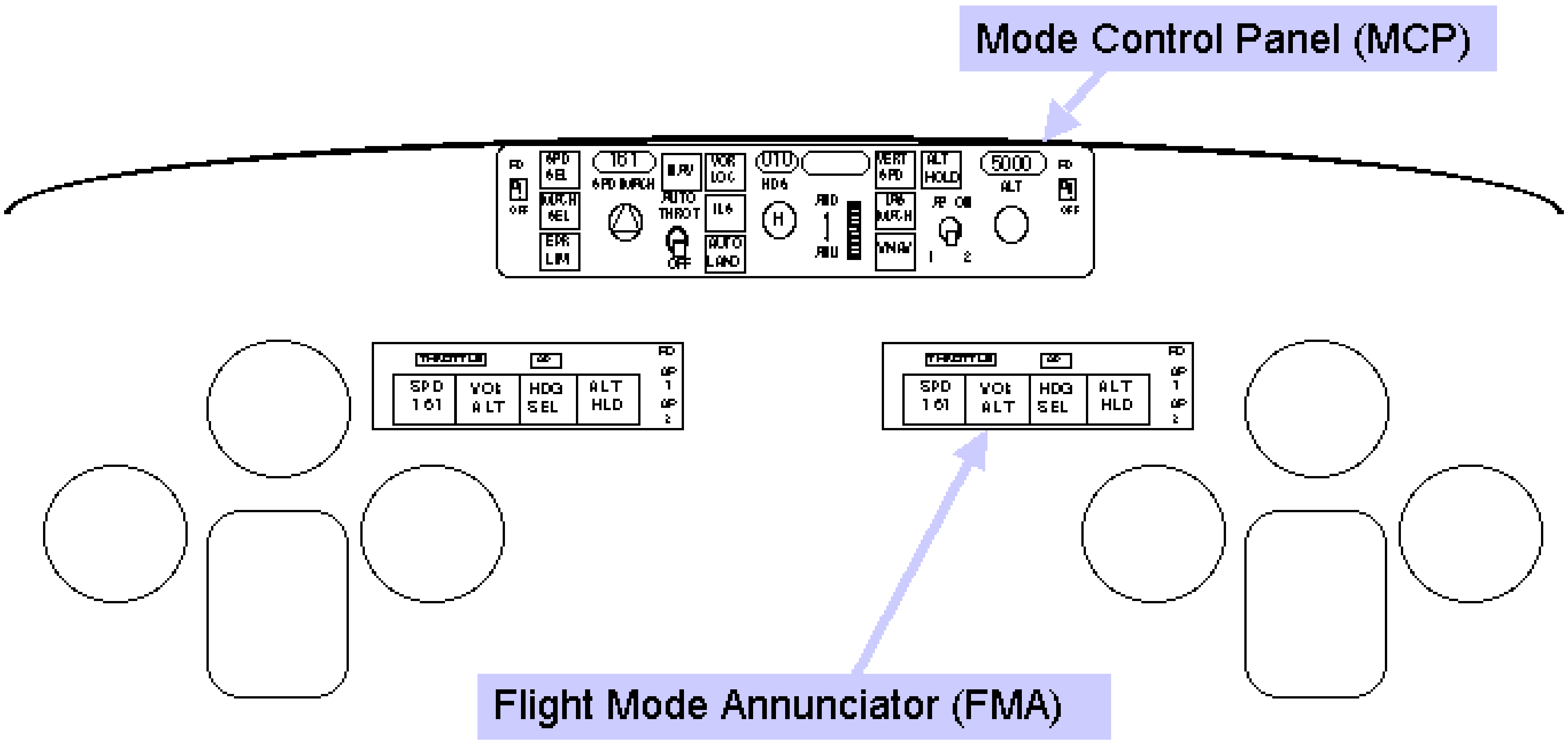
Facts

- The behavior of automated systems can be formulated in terms of (interacting) state machines
- These state machine descriptions are increasingly being used to document requirements and designs (cf. Statemate, UML)
- A technology called model checking can be used to examine the complete behavior of very large state machines
  - Can examine many millions of states
  - Used routinely in h/w design, s/w requirements analysis
  - Is largely automatic
- Can check whether certain properties are always true
  (e.g., every operator input is eventually acknowledged)
- Or can compare whether two state machines are consistent
- Produces counterexample when divergence found

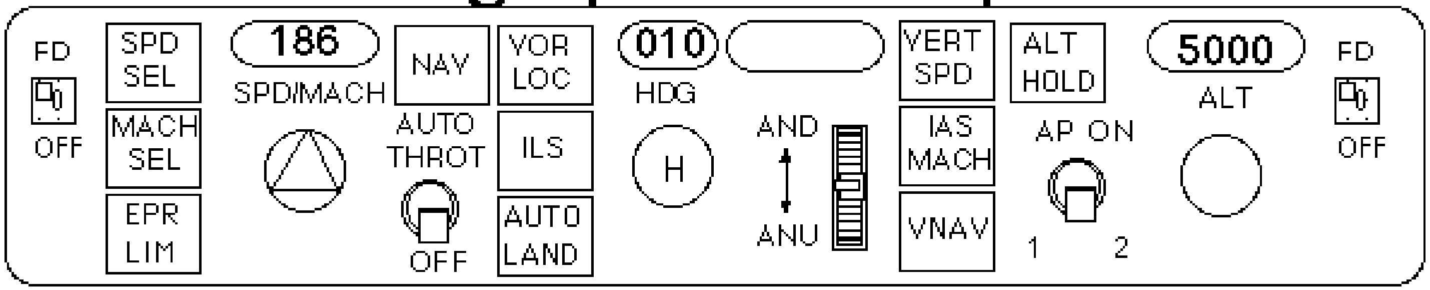
Putting These Together

- Take the design of an automated system
  - Represented as a state machine
- And a (plausible, or actual, or desired) mental model
  - Also represented as a state machine
- And check them for consistency
  - Using model checking
- Mental model should be faithful simplification of real system
- Any counterexamples will be potential automation surprises

Example: Altitude Bust Scenario from MD-88



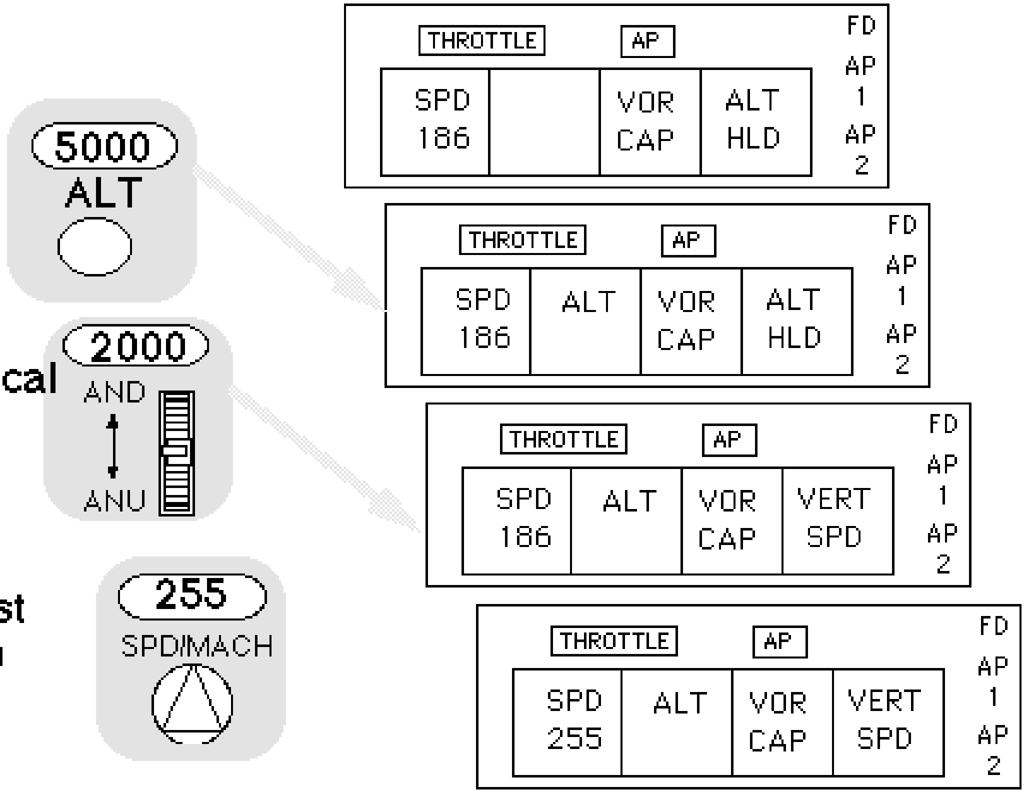
Setting up the Autopilot



1. Set altitude to 5000

2. Set vertical speed to to 2000

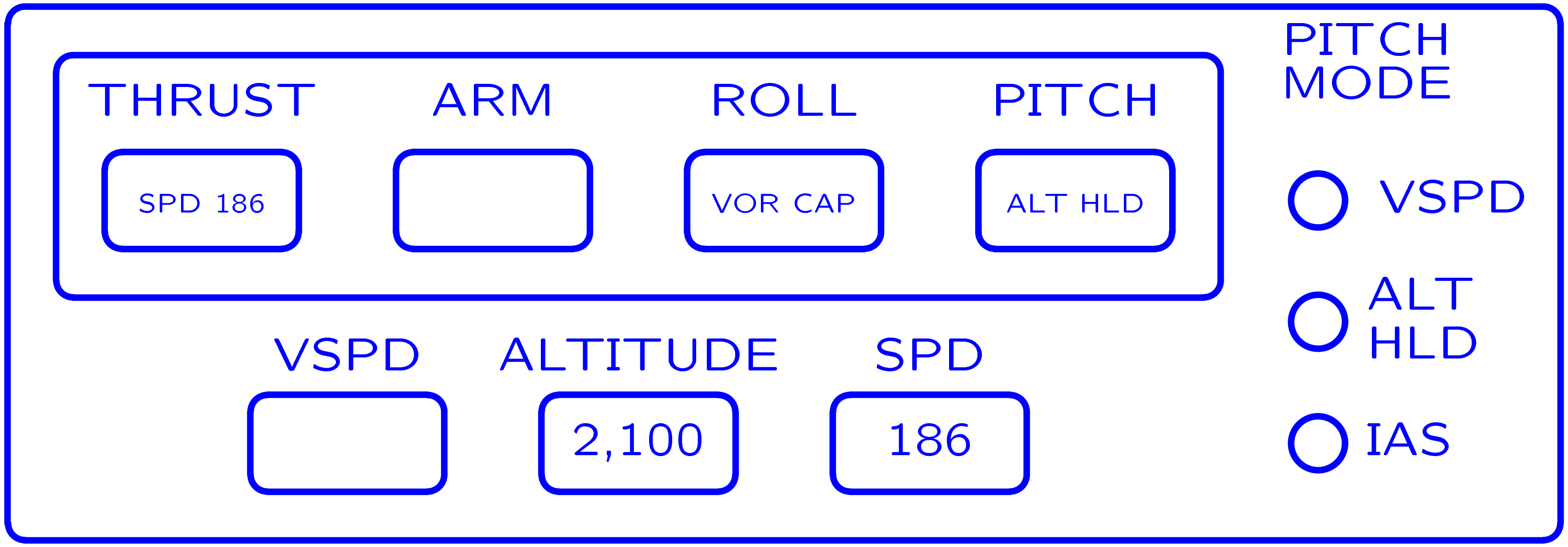
3. Set thrust to maintain a speed of 255.



8

Example: Altitude Bust Scenario

- Scenario describes an automation surprise in the MD-88 autopilot (from Ev Palmer)
- Crew had just made a missed approach
- Climbed and leveled at 2,100 feet

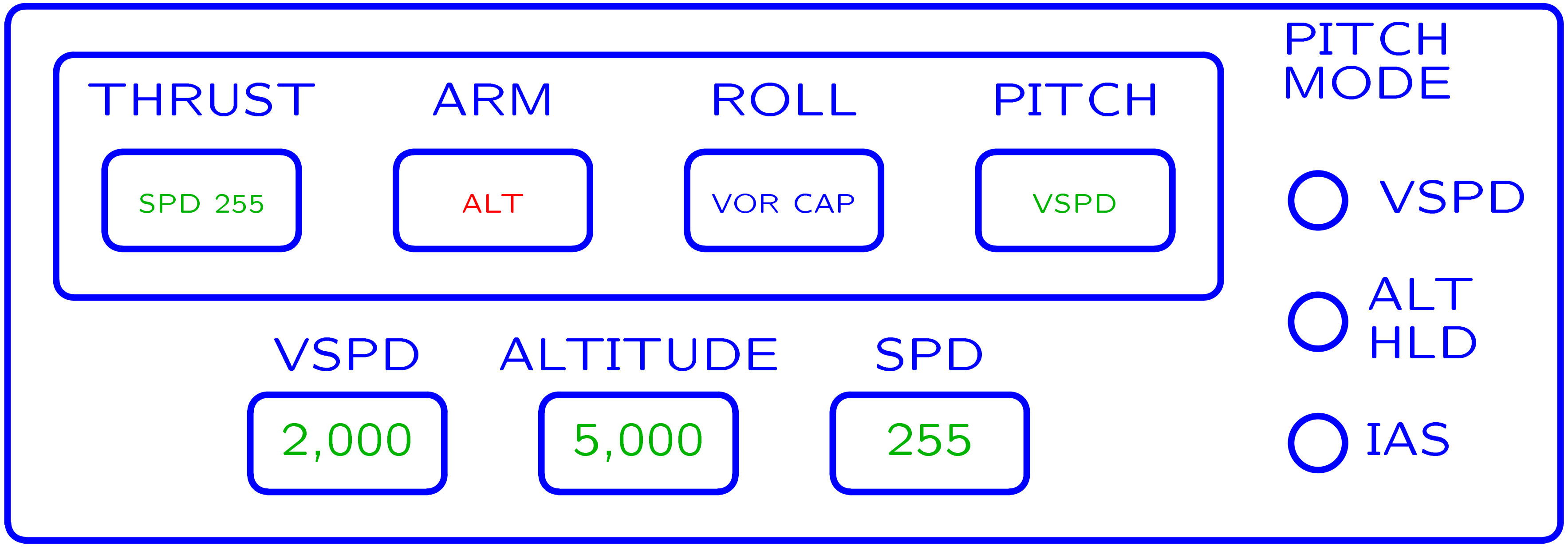


Altitude Bust Scenario: Mental Model

- The pitch modes determine how the plane climbs
  - VSPD: climb at so many feet per minute
  - IAS: climb while maintaining set airspeed
  - ALT HLD: hold current altitude
- The altitude capture mode determines whether there is a limit to the climb
  - If altitude capture is armed
    - ⋆ Plane will climb to set altitude and hold it
    - ⋆ There is also an ALT CAP pitch mode that is used to end the climb smoothly
  - Otherwise
    - ⋆ Plane will keep climbing until pilot stops it
- Color code: done by pilot, done by others or by automation

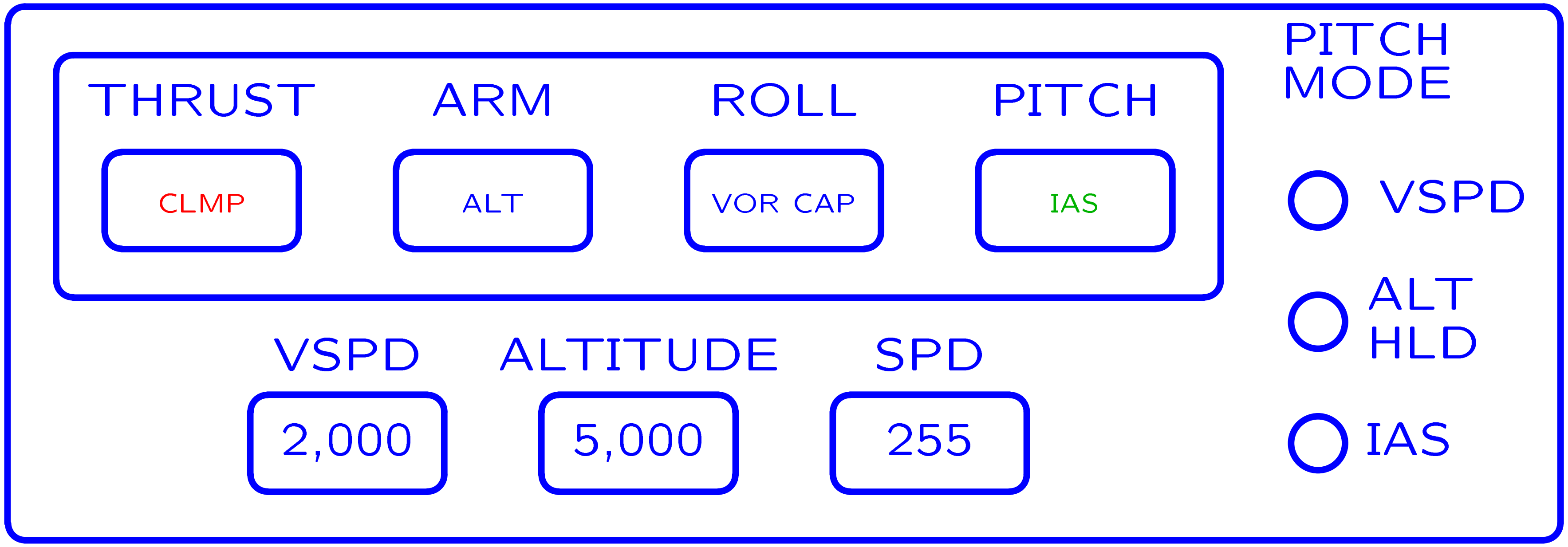
Altitude Bust Scenario—II

- Air traffic Control: "Climb and maintain 5,000 feet"
- Captain set MCP altitude window to 5,000 feet
  - Causes ALT capture to arm
- Also set pitch mode to VSPD with a value of 2,000 fpm
- And autothrottle (thrust) to SPD mode at 255 knots



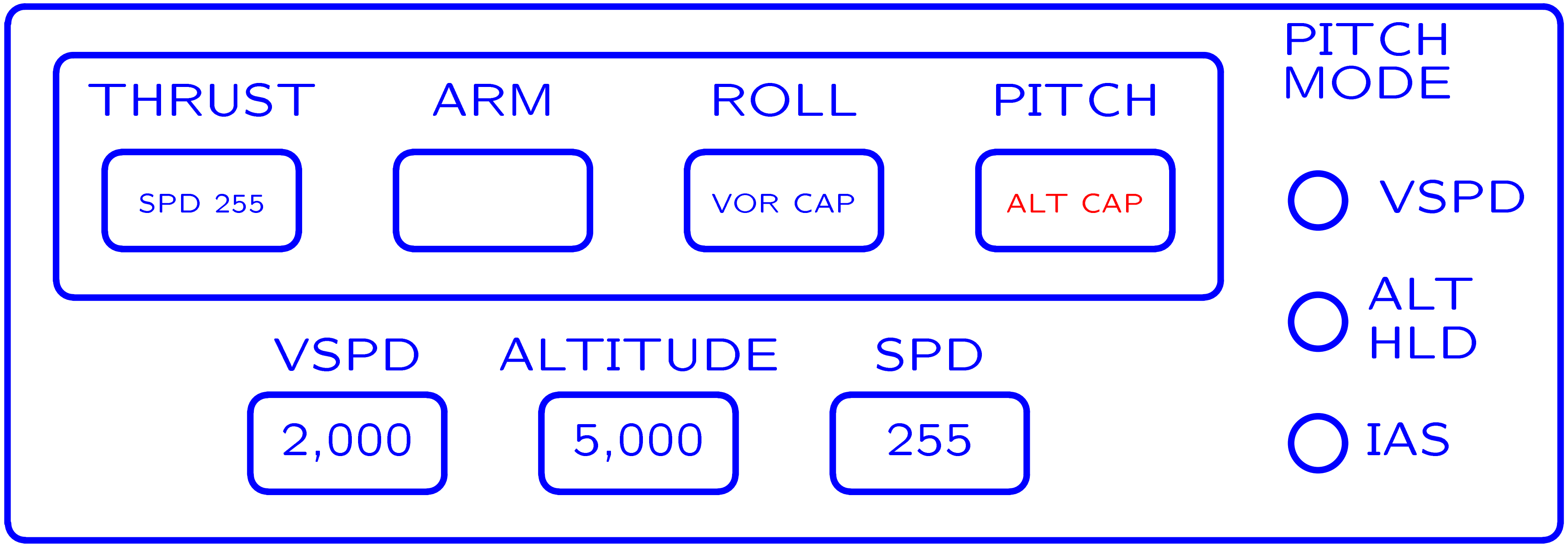
Altitude Bust Scenario—III

- Climbing through 3,500 feet, flaps up, slats retract
- Captain changed pitch mode to IAS
    - Causes autothrottle (thrust) to go to CLMP



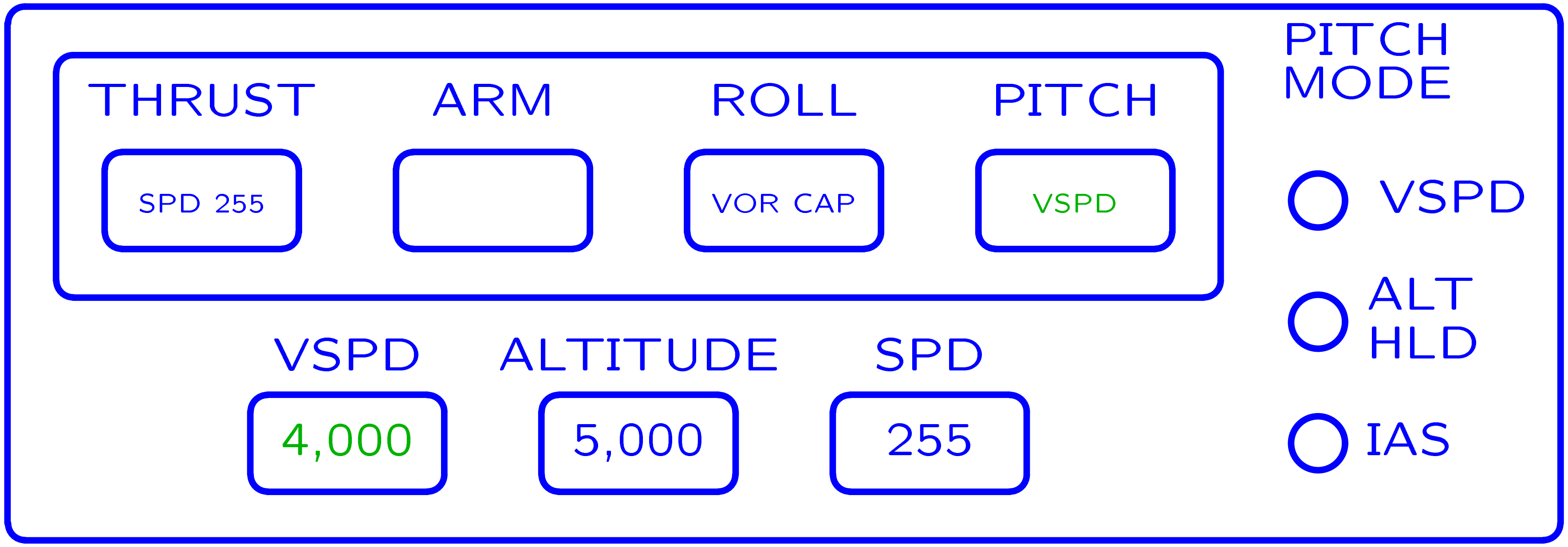
Altitude Bust Scenario—IV

- Three seconds later, nearing 5,000 feet, autopilot automatically changed pitch mode to ALT CAP
  - And disarmed ALT capture



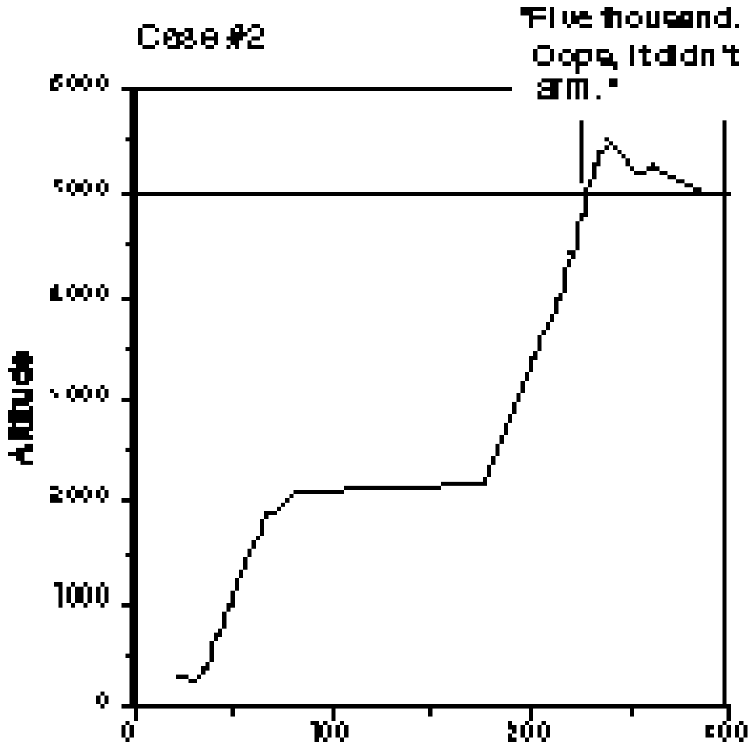
Altitude Bust Scenario—V

- 1/10 second later, Captain changed VSPD dial to 4,000 fpm



Altitude Bust: Outcome

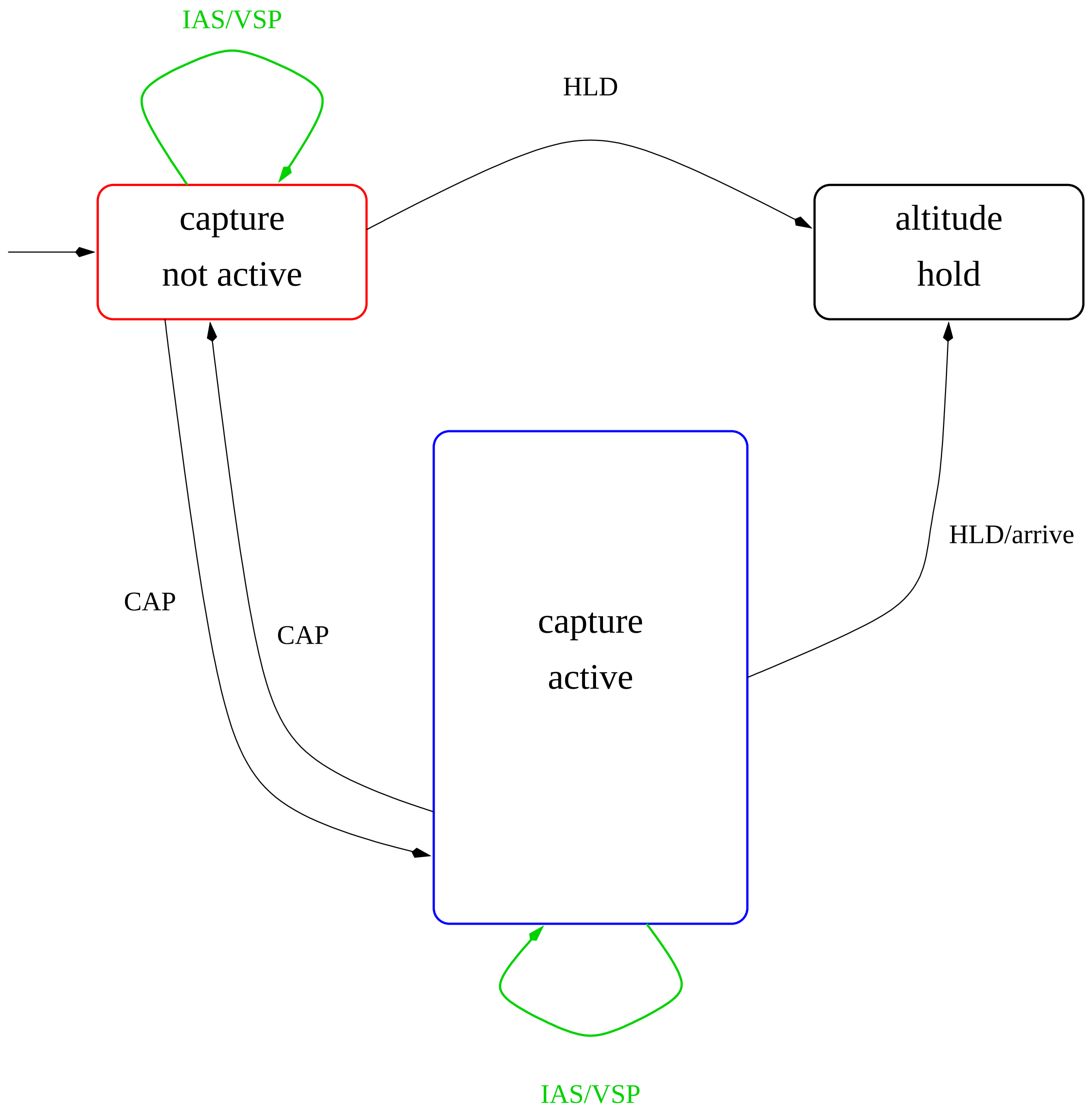
- Plane passed through 5,000 feet at vertical velocity of 4,000 fpm
- "Oops: It didn't arm"
- Captain took manual control, halted climb at 5,500 with the "*altitude—altitude*" voice warning sounding repeatedly



Automated Discovery of the Altitude Bust Scenario

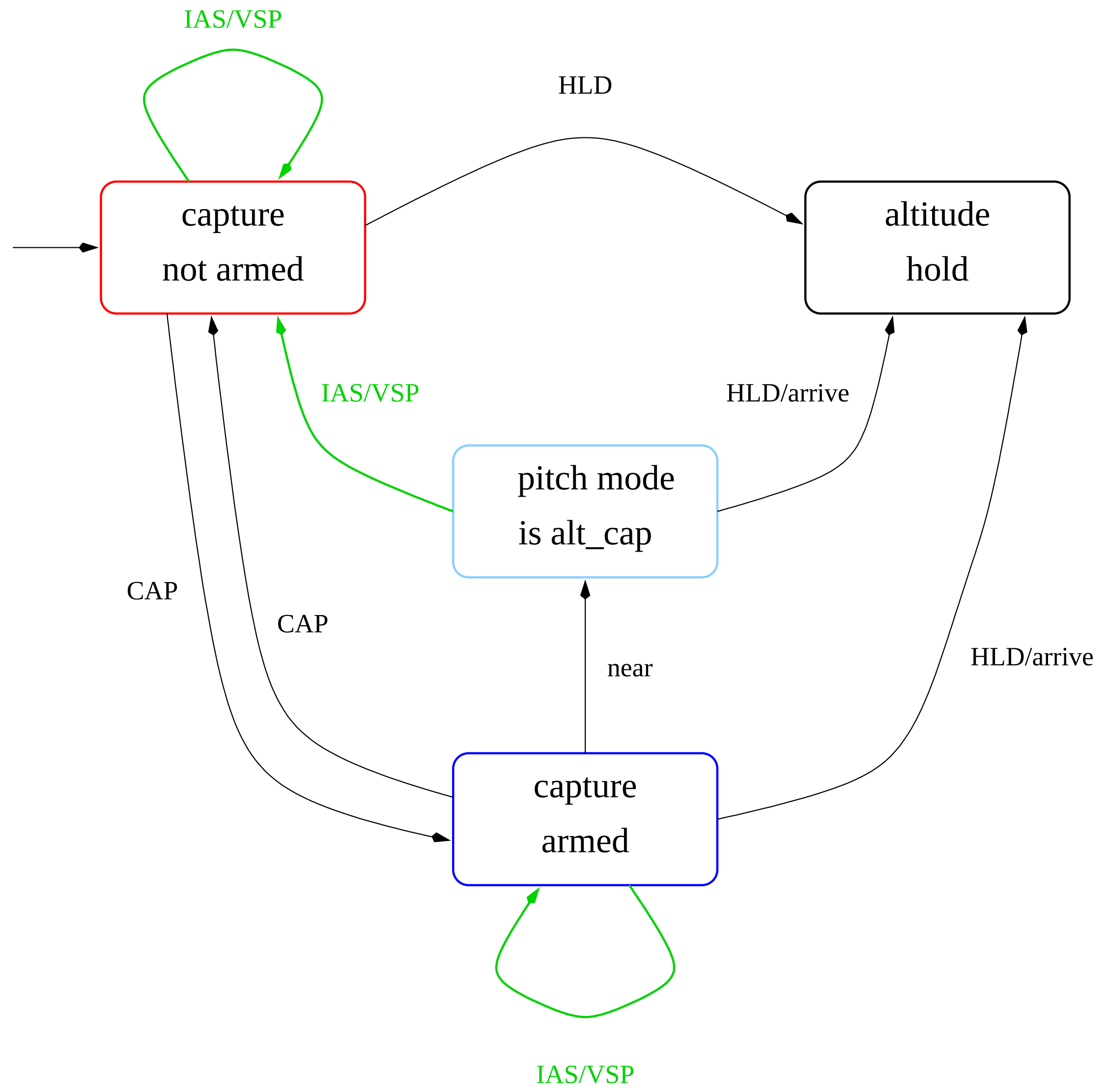
- I did it using a model checker called Murϕ
  - Comes from David Dill's group at Stanford
- But first I'll explain it using diagrams
  - These diagrams are not faithful to the actual models
  - Intended only to convey the idea

Mental Model



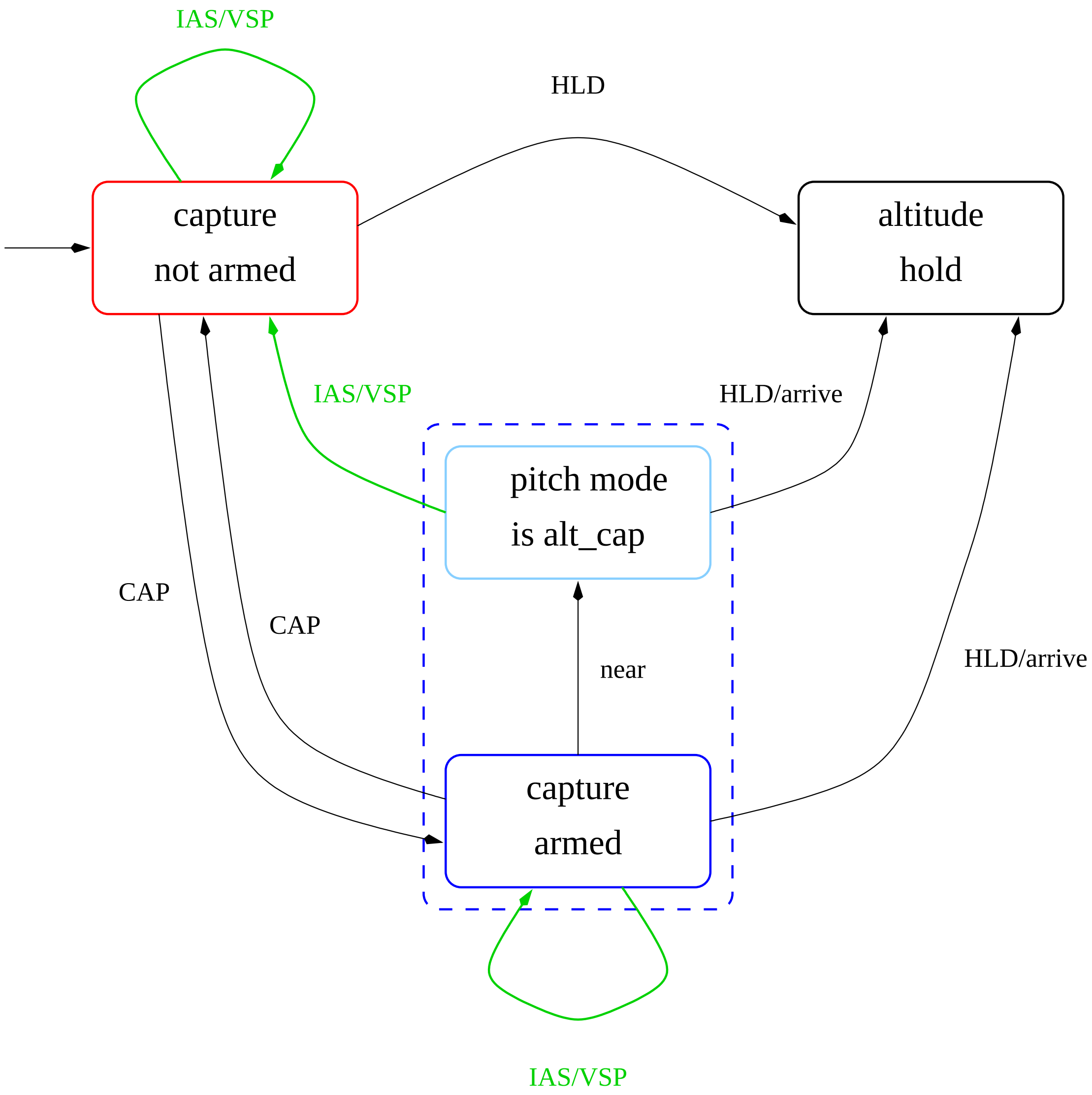
Whether capture is active is independent of the pitch mode

Actual System



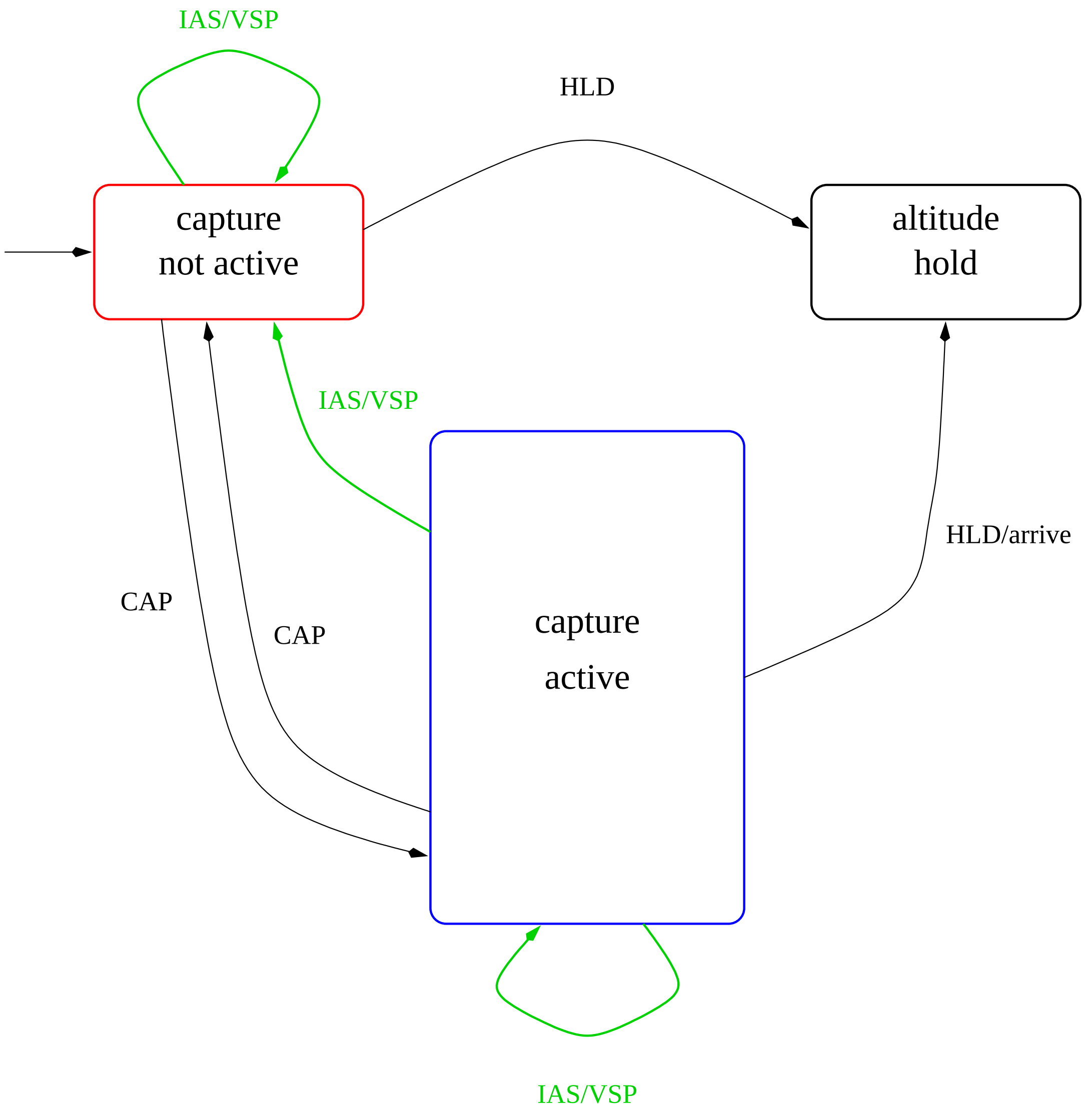
There is an `alt_cap` pitch mode that flies the final capture

Focus (Abstract) on Whether Capture Is Active



Capture is active if it is armed or if pitch mode is `alt_cap`

Abstracted System



Can compare this description directly with the mental model

Altitude Bust: Murϕ Specification

```
type
  pitch_modes: enum{vert_speed, ias, alt_cap, alt_hold};
var
  pitch_mode: pitch_modes;
  capture_armed: boolean;
  ideal_capture: boolean;

startstate
begin
  clear pitch_mode;
  capture_armed := false;
  ideal_capture := false;
end;

rule "ALT CAPTURE"
begin
  capture_armed := !capture_armed;
  ideal_capture := !ideal_capture;
end;

rule "HLD"
begin
  pitch_mode := alt_hold;
end;

rule "IAS"
begin
  pitch_mode := ias;
end;

rule "VSPD"
begin
  pitch_mode := vert_speed;
end;

rule "near"
begin
  if capture_armed then
    capture_armed := false;
    pitch_mode := alt_cap;
  endif;
end;

rule "arrived"
begin
  if pitch_mode = alt_cap then
    pitch_mode := alt_hold;
  endif;
  if capture_armed then
    pitch_mode := alt_hold;
    capture_armed := false;
  endif;
  if ideal_capture then
    ideal_capture := false;
  endif;
end;

invariant ideal_capture = (capture_armed | pitch_mode = alt_cap);
```

Altitude Bust: Murϕ Specification (closeup view)

```
rule "arrived"
begin
  if pitch_mode = alt_cap then
    pitch_mode := alt_hold;
  endif;
  if capture_armed then
    pitch_mode := alt_hold;
    capture_armed := false;
  endif;
  if ideal_capture then
    ideal_capture := false;
  endif;
end;

invariant ideal_capture = (capture_armed | pitch_mode = alt_cap);
```

Altitude Bust: Murϕ Analysis

```
Invariant "Invariant 0" failed.

Startstate Startstate 0 fired.
pitch_mode:vert_speed
capture_armed:false
ideal_capture:false
----------
Rule ALT CAPTURE fired.
capture_armed:true
ideal_capture:true
----------
Rule near fired.
pitch_mode:alt_cap
capture_armed:false
----------
Rule VSPD fired.
The last state of the trace (in full) is:
pitch_mode:vert_speed
capture_armed:false
ideal_capture:true
```

Altitude Bust: Results

- Found the "surprise" scenario (in 0.24 seconds)
- So did Leveson and Palmer
  - By looking for indirect mode changes
- They suggested a fix (see paper)
- I incorporated it in my model
- And found that it caused another surprise
- I fixed that
- And found yet another surprise
  (also present, in a different form, in original specification)
- I fixed that, and the system and the mental model now align

Altitude Bust: Additional Experiment

- Mode confusions can arise even with consistent models if operator loses sync
- I introduced a rule to model a forgetful operator
  (nondeterministically flips the mental state)
- Obviously this introduces mode confusions
- I then modified the mental model to "reload" the state of the mental model from a display that indicates whether altitude capture is armed
- This works (no surprises), even with a forgetful operator
- Can be used to validate choice of cues to be displayed
  - Human factors determines how to display those cues

Observations

- Once the initial model was constructed, these experiments required negligible effort (and only seconds of machine time)
- Provides complete demonstration of consistent behavior
  - Relative to the models used
  - Experience is that you learn more by examining all possibilities of simplified model than by probing some of the possibilities of the full thing (cf. simulation or testing)
- Approach does not supplant the contributions of those working in human factors and aviation psychology
  - Provides a tool to examine properties of their models using automated calculation

Speculation

- Mental model can also be interpreted as a requirements specification
  - I.e., reflect desired rather than observed operator behavior
- Lack of an accurate and simple mental model then suggests overly-complex design
  - How many states are needed?
  - Any complex data structures (e.g., a stack)?
- Can do design exploration on effects of
  - Simpler design
  - New operating instructions
  - Improved displays
  - Faulty operator

Comparisons

- Leveson enumerates error-prone design elements (e.g., indirect mode transitions)
    - And examines system design to locate them
        - ⋆ Must then determine whether those found are real problems in their specific context
    - Examination is not automated
    - Tension between examining too much and too little
- Butler (NASA Langley), Miller (Collins) and colleagues use mechanized formal methods (theorem proving and model checking) to examine specification of autopilot for safety invariants (e.g., no mode change without pilot input)
    - Similar to my approach
    - But mental model is richer specification than an invariant

Plans

- This approach should scale to bigger examples
- Hope to try it on
  - Collins specification of mode switching in a model autopilot
  - Honeywell specification for a FMS
  - Or FANS handoff
- Would like to develop a dialog between the automated formal methods and human factors communities
  - Denis Javaux (Liege) has a cognitive model for the simplifications subconsciously applied to mental models
  - Plan to apply my technique to his models

Technical Challenges: Methodological

Can only go so far modeling just the mode behavior

And abstracting everything else away

- Need to investigate incorporating limited models of the environment and of the control behavior
  - E.g., to distinguish climbing from descending, up from down
  - Qualitative physics may prove adequate
    - ⋆ Reason about signs of quantities and rates of change
    - ⋆ E.g., climb means height increases (derivative is +ve)
  - May need hybrid automata (and model checkers for these)
- Also need to look at real time issues (e.g., delay between reading display and taking action)

Technical Challenges: Formal Methods

- Application of mathematical modeling and mechanized analysis to computational systems
- Just like the use of mathematical modeling in other engineering disciplines (e.g., finite-elements analysis)
    - Motivation is pragmatic, mechanization is the essence
- Model checking is a fairly mature technology but runs out of steam around 300 state bits (e.g., 20 state variables)
- Larger and more complex examples will require greater support for automated abstraction (faithful simplification)
- And more aggressive integration of theorem proving (to justify or construct abstractions) and model checking
- This coincides with the research agenda in formal methods

To Learn More (About Formal Methods)

- Browse general papers and technical reports at http://www.csl.sri.com/fm.html
  - /reports/html/hessd99.html describes this work and provides the Murϕ code
  - Links to Murϕ there also
- Information about our verification system, PVS, and the system itself are available from http://www.csl.sri.com/pvs.html
  - Freely available under license to SRI
  - Allegro Lisp for Suns or Linux
  - Need 64M memory, 100M swap space, 200 MHz or better