A Principled Approach to the Specification of System Architectures for Space Missions

Mark L. McKelvin, Jr., Robert Castillo, Kevin Bonanne, Michael Bonnici, Brian Cox, Corrina Gibson, Juan P. Leon, Jose Gomez-Mustafa, Alejandro Jimenez
Jet Propulsion Laboratory, California Institute of Technology

Azad Madni
University of Southern California

AIAA Space 2015 Conference, August 31 – September 2, 2015
Pasadena Convention Center, Pasadena, CA

Copyright 2015, California Institute of Technology. Government sponsorship acknowledged.

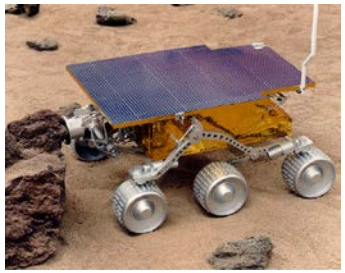
Outline

- Challenges in Practice
- Key Principles of Approach
- Modeling Flight System Architectures
- Tools
- Conclusion

Challenges in Systems Engineering Practice

Increasing complexity of component and people interactions

- Traditional system development is driven by loosely coupled documents (document-centric)
- Transforming mission requirements to implementation is ad-hoc
- Managing engineering design information and design changes is challenging
- Increases risk of meeting project cost and schedule



Mars Pathfinder

- 850 interface signals
- 4 instruments

Mars Exploration Rover (MER)

- 1750 interface signals
- 9 instruments

Mars Science Laboratory (MSL)

- 2350 interface signals
- 10 instruments

MSL 2020

- ~2400 interface signals
- 8-10 instruments

Increasing complexity in interface management as demand for more functionality increases

Need rigorous design methods and supporting tools to enable specification, design, and analysis of flight system architecture early in the development lifecycle.

System Architecture

Developing a system architecture is a system engineering function that:

- Translates mission needs and objectives into a technical solution
- Constrains the design space
- Enable trade-studies of alternatives
- Support “make-buy” decisions

Artifacts

A system architecture is a collection of artifacts that communicate system characteristics such as behavior, structure and qualities

Models

Key artifacts include models:

- Graphical FFBD, DoDAF, AADL, SysML
- Analytical: mathematical equations
- Prototypes: wind tunnel, mockups

Views

Artifacts are organized by views - a view communicates a certain aspect via artifacts

Requirements

- Describes problem to be solved
- Typically captured in textual format
- Admits many solutions

Architecture

- Captures system structure
- Exposes high-level system properties
- Facilitates trade-off studies

Implementation

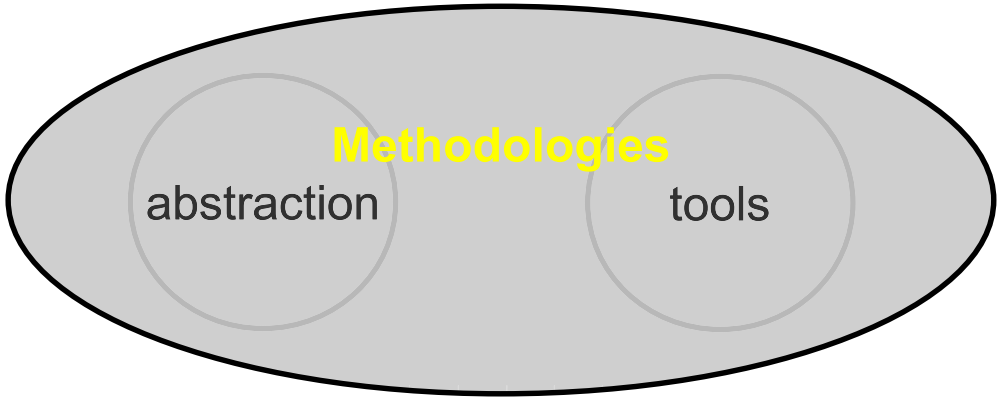
- Refines architecture
- Costly to change and verify
- Realized solution

General Principles to Managing Complexity

- **Abstraction:** Eliminates unnecessary details with respect to the goal at hand

- **Decomposition**
  - Break system development into semi-independent parts ("divide-and-conquer")
  - Separation of concerns (i.e. "*what*" vs. "*how*"; *computation* vs. *communication*)

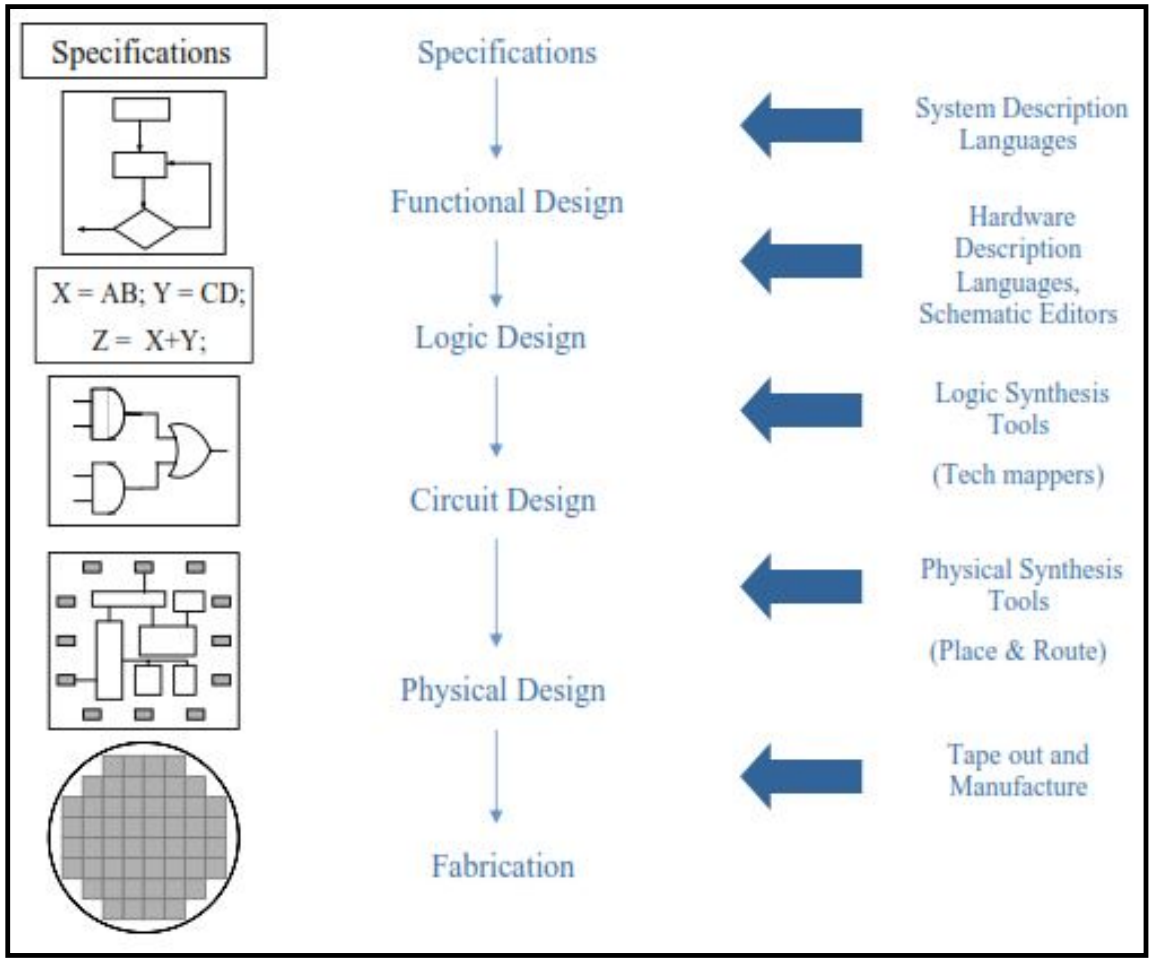
- **Construction**
  - "Artificially" constrain the design space
  - Define refinement steps from high-level abstraction to final implementation



Essential to any design methodology is ***modeling***!

Source: A. Sangiovanni-Vincentelli, Corsi e Ricorsi: The EDA Story, IEEE Solid State Circuits Magazine, 2010

Example: Generic VLSI Design Flow

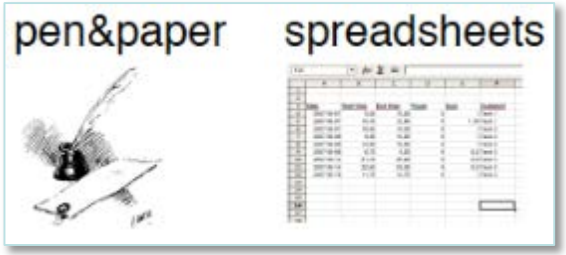


In addition,

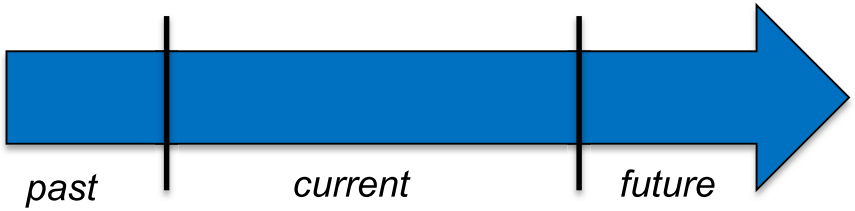
- **Back-annotate to provide predictability; leads to IP libraries**
- **Balance multiple design metrics (i.e. area, timing, power, signal integrity, reliability)**
- **Explore the design space to achieve balance between complexity and performance**
- **Utilizes IP blocks for design reuse and performance**

This structured design flow has enabled automatic synthesis and optimization techniques which have increased productivity and improved the quality of VLSI, system-on-chip, and embedded system designs.

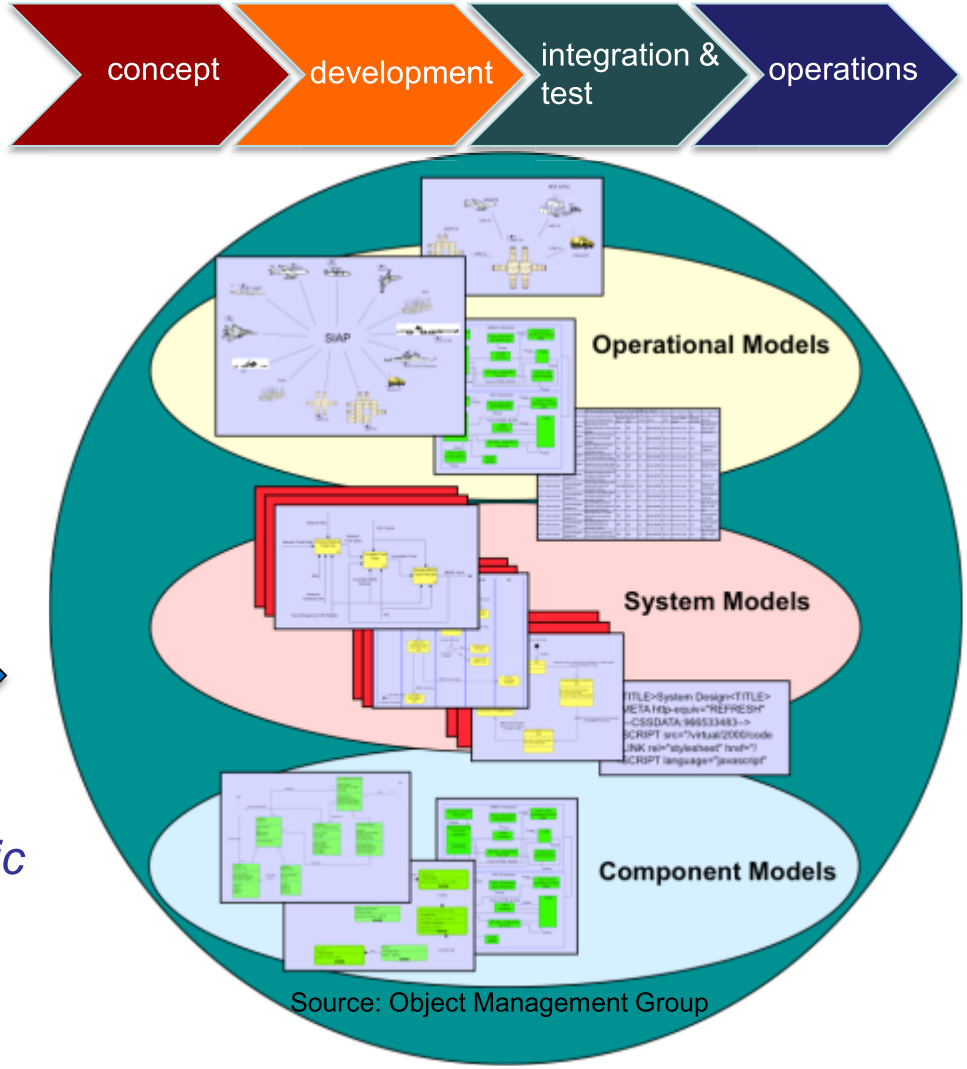
Model-Based System Engineering



Document-centric



Model-centric



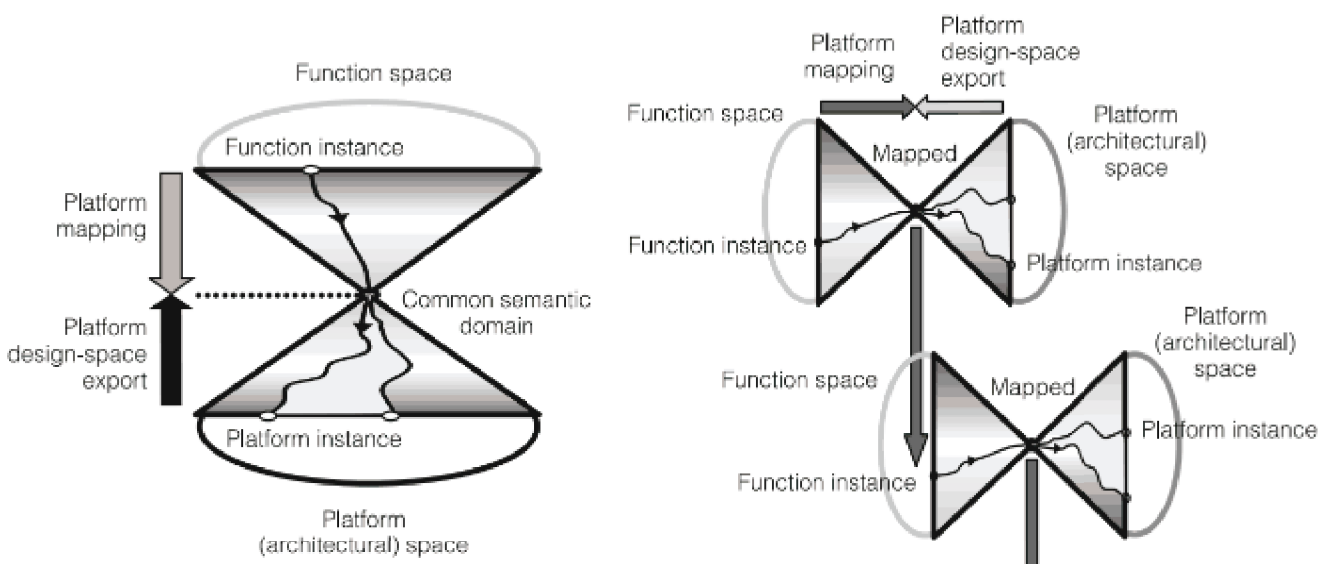
Source: Object Management Group

Shift in system engineering towards integrated models

Platform-Based Design

Key Principles of Platform-Based Design:

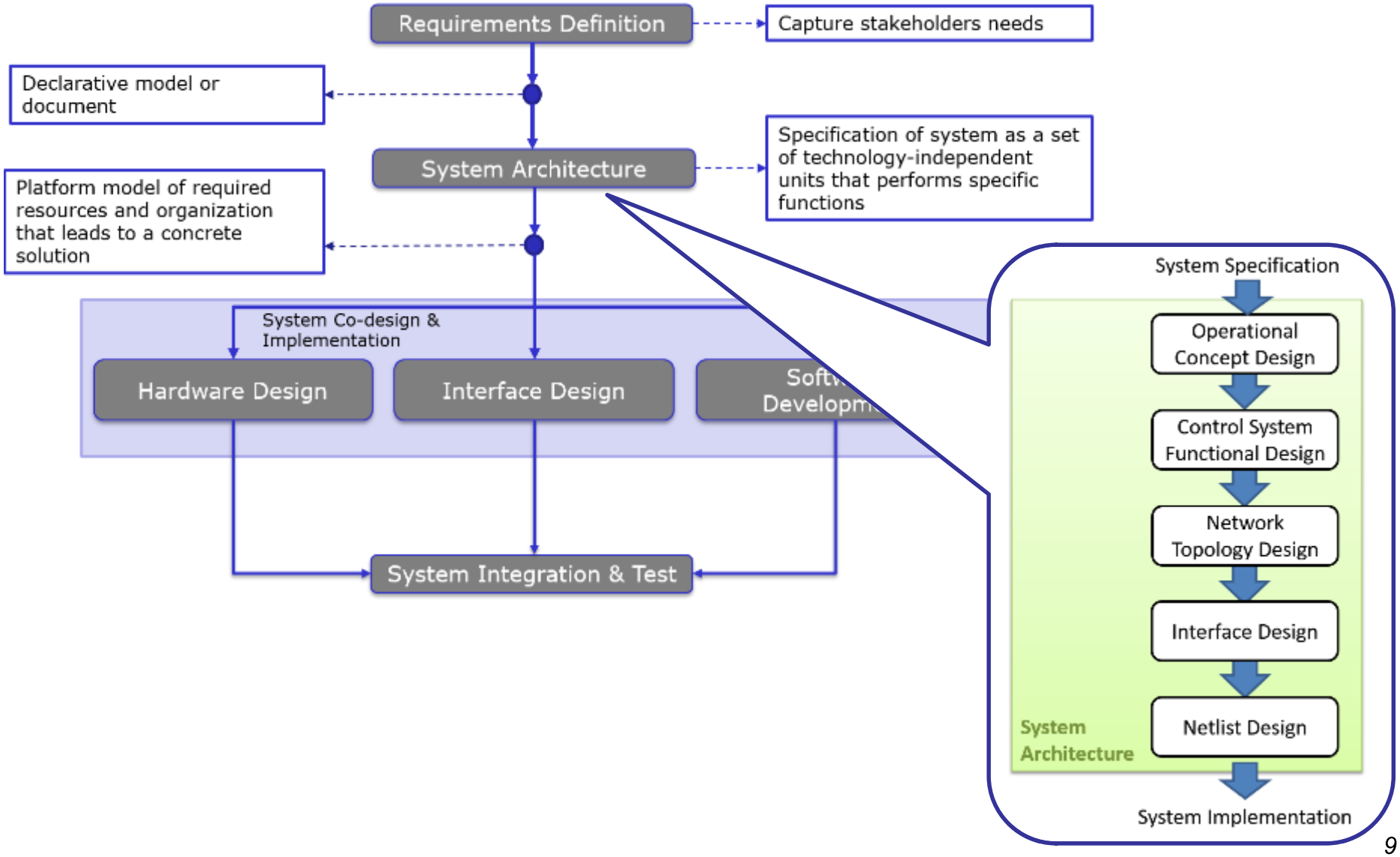
- A “meet-in-the-middle” design process where successive refinements of specification meet abstractions of potential implementations
- Provides a mechanism for identifying critical hand-off points in design chain
- Provides a structured method for design reuse at all levels of abstraction



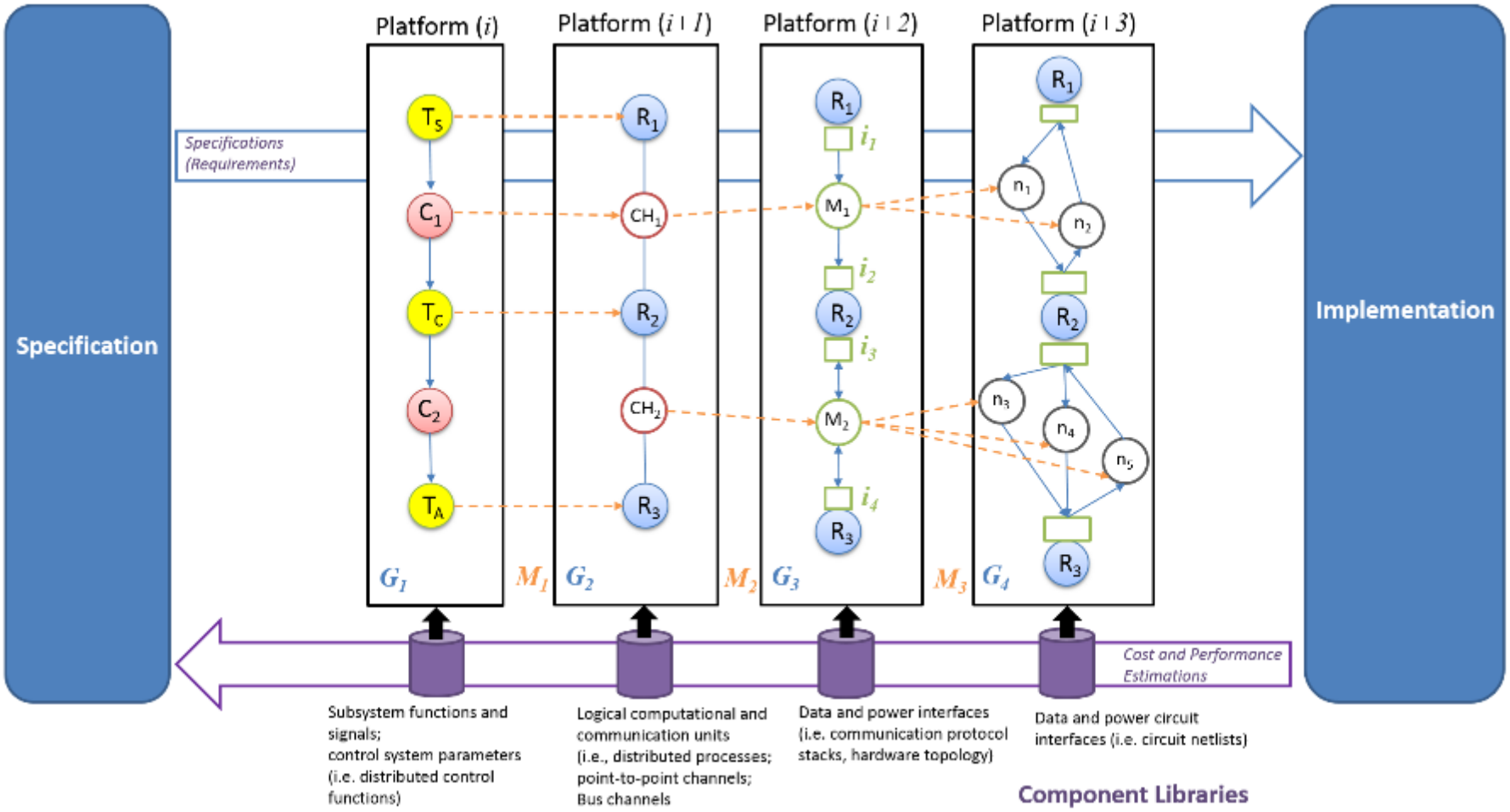
Fractal nature of design

Source: Sangiovanni-Vincentelli, A. "Is a unified methodology for system-level design possible?." *Design & Test of Computers, IEEE* 25.4 (2008): 346-357

Proposed Design Flow for Space System Architectures



Abstract Syntax



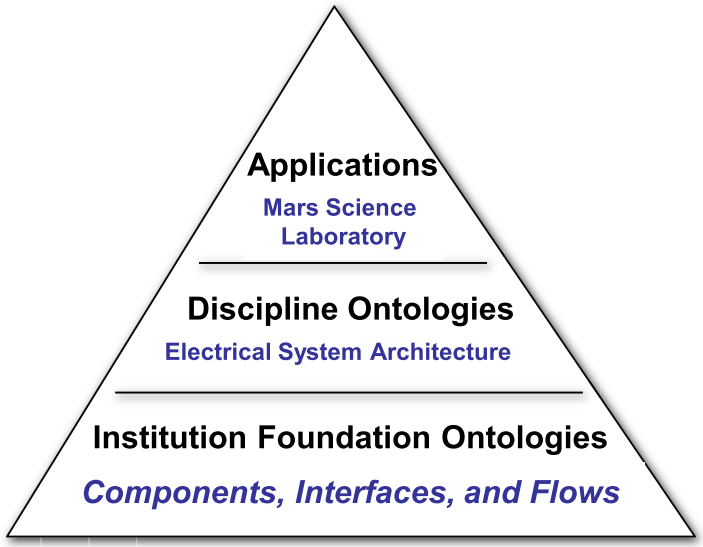
Capturing Semantics with Ontologies

Ontology

- An explicit, formal specification of knowledge in a domain

Consists of:

- A ***vocabulary*** relevant to the domain
- Intended ***meaning*** of the vocabulary
- Assertions, or ***axioms,*** that constrain interpretation of terms
- Logical inference statements, or ***rules***, that are drawn from assertions



Example Set of Ontological Concepts

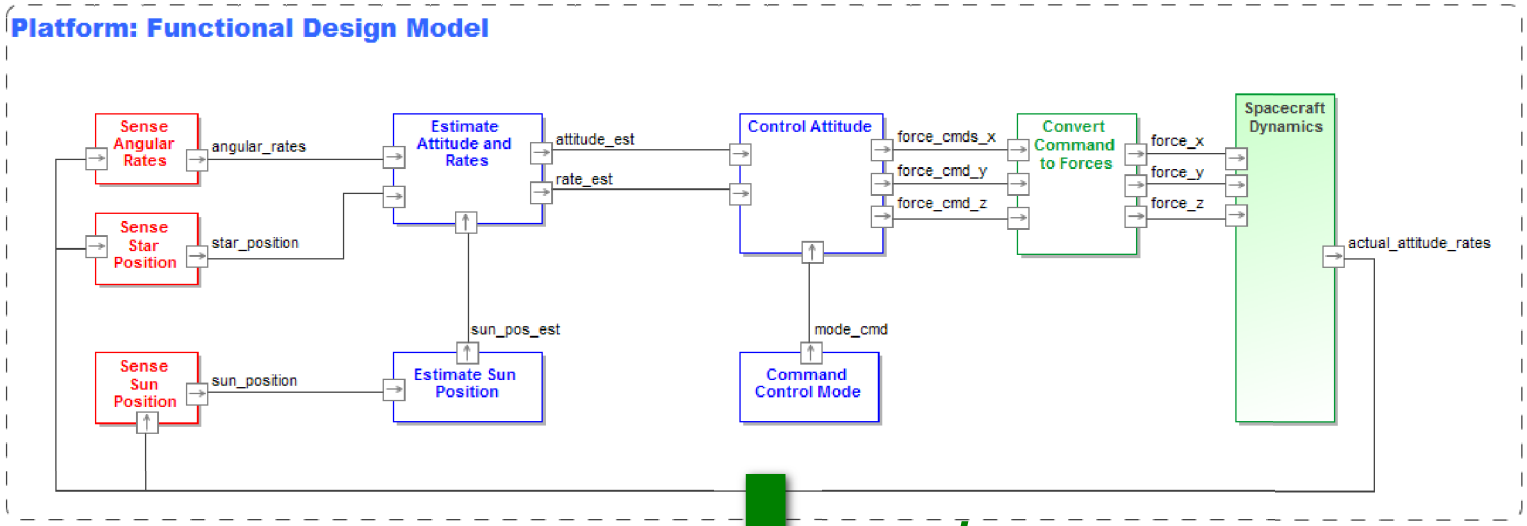
Represents logical entities of the *logical platform*

Logical Channel
Logical Component
Logical Signal
Logical Interface
presents
isType
Force
Heat
Data Signal
Power Signal
Hydraulic
transfers
Mechanical Interface
Thermal Interface
Data Interface
Power Interface
Pressure Interface
joins
2..*
1..*
characterizes
estimated quantity
application identifier
process

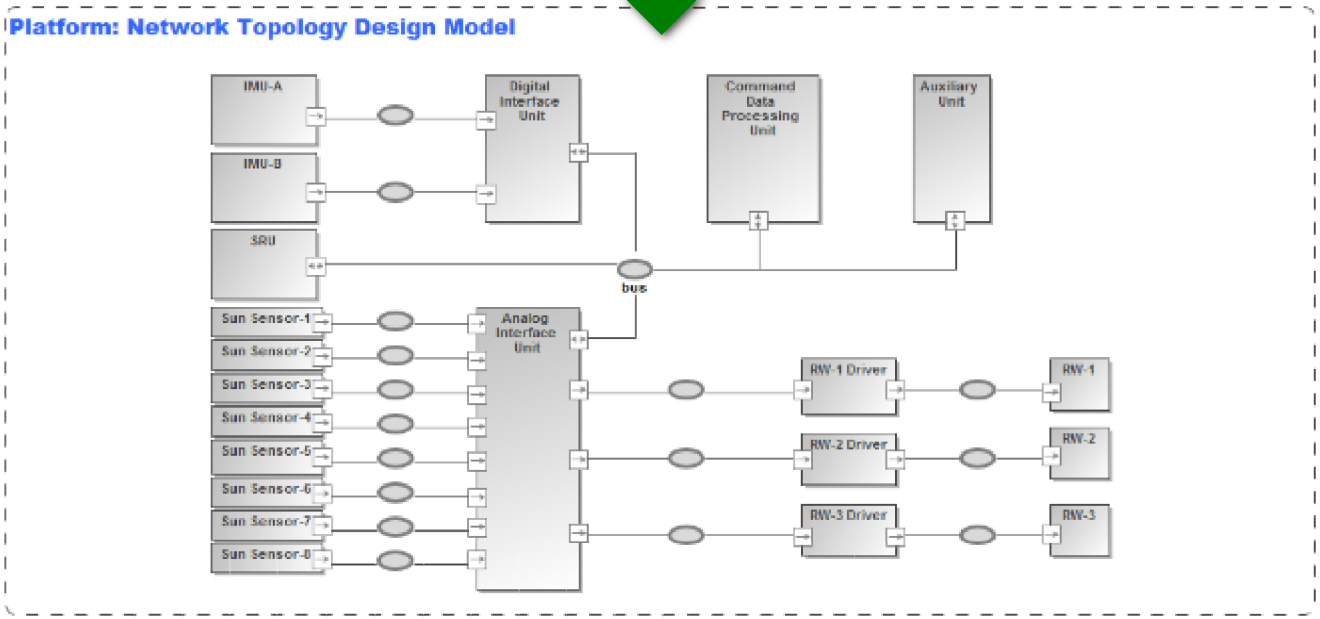
Attributes label entities with parameters and characteristics

A set of ontologies is used to capture abstract syntax rules and different aspects of the flight system architecture. Our models are annotated with ontologies.

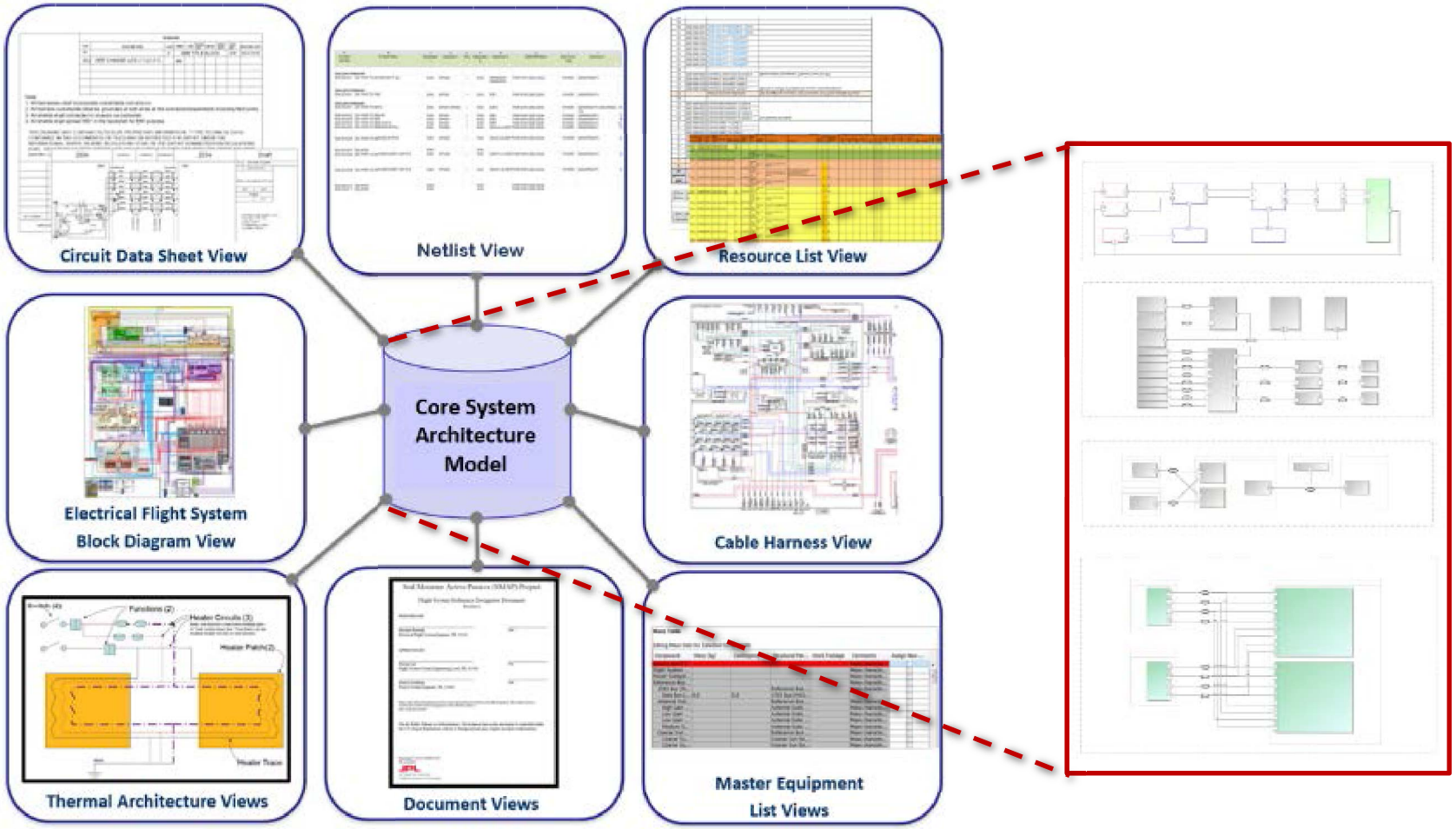
Examples of Platform Models in SysML



mapping

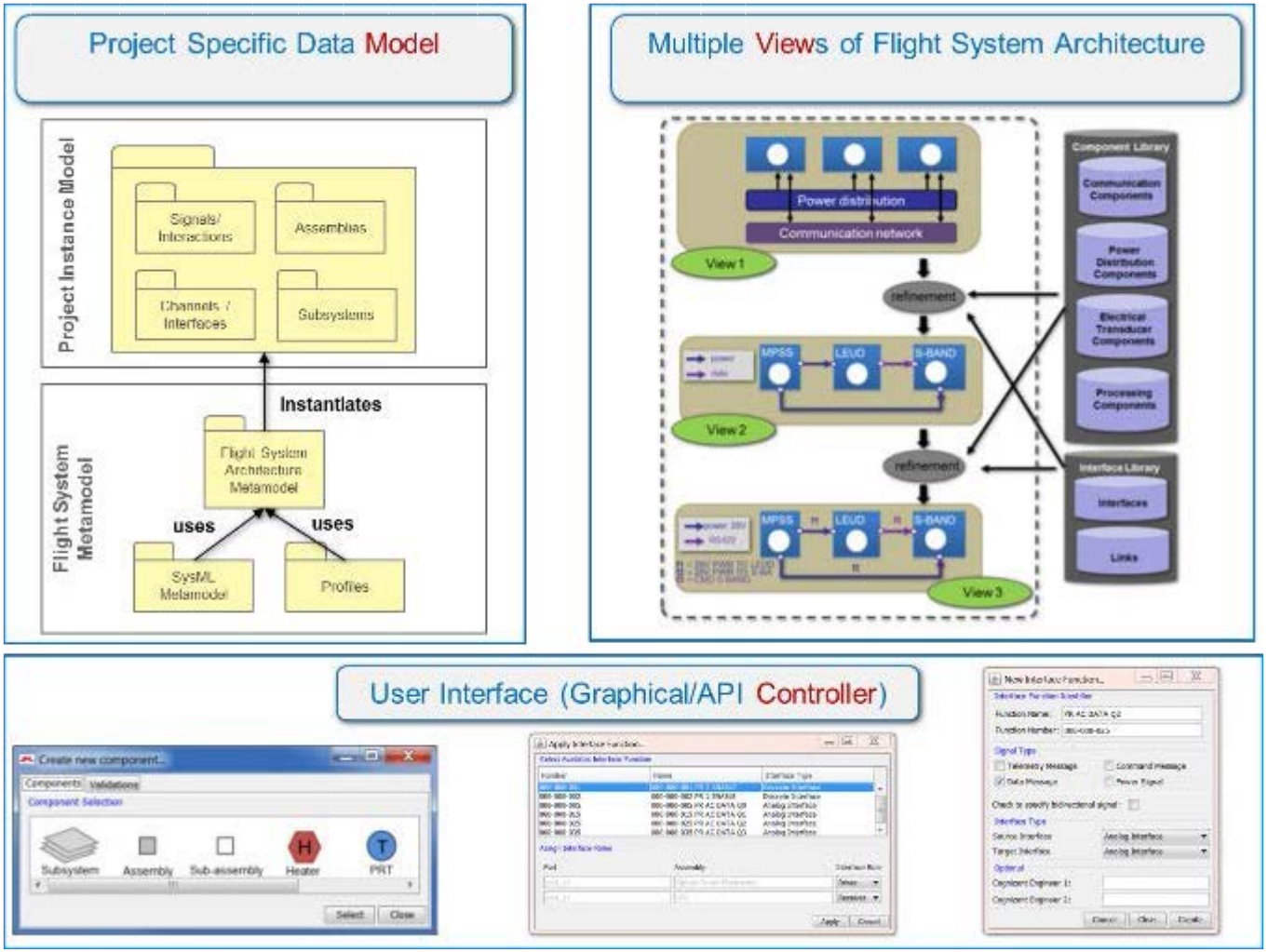


Common Views and Artifacts



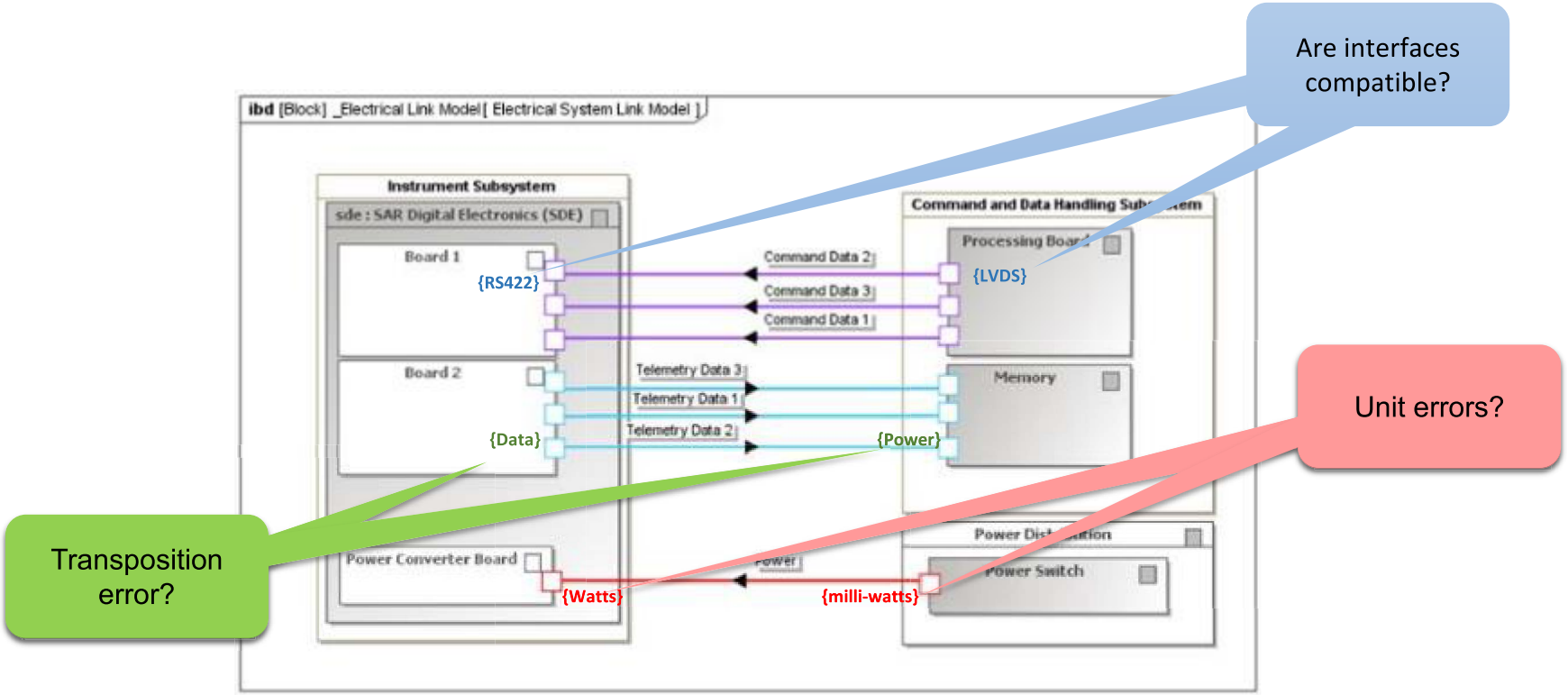
Core system model is the set of system architecture platform models annotated with ontologies → enables semantic data integration and consistency across views, artifacts, and tools. The views are projections of the core system model from which artifacts may be generated by custom tools.

Tools: Custom Plugin for MagicDraw + SysML



The tool also manages consistency of data that is created, read, updated, and deleted through the user interface. Managing information consistency is done in real-time on the back-end of the tool.

Tools: Model Composition Error Checking and Static Semantic Analysis

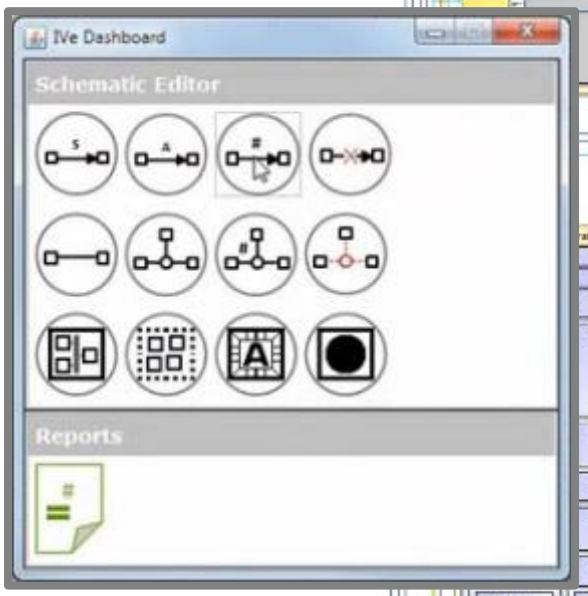


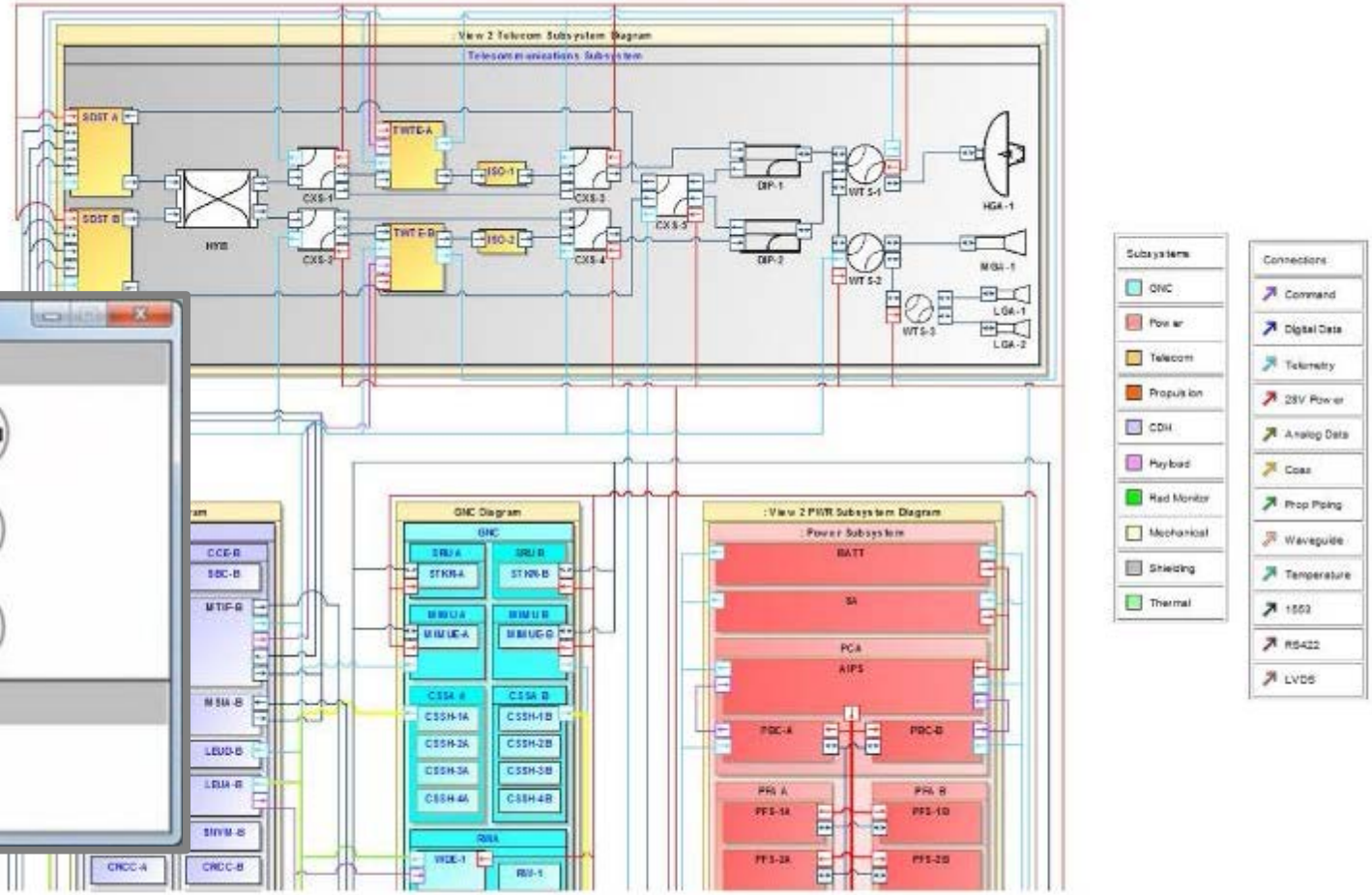
- Detect interfacing errors
- Check model completeness (or incompleteness)
- Minimize manual annotations by using static inference and correct-by-construction
- Improve communication between teams by augmenting semantic information

Tools: Schematic Editor Tool for Diagramming

Flight System Configuration

Custom Schematic Editing Tools





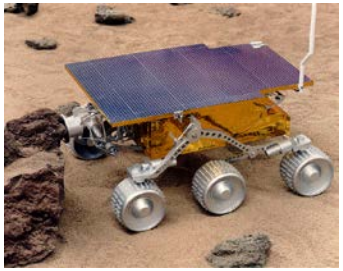
Schematic editing tools provide context and ease of use by system engineers.

Key Benefits

- Reduces ambiguous specifications of system architecture
- Partitions models along key articulation points in the design process which allows for capturing explicit design decisions, assumptions, and constraints
- Improves consistency of information that describes a system
- Enhances traceability of system architecture through lifecycle process phases
- Enables semantic design data and tool integration
- Type checking and inference allows for effective static analysis to catch common semantic and structural errors during model composition

Additional Slides

Complexity of Electrical System Interfaces



Mars Pathfinder

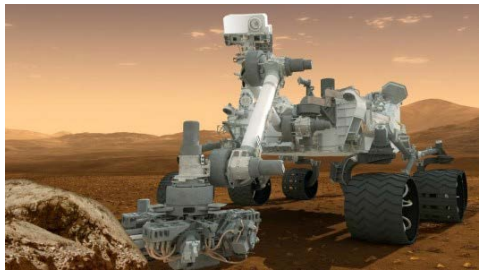
- 850 interface signals
- 4 instruments

Mars Exploration Rover (MER)

- 1750 interface signals
- 9 instruments

Mars Science Laboratory (MSL)

- 2350 interface signals
- 10 instruments



MSL 2020

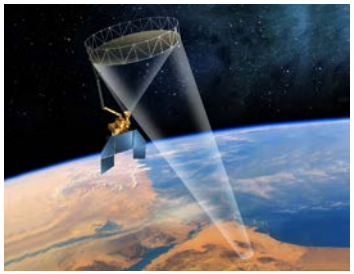
- ~2400 interface signals
- 8-10 instruments

Increasing complexity in interface management as demand for more functionality increases

Functional decomposition and allocation to physical components drive interface requirements and subsequent wiring and cabling design.

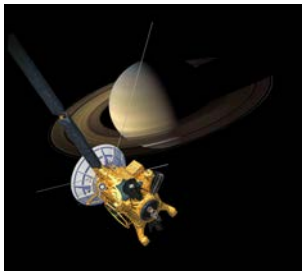
- 1000 interface signals
- 2 instruments

SMAP



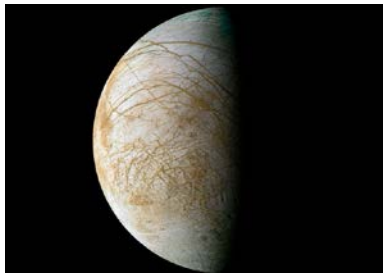
- 2500 interface signals
- 15 instruments

Cassini



- ~3000 interface signals

Europa



System Engineering Problem Areas

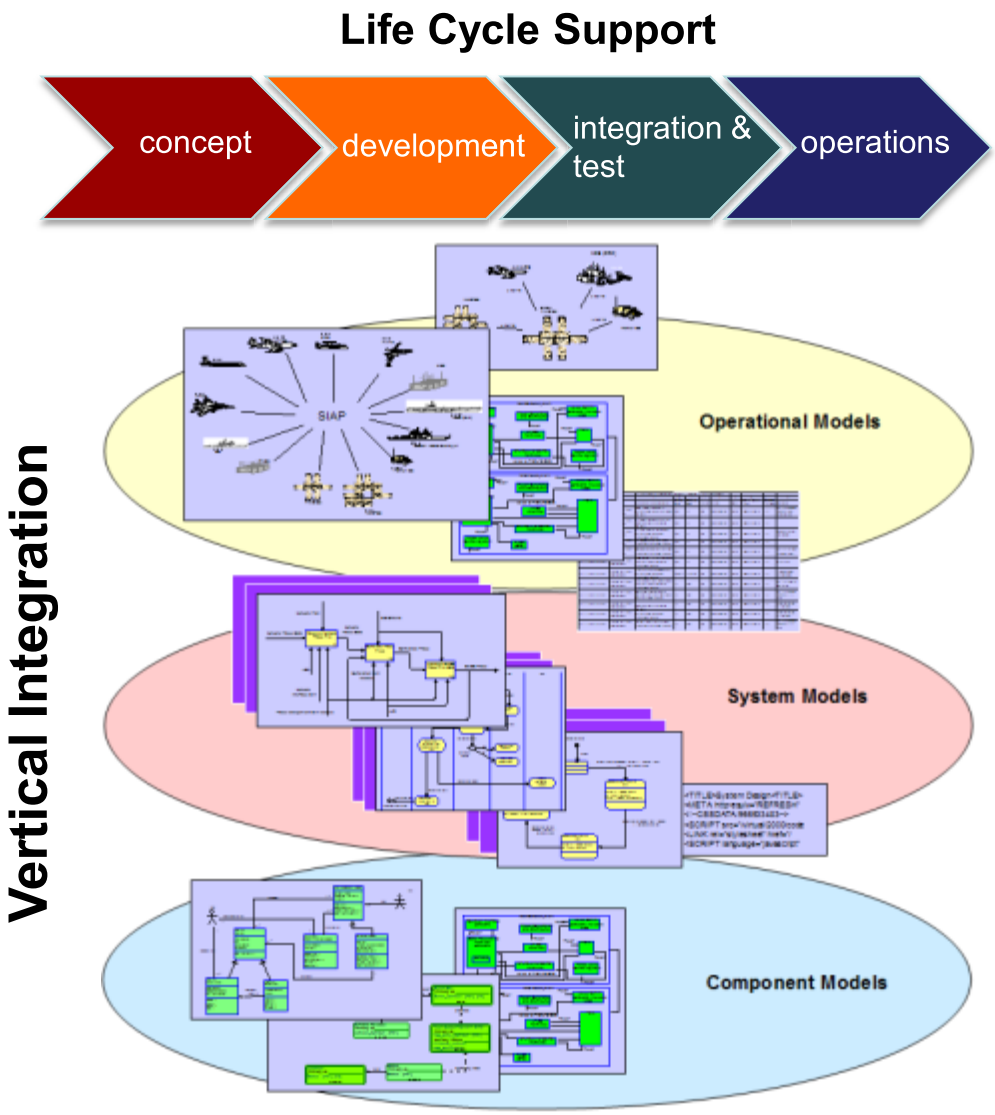
1. **Mission complexity is growing faster than our ability to manage it**
   …increasing mission risk from inadequate specification & incomplete verification

2. **System design emerges from the pieces, not from an architecture**
   …resulting in systems which are brittle, difficult to test, and complex and expensive to operate.

3. **Knowledge and investment are lost at project lifecycle phase boundaries**
   …increasing development cost and risk of late discovery of design problems.

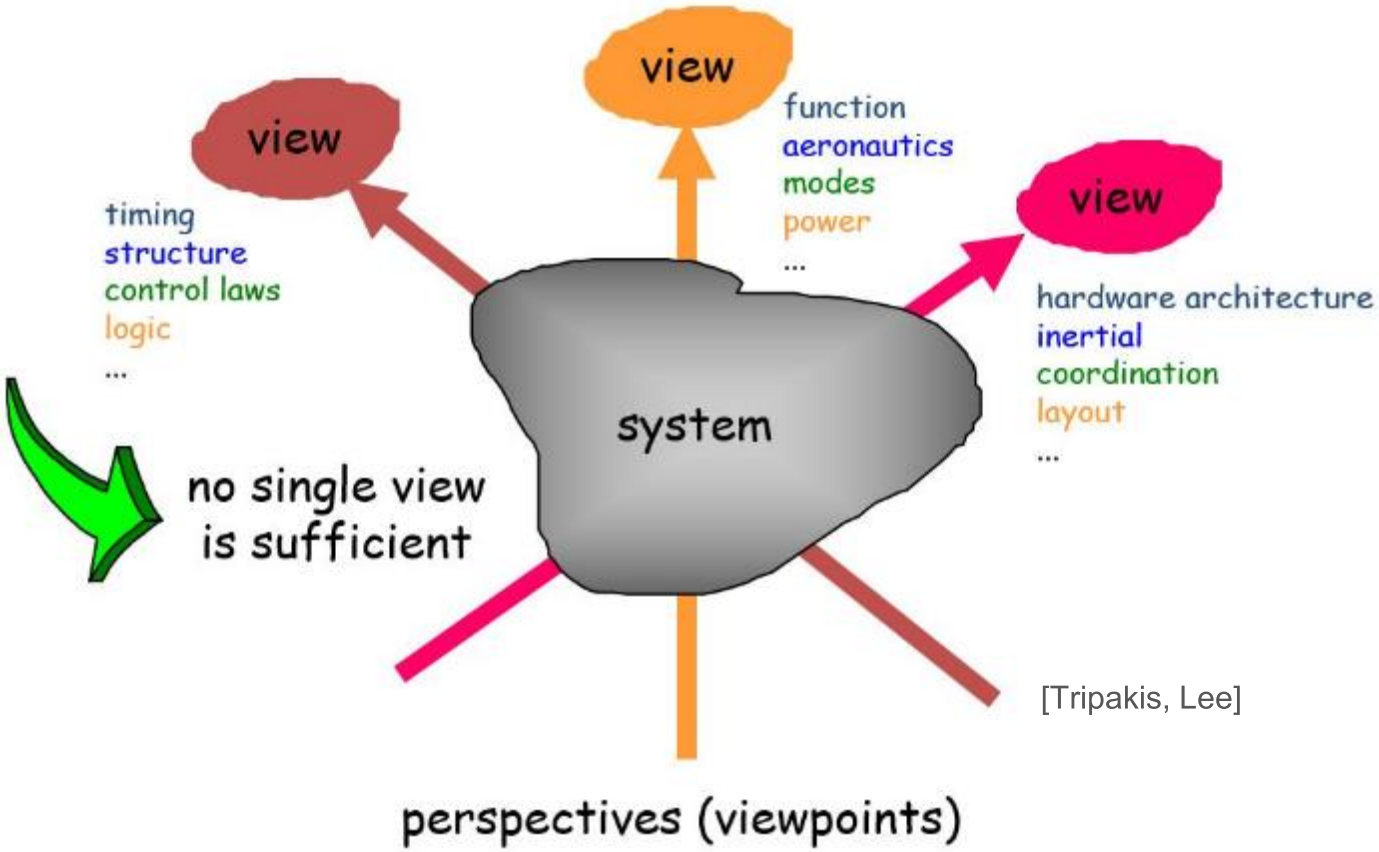
4. **Knowledge and investment are lost between projects**
   …increasing cost and risk; damping the potential for true product lines

5. **Technical and programmatic sides of projects are poorly coupled**
   …hampering effective project decision-making; increasing development risk.



Source: Object Management Group

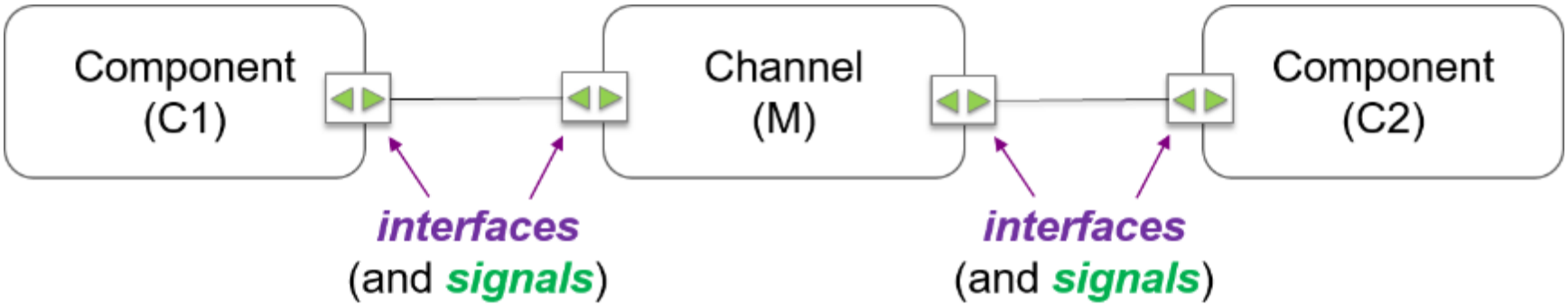
Multi-View Modeling



[Tripakis, Lee]

Complex system ⇒ many design teams ⇒ many viewpoints ⇒ many perspectives ⇒ many models = views

Component-Based Design



Concept	Description	Examples
signal	represents discrete messages and flows	command message, power, current
component	an entity that encapsulates behavior, produces and consumes signals	power converter, assembly, circuit, reaction wheel
interface	a point of interaction between a component and its external environment	message port, RS422 circuit, serial data interface
channel	logical or physical medium for communication abstractions	communication medium, wire, signal path

Key principles: strongly encapsulated design entities (components) with rigorous interface specifications ⇒ reusable, replaceable (modular), minimal dependencies between components

Platform-Based Design

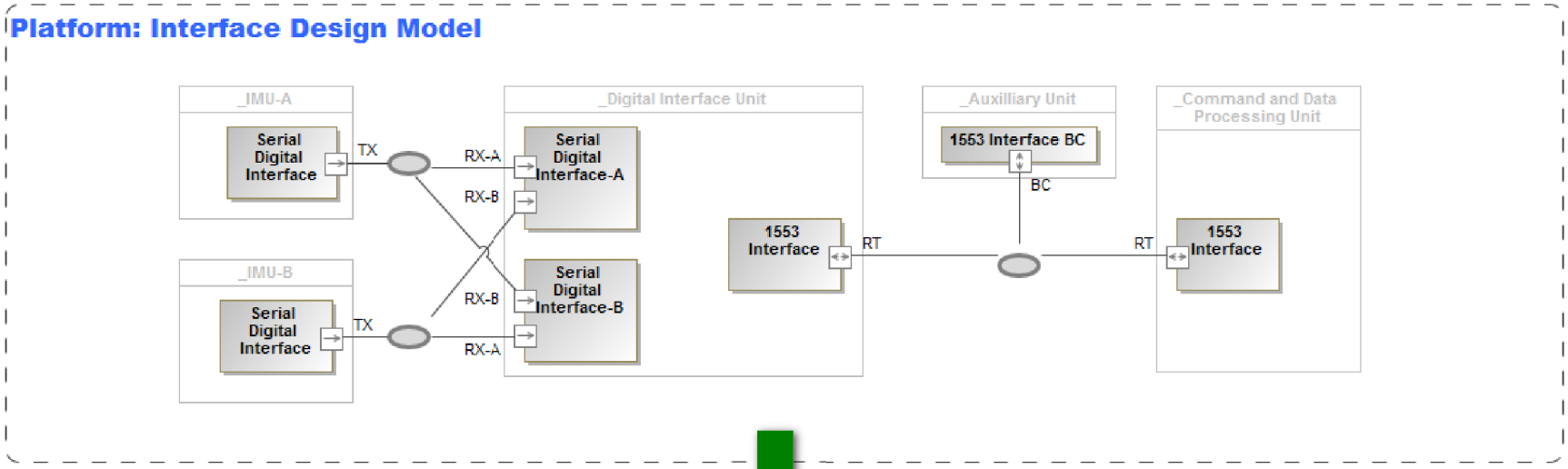
Definitions:

- A *platform* is a collection (library) of components that can be assembled to define an abstraction layer
  - Hides unnecessary details
  - Exposes only relevant parameters for next design step
- A *platform instance* is a set of components selected from the library and whose parameters are set

Observations:

- A platform is a parameterization of the space of possible solutions.
- Not all elements in the library are pre-existing components. Some may be “place holders" to indicate the flexibility of “customizing" a part of the design ⇒ we do not have a complete characterization of the element since its performance parameters depend upon a lower level of abstraction.
- Different situations will employ different intermediate platforms
- Critical step is defining intermediate platforms to support predictability (abstraction to facilitate higher-level optimizations) and verifiability (ability to ensure correctness)
- Emphasizes reuse and abstraction

Examples of Platform Models in SysML

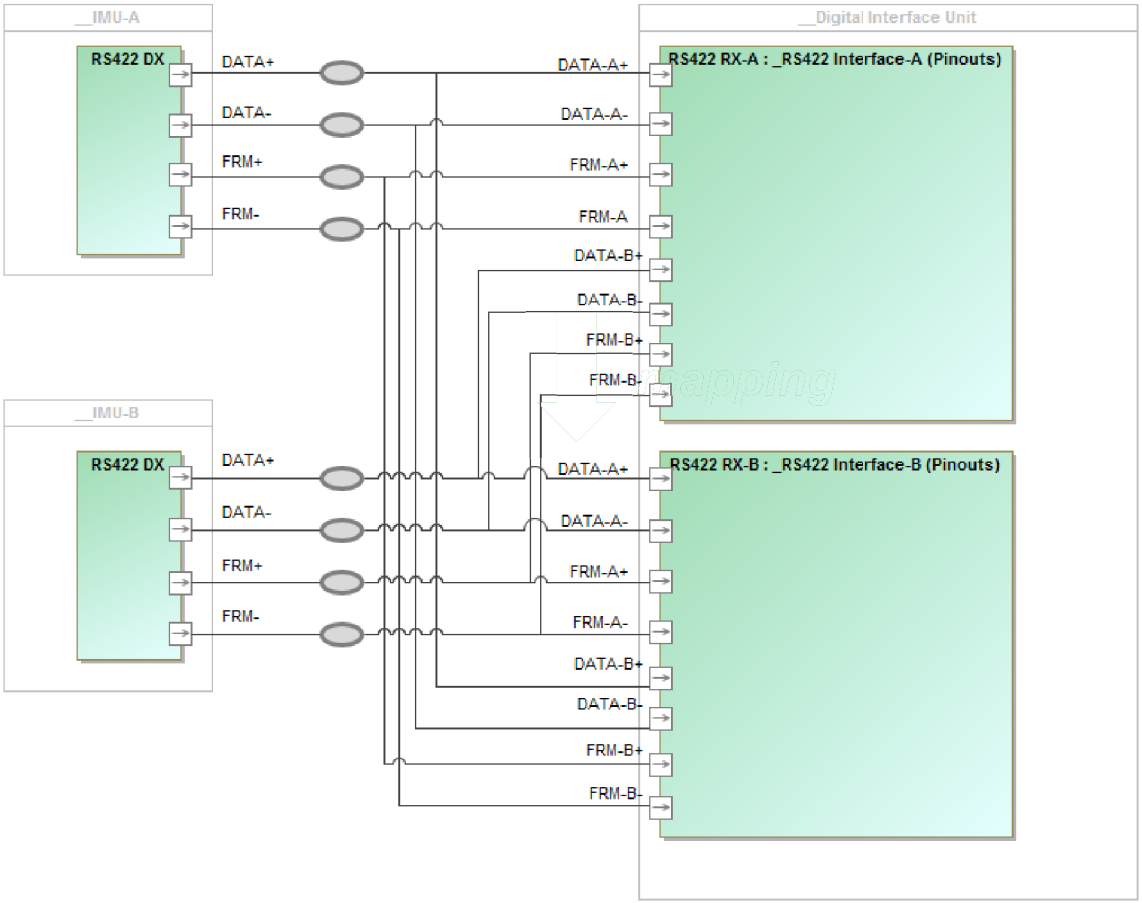


mapping

Platform: Circuit Netlist Design Model

Examples of Platform Models in SysML

Platform: Circuit Netlist Design Model



Views

Definitions:

- A *view* is a set of attributes that describe aspects of a system
  ⇒ *system* is a set of attributes

- A set of views is said to be *consistent* if there exists a system that generates these views
  - Semantically: systems and views are sets of attributes defined by some ontology
  - Syntactically: a view can have any notation that captures the attributes (i.e. table, block diagram, document, etc.)

- A view of a system V_i, is a projection (an abstraction function) π_i, of the system S. Thus, a set of N views are consistent w.r.t. a set of projections if $V_i = \pi_i(S), for\ i = 1 \ldots N$.