A Formal Approach to Specifying and Verifying Spacecraft Behavior

Allan McInnes
2006-02-28

Supervisor: Dr. Charles Swenson
Outline

- Motivation
- Existing work
- Approach
- Road ahead
Spacecraft behavior: a loose definition

- Interaction with other systems
- Sequencing of inputs, outputs, and state changes
- System modes and mode changes
- System-level interaction between subsystems
Current practice: subsystem design

- Subsystems are precisely specified through drawings, schematics, and code
- Subsystem designs undergo rigorous analysis
Current practice: spacecraft behavior

- Ambiguous behavior specifications

ACS.REQ.ASF.5 ACE SAFEHOLD MODE TRANSITIONS
Only a ground command or ACE box 8085 processor failure shall transition the ACS out of ACE safehold.

ACS.REQ.SSF.2 SCS SAFEHOLD MODE TRANSITIONS
Only a ground command shall transition the ACS to a higher mode. The ACS shall transition back to ACE safehold either by ground command, SCS microprocessor cold start, or by the ACS failure detection and handling logic.

- Extremely limited design and analysis of behavior
The problem

Without analysis, incorrect system behavior is unlikely to be detected until I&T
Existing work

- NASA has experimented with formalizing behavior
  - Fault-protection verification with PVS [Easterbrook]
  - Promela/SPIN for a variety of verification tasks
    - Fault protection [Feather, Barltrop]
    - Autonomy [Havelund, Smith]
    - Software [Gluck, Visser]
  - Model-based programming with RMPL [Ingham]
- Limited formal modeling on non-NASA programs
  - SACI-1 C&DH fault-tolerance [Mota]
  - Abrixas embedded power controllers [Schlingloff]
Approach

- Use *process algebra* to model spacecraft behavior
- Build simplified models that capture key behavior
- Link models to existing spacecraft terminology
- Create tools to make model-building easier
Communicating Sequential Processes

- A *Process Algebra*

  - Process = Abstract behavior
  - Algebra = Operators and axioms

- A language for specifying concurrent behavior
- A language for modelling concurrent systems
- A theory for reasoning about behavior and systems
Is it practical?

- CSP in industry: a few examples
  - INMOS – processor and logic design
  - Qinetiq – fault-tolerant ad hoc wireless networks
  - Praxis High-Integrity Systems – secure e-commerce
  - Daimler Aerospace – fault-tolerant software for ISS
Simple example

\[ SC = (InstrIF \\
[ instr\_out \leftrightarrow buff\_in ] \\
TlmBoard) \\
[ instr\_cmd \leftrightarrow instr\_cmd, buff\_status \leftrightarrow buff\_status ] \\
OBC \]
Example process

Telemetry Board Behavior

- Both stores and transmits data
- Only accepts data if it has room
- Only downlinks if it has data
- Downlinks 1 data item at a time
- Signals changes in status

\[
TlmBoard = \text{buff\_status!notfull} \rightarrow TB(\text{MaxBuff}, \langle \rangle)
\]

\[
TB(N, \text{buff}) = \begin{cases} 
(# \text{buff} < N) & \& \text{Store}(N, \text{buff}) \\
\text{\(\square\)} & \\
(# \text{buff} > 0) & \& \text{Transmit}(N, \text{buff}) 
\end{cases}
\]

\[
\text{Store}(N, \text{buff}) = \text{buff\_in?d} \rightarrow \\
\quad \text{if } # \text{buff} == N - 1 \\
\quad \text{then } \text{buff\_status!full} \rightarrow TB(N, \text{buff} \wedge \langle d \rangle) \\
\quad \text{else } \text{buff\_status!notfull} \rightarrow TB(N, \text{buff} \wedge \langle d \rangle)
\]

\[
\text{Transmit}(N, \text{buff}) = \text{downlink!head (buff)} \rightarrow \\
\quad \text{if } # \text{buff} == N \\
\quad \text{then } \text{buff\_status!notfull} \rightarrow TB(N, \text{tail (buff)}) \\
\quad \text{else } TB(N, \text{tail (buff)})
\]
Spacecraft behavior: definition revisited

- Function sequences
- Event sequences
- State transitions
- Mode transitions
- Behavior constraints
- Inter-subsystem communication
- Shared states and resources
- Triggering events
Framework for behavior specification

- Combine behavior “components” to specify
  - Simple subsystem and systems behaviors
  - More complex autonomous behaviors
  - Fault responses

- Use COTS tools to verify that
  - Specifications are internally consistent
  - Specified behavior is correct
The road ahead

- Collaboration with Dr. Brandon Eames, Jared Crace, and Joe Graham
  - **Short term**: GUI specification tools (built on GME)
  - **Long-term**: code and FPGA generation from specs
Summary

- Spacecraft system-level design should be as **rigorous** as subsystem-level design.
- Our research
  - builds on **industrially-proven** formal methods
  - allows rigorous **system-level specification**
  - aims to make system-level rigor **accessible**