Integrating Model-Based Systems & Software Engineering

Robert D. Rasmussen, Ph.D.
Two Worlds

Systems Engineering

Software Engineering
Typical Spheres of Concern

Software Engineering

Inward looking:
- Threads & communication
- Modularity, encapsulation...
- Languages, libraries, operating systems...
- Time & memory management
- Real time execution
- Commands & telemetry
- Patterns, abstractions...
- Data representation, serialization...
- Deadlocks, access violations, exceptions...
  etc.

Outward looking:
- Mission scenarios
- Functional decomposition
- Component selection
- Resource allocation
- Performance
- Uplink and downlink
- Flight rules and constraints
- Algorithms
- Failure modes & fault protection etc.

System behavior
The Role of Software

Then...

System

Flight
- Sub-system
  - Assembly
  - Computer
- Instrument

Ground

similar

...what the system is
Ascending
The Role of Software

Now...

System

Instrument

Flight Software

Ground Software

Computer

Sub-system

Assembly

similar

...what the system does
The Behavior Crisis

- Affordability vs. confidence
  - Growing requirements
  - Poorly understood complexity

- Operability vs. flexibility
  - Uncertain dynamic environments
  - Poorly handled by simple modal behaviors

- Robustness vs. effectiveness
  - Surprising system behaviors
  - Poorly understood complexity
Converging Interests

- Systems and software engineers share a common and growing interest in system behavior

but

- What’s the difference between…
  - specifying
  - defining
  behavior?

- When models are involved, not much!
Why Talk about Models?

- Models help...

  **Separate Essentials from Incidentals**

  - **Software engineers** have adopted models to describe and understand designs...
    - Composition, relationships, behavior, etc.
    - separate from arcane details of implementation

  - **Systems engineers** have begun to use similar methods to describe and understand systems
    - Resources, scenarios, interactions...

- Both help see systems as an *integral whole,* not merely an *integration of parts*
Lingua Franca

With models, systems and software engineers can talk about the essentials of system behavior in common terms

- Separate from the incidentals of system construction
- Separate from the incidentals of computer programming
Modeling Languages

Do these...

UML

SysML

do the job?

No. Not by themselves

Other essentials: Architecture, Methodology, Infrastructure, Understanding, Commitment
Architecture

- Concepts
  - What are the key ideas behind your model?

- Composition
  - What generic types of elements comprise a model, and how should they relate to one another?

- Principles
  - What fundamentals of good design does your modeling approach enforce?
Methodology

Order

- Where should you begin, and what steps and criteria can you offer to guide the modeling activity?

Structure

- Is there a canonical organization that can be used to shape modeling products?

Discipline

- How will your model be reviewed and validated for correctness and adherence to the architecture?
Infrastructure

Formality

- How are architectural and methodological concepts and relationships formally captured?

Access

- Where and how are models collected, and how are related items found, compared, and composed?

Tools

- How is the generation, communication, and use of models and modeling products facilitated?
Understanding

Training
- By what means do practitioners become fluent in the modeling architecture, methodology, and infrastructure?

Value
- How will you know your modeling activities are gaining the benefits you seek, and how will you convince others?
An Unified Approach to Modeling System Behavior

- **Architecture**
  - Key concepts regarding the cognizant control, concrete objectives, and failure management

- **Methodology**
  - An incremental exploration guided by structure and principles of the architecture

- **Infrastructure**
  - Captured in both formal modeling tools and software implementation frameworks

- **Understanding**
  - Extensive seminars, classes, examples, and measured pilot activities
Overview

State Analysis

Model of System Under Control

Control System Architectural Framework

System Under Control

http://mds.jpl.nasa.gov/
Experience, So Far

- Approach has been quite effective on a modest scale
  - Concepts have matured, demonstrating great powers of explication
  - Systems and software communicate effectively and penetrate rapidly
  - Framework support for software implementation is mature
  - Support tools remain incomplete
  - Extension to distributed and hierarchical systems needs further development

- Not yet applied on a large project as a mainstream deliverable
  - Huge “not me first” problem — which leads to …
Commitment (Hard Won Lessons)

- This is **Big!**
  - Systems and software engineering *by their very nature* are broadly crosscutting disciplines
  - Therefore, you can’t do this in “baby steps”

- Commitment is essential, but **Difficult!**
  - Without institutional backing, not much will happen
  - This requires careful planning, patient persuasion, and mutual trust

- The **Dilemma:**
  - Find a way to get there incrementally
Progress?

- 10+ years and counting
  - Nascent architectural concepts and state analysis methodology first clearly articulated in 1998
    - Derived from work in early and mid ‘90s
  - Initial support from progressive managers was vital
    - Took several years to develop and mature
    - Needed to nurture a core group of cognoscenti
    - Had to tolerate inevitable false starts

- Broad backing finally within reach

- A rocky road, to say the least
Where To from Here?

- Time will tell
  - Gradual ramping up anticipated
    - Capturing in SysML profile and other formalisms (Alloy, OWL…)
    - Putting institutional infrastructure in place
    - Additional pilot efforts under way
  - Continuing outreach (papers, courses, etc.)
  - Looking at potential standardization

- 10 more years?!
Final Thoughts

Murphy is alive and well
- Progress has been about half as fast as even “pessimistic” predictions

Worth it? Absolutely!
- Modeling discipline pays generous dividends that you might never anticipate
- The model-driven effort to be philosophical, formal, and unrelentingly principled on engineering fundamentals has been an eye-opening education