Fundamentals of State Analysis

Outline

- Introduction & Overview
- Model-based Programming
- Execution of Model-based Programs
- Fundamentals of Model-based Reasoning
- Modeling via State Analysis
  - Model-based Programming in the Systems Engineering Context
  - State-based Control Architecture
  - Overview of State Analysis, a Model-based Systems Engineering methodology
  - Example
- Advanced Methods
- Conclusion
Where does the Model-based Program come from?

- Who produces it?
- How is it produced?
- When in the Project Lifecycle?
- Need to put Model-based Programming into the Systems Engineering context...

Facts of Life

- Somehow, the systems engineer’s understanding of the spacecraft behavior (and its environment) must inform what software designers build
  - Whether overt and explicit, or hidden quietly in the minds of the engineers, models have always existed
  - Understanding and modeling are essentially the same thing
  - Software design is ultimately a reflection of this understanding, and therefore a reflection of these models

- To the extent the software design reflects the systems engineer’s understanding, the software will perform as the systems engineers desire
  
  That is, …
System Software is a Surrogate for Systems Engineers...

...and Software Engineers perform the transformation

The Holy Grail...

Models from Systems Engineers are provided directly to the embedded system, which is capable of reasoning through them to accomplish mission objectives!
**Reality Check**

- Even the State-of-the-Art is far from fully achieving this Vision
- Yet even incremental steps toward this Vision can greatly improve on the current practice, producing software that is:
  - Less error-prone
  - More robust to off-nominal situations
  - Cheaper
  - Easier to verify
  - Etc.
- Two parts to this challenge:
  - Must develop representations and algorithms that endow the embedded system with the requisite reasoning capabilities (e.g., Model-based Programming and Execution)
  - Equally importantly, must provide Systems Engineers with methods and tools that help them reason through the system design and develop models in a rigorous way
- Recent advances in Model-based Engineering show promise
  - One model-based systems engineering methodology is particularly well-suited for this purpose...

**State Analysis**

- A uniform, methodical, and rigorous approach for...
  - Discovering, characterizing, representing, and documenting the states of a system
  - Modeling the behavior of states and relationships among them, including information about hardware interfaces and operation

An example State Effects Diagram
State Analysis

• A uniform, methodical, and rigorous approach for…
  – Discovering, characterizing, representing, and documenting the states of a system
  – Modeling the behavior of states and relationships among them, including information about hardware interfaces and operation
  – Capturing the mission objectives in detailed scenarios motivated by operator intent
  – Keeping track of system constraints and operating rules, and
  – Describing the methods by which objectives will be achieved

• For each of these design aspects, there is a simple but strict structure within which it is defined…

State-based Control Architecture

- System operation via overt, objective statements of intent
- Clear delineation between control system and system under control
- Explicit state variables
- Models inform all aspects of control system
- Separation of estimation from control
Overview of State Analysis

1. System to be controlled

2. State Analysis produces model

3. Model informs software design

4. Model informs operations

Coverage of System States is Wide

- Dynamics
  - Vehicle position & attitude, gimbal angles, wheel rotation, …

- Environment
  - Ephemeris, light level, atmospheric profiles, terrain, …

- Device status
  - Configuration, temperature, operating modes, failure modes, …

- Parameters
  - Mass properties, scale factors, biases, alignments, noise levels, …

- Resources
  - Power & energy, propellant, data storage, bandwidth, …

- Data product collections
  - Science data, measurement sets, …

- Data management & transport policies
  - Compression/deletion, transport priority, …

- Externally controlled factors
  - Space link schedule & configuration, …

Note that this also covers types of states not addressed by the “discrete” Model-based Programs presented earlier. State Analysis is a very general approach, compatible with, but not limited to, Model-based Programming.
**Following Leads — An Example**

### Standard Questions:

- **What do you want to achieve?**
  - Move rover to rock

- **What's the state to be controlled?**
  - Rover position relative to rock

- **What evidence is there for that state?**
  - IMU, wheel rotations, sun sensor, stereo camera

- **What does the stereo camera measure?**
  - Distance to terrain features, light level, camera power (on/off), camera health

- **How do you raise the light level?**
  - Wait until the sun is up

- **Where is sun relative to horizon?**
  - Etc.

### Common Elements:

- **Goal**
- **State Variable**
- **Measurements**
- **State Effects Model**
- **Etc.**

---

**State Discovery Overview**

- **Start with needs** — objectives for controlling the system
- **Identify what you know** — states that represent what needs to be controlled to meet the objectives
  - Identify additional state variables that affect identified state variables
    - Describe *state effects models* between state variables
  - Identify measurements needed to estimate the state variables
    - Define *measurement models*, these may uncover additional state variables
  - Identify commands needed to control the state variables
    - Define *command models*, these may uncover additional state variables
- **State discovery is an iterative process**
  - Repeat above 4 steps on newly discovered state variables
- **Repeat for each objective**
State Discovery Example

- Top level objective: fire one of the two engines...
State Discovery Example

- State effect: Valve 1 & 2 Outflow affect Engine Mode

State Discovery Example

- State effect: Valve Position and Tank Pressure affect Flow
State Discovery Example

**Engineering decision:**
Let’s *not* include Valve Outflow as an explicit state variable to be modeled. Rather, we’ll capture the Tank Pressure and Valve Position effects on Engine Mode, *directly* (Valve Outflow is considered part of the state effects model. It’s also a measurement).

State Discovery Example

- Measurement: Valve Outflow is affected by Tank Pressure and Valve Position
**State Discovery Example**

- Command: Engine Command affects Engine Mode

**State Discovery Example**

- State effect: Battery Charge and Power Switch Position affect Engine Mode
State effects are considered in the design of the control program; in our example, the Camera Health can be adversely affected by the Engine in Firing mode…
System Model Underlying the State Effects Diagram

Camera Model

- **Off**: (power_in = zero) & (shutter = closed)
- **Resettable**: (cam_cmd = turnoff)
- **Contaminated**: (image = faulty)

Camera Model

- **Standby**: (power_in = nominal) & (shutter = open)
- **Failed**: (thrust = zero) & (power_in = zero)
- **Firing**: (thrust = full) & (flow_in1 = nominal) & (flow_in2 = nominal)
### State Analysis and MBP in the Project Lifecycle

<table>
<thead>
<tr>
<th>Phases</th>
<th>Advanced Studies</th>
<th>Formulation</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-Phase A</td>
<td></td>
<td>Phase A</td>
<td>Phase B</td>
</tr>
<tr>
<td>Advanced Studies</td>
<td>Mission &amp; System Definition</td>
<td>Preliminary Design</td>
<td>Design &amp; Build</td>
</tr>
<tr>
<td>Phase C</td>
<td></td>
<td>Phase D</td>
<td>Phase E</td>
</tr>
<tr>
<td>Approval to Lead</td>
<td>Assembly Test (ATL)</td>
<td>Start Launch</td>
<td>Operations</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key Events</th>
<th>Concept Approved</th>
<th>Preliminary Agreement</th>
<th>Contract &amp; Approval</th>
<th>Approval to Test</th>
<th>ATL Start</th>
<th>Approval to Launch</th>
<th>Critical Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reviews</td>
<td>MCIR</td>
<td>PMIR</td>
<td>PDR</td>
<td>CDR</td>
<td>ARIR</td>
<td>MRR</td>
<td>PLAR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sys. Eng.</th>
<th>High-level SA for early design trades</th>
<th>State Discovery &amp; Initial SA</th>
<th>Preliminary Detailed SA</th>
<th>Final Detailed SA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Refined s/w performance model</td>
</tr>
<tr>
<td>V&amp;V</td>
<td></td>
<td></td>
<td></td>
<td>S/W adaptation updates</td>
</tr>
<tr>
<td>Operations</td>
<td></td>
<td></td>
<td></td>
<td>V&amp;V</td>
</tr>
<tr>
<td>Mgmt &amp; Costing</td>
<td></td>
<td>Initial s/w cost model</td>
<td>Detailed s/w cost model</td>
<td>Tracking progress against model</td>
</tr>
</tbody>
</table>

### For More Info on State Analysis...

- **2-day State Analysis Training Course:**
  - Course material distributed by Caltech/JPL
  - Contact Mitch Ingham (michel.d.ingham@jpl.nasa.gov) for more information
- **Overview paper:**
Up Next…

- Advanced Methods in Model-based Autonomy
  - Extensions and evolutions of the fundamental approach we’ve described
  - Recent & ongoing work in Model-based Programming and Model-based Execution

Backup Slides
State Analysis Is
a Gradual Discovery Process

• Prompted by a standard set of basic questions
• The answer to each question is expressed in terms of a few basic canonical elements

- Each answer prompts additional questions, and so on
- The model unfolds a step at a time until all the elements are identified
• This is enabled by adopting a regular architectural structure

Some of the State Analysis Questions

A closely integrated set of control concepts

State Knowledge
- What state variables describe the system under control?
- How do they represent variation in time, uncertainty, etc.?
- To what extent must they describe past and future?
- How are they stored or transported?

State Constraints
- What type of constraints must the control system be able to enforce?
- How can constraints be elaborated, projected, and scheduled?
- How is constraint execution regulated and monitored?

State-based Models
- How do states of the system under control behave and affect one another?
- How do commands affect states, nominally or otherwise?
- In what way do measurements depend on them?

Goal Achievers
- How are models used in determining state knowledge from potentially inconsistent or uncertain evidence provided by measurements, commands, and other states?
- How are models used in determining what commands to issue to achieve goals?

Measurements and Commands
- What are the measurements and commands at the interface to the system under control?
- How does the system under control handle them?
- How are they stored or transported?
- How is system data managed and transported?

Deployments
- How are control system functions deployed across the system?
- How can computing and communication attributes be used to assign responsibilities among deployments?
**Where Do You Start?**

- **Order isn’t important**
  - Start with what you know best
  - All analysis elements are developed concurrently
  - Proceed as issues and information dictate
  - Questions not immediately decided will be queued for later answers

- **The process is iterative**
  - Breadth is initially more important than depth
  - Early results needn’t be complete or accurate
  - Refinement can continue throughout development, into test, and even into operations, if necessary

- **It tends to drive early discovery of issues**
  - Standard questions prompt and guide broad exploration

---

**Am I Done Yet, Have I Gone Too Far?**

- **You’re done when everything you care about is accounted for as states or effects**
  - Make models only as complex as needed; apply good engineering judgment
  - If a state doesn’t affect any state you care about, and you don’t care about the state, then you don’t need to model it (e.g. location of Venus for a Mars Lander)
  - If a state, command, measurement, or effect is purposely omitted because it is deemed insignificant, the reason should be documented

- **You’ve gone too far if the same state is represented in more than one state variable**
  - Unique state representation ensures consistency and simplifies implementation