Speaker Meeting

June 11, 2019

International Council on Systems Engineering (INCOSE)
Los Angeles Chapter

Agenda

- 05:30 Networking & Introductions
- 06:00 Announcements
  Working Group Overview
- 06:30 "Systems Architecting,"
  Mark L. McKelvin, Ph. D.,
  President, INCOSE Los Angeles Chapter
- 07:30 Adjourn

Welcome!

Host and Remote Sites

Remote Site	Coordinator	Contact
Aerospace Corp./El Segundo (Host)	Deborah Cannon	deborah.a.cannon@aero.org
Antelope Valley/Palmdale	Dr. J. S. Shelley	J.Shelley@csulb.edu
CSU, Dominguez Hills	Dr. Antonia Boadi	aboadi@csudh.edu
Ricardo Control Point/Goleta	Paul Stowell	paul.stowell@control-pt.com
NGC/Redodo Beach	Deanna Regalbuto	deanna.regalbuto@ngc.com
Virtual Attendee	Deborah Cannon	deborah.a.cannon@aero.org

Thanks to all for registering in advance

Please contact Programs Director, Nazanin Sharifi (programs@incose-la.org) if you would like to host a remote site.

Virtual Networking

- Briefly introduce yourself (e.g., name, title, company)
  - Host site (The Aerospace Corporation, El Segundo)
  - INCOSE Los Angeles remote sites
  - Other INCOSE chapters
  - Other virtual attendees
- Announcement of job openings

Welcome New Members!

Name	Organization
David Utley	CEB Metasystems, Inc.
Kamran Ossia	Fresca Medical
Behnam Afsharpoya	Dassault Systemes
Tim Bode	LinQuest Corporation
Marc Carithers	LinQuest Corporation

Announcements

- Chapter Events
  - Board of Directors meeting, next meeting is Friday, June 14
  - Next speaker meeting, Dr. Antonia Boadi, August 13
  - Other local chapter events?
- Chapter Conferences
  - Sept 13-15, Western States Regional Conference @ LMU
    - More information: www.incose.org/wsrc2019 or Phyllis Marbach
  - March 19-21, Conference on Systems Engineering (CSER) 2020 @ Redondo Beach Crowne Plaza
    - Save the date; currently looking for volunteers
    - Contact: Eric Belle (eric.belle@incose.org)
- Around INCOSE: www.incose.org/
- Other events sponsored by INCOSE “West Coast”?

Volunteer Opportunities

- Treasurer
  - Manages finances, reimbursements
  - Represents Chapter on financial matters
- Remote site coordinator
  - Coordinates communication with remote sites for speaker meetings
- Networking coordinator
  - Plan networking events
  - Propose new methods to improve Chapter networking
- Membership records management support
  - Support the Membership Director in collecting attendance records and membership matters
- Conference committee members (WSRC, CESR)
  - Management
  - Technical

Working Group Spotlights

- Model-Based Systems Engineering Initiative
  - https://www.incose.org/incose-member-resources/working-groups/transformational/mbse-initiative
  - http://www.omgwiki.org/MBSE/doku.php

- Tool Integration and Model Lifecycle Management Working Group
  - https://www.incose.org/incose-member-resources/working-groups/transformational/tools-integration-interoperability

Speaker: Dr. Mark McKelvin

- **President** of INCOSE Los Angeles Chapter
- **The Aerospace Corporation**, emphasis on systems modeling, analysis, and infusion of model-based techniques and tools into practice
- **Lecturer at University of Southern California**, System Architecting and Engineering Program (since 2015)
- UC Berkeley, Electrical Engineering and Computer Sciences (Ph.D.)
  - Design, Modeling, and Analysis for embedded systems, electronic systems, cyber-physical systems, and fault tolerant automotive control systems
- Clark Atlanta University, Electrical Engineering (B.S.)
- Previous experience: JPL, General Motors R&D, Intel Corporation, HRL, Sandia National Lab, Army Research Lab, Army High Performance Research Center
- Key roles in space systems (previous experience): Fault Protection Engineer, Electrical Systems Engineer, Software and Systems Engineer

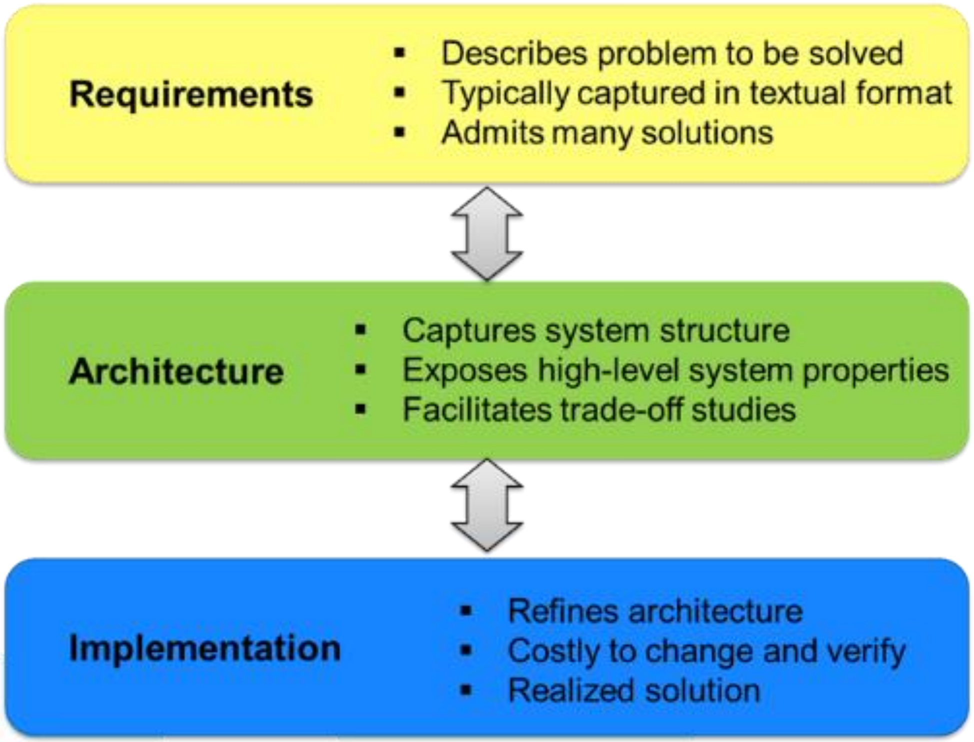
What is Architecture?

- It is the fundamental and unifying system structure defined in terms of system elements, interfaces, processes, constraints, and behaviors. [INCOSE]
- It is the structure of components, their relationships, and the principles and guidelines governing their design and evolution over time. [DoD]
- The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution. [ANSI/IEEE 1471-2000]

Architecting

- Architecting is the process for developing an architecture
  - Characterizing and selecting the problem
  - Generating and selecting an overall concept
  - Evaluating concepts for feasibility, or fitness for use

- ➢ Translates needs into a technical solution
- ➢ Constrains the design space
- ➢ Enable trade-studies of alternatives
- ➢ Support "make-buy" decisions



Used to help develop both requirements and design

Why do Architecting?

- A system is more than the sum of its parts
- Systems come to realization through interactions
- Complexity is a consequence of interactions

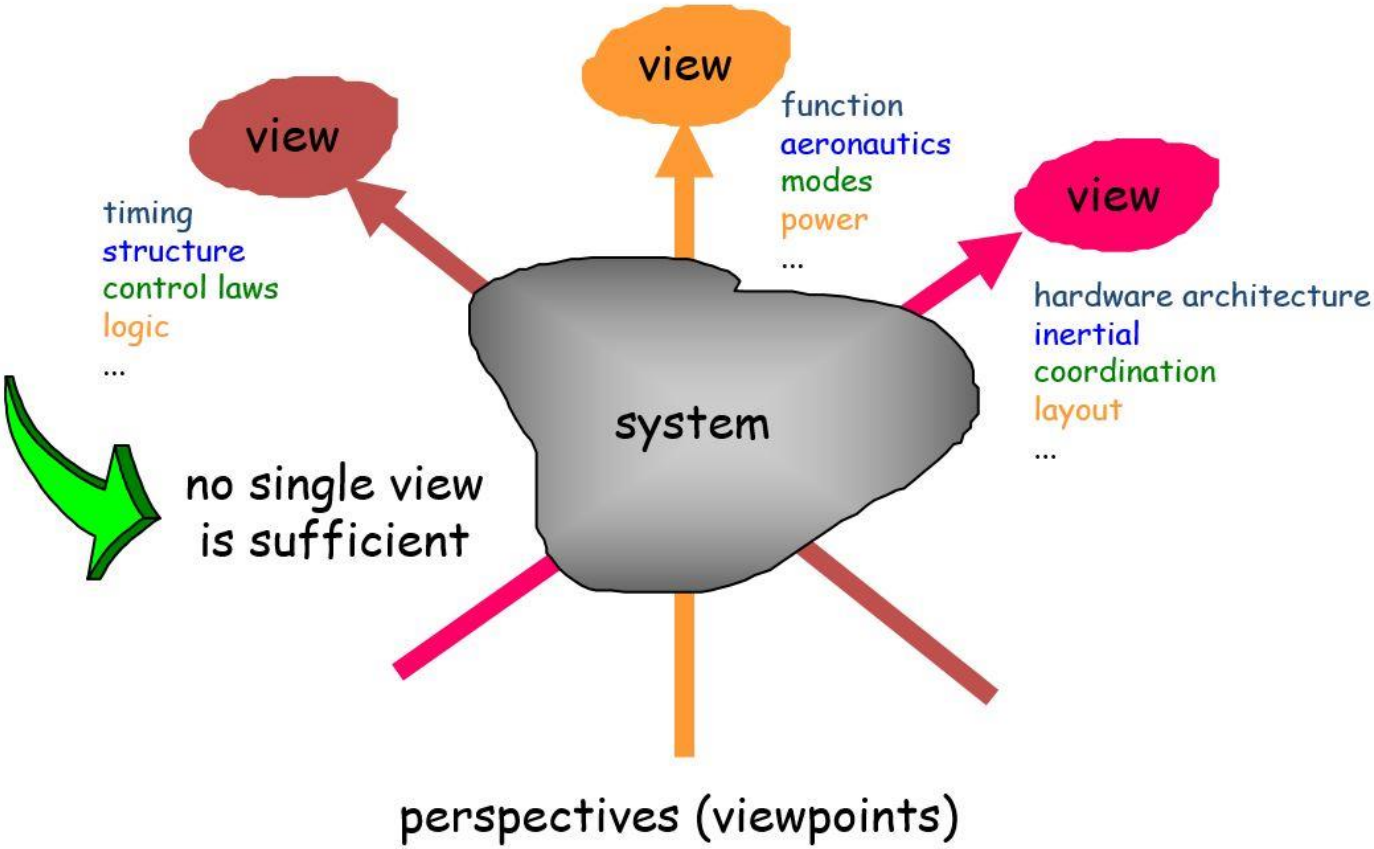
Systems → Interactions → Complexity

- Anything can seem complex if one does not understand it
- Complexity is not a system property that can be judged or regulated in absolute technical terms (e.g., what is the threshold for “too complex”)
- Complexity is a comparative measure of a system against those who must understand it

Complexity leads to misunderstanding

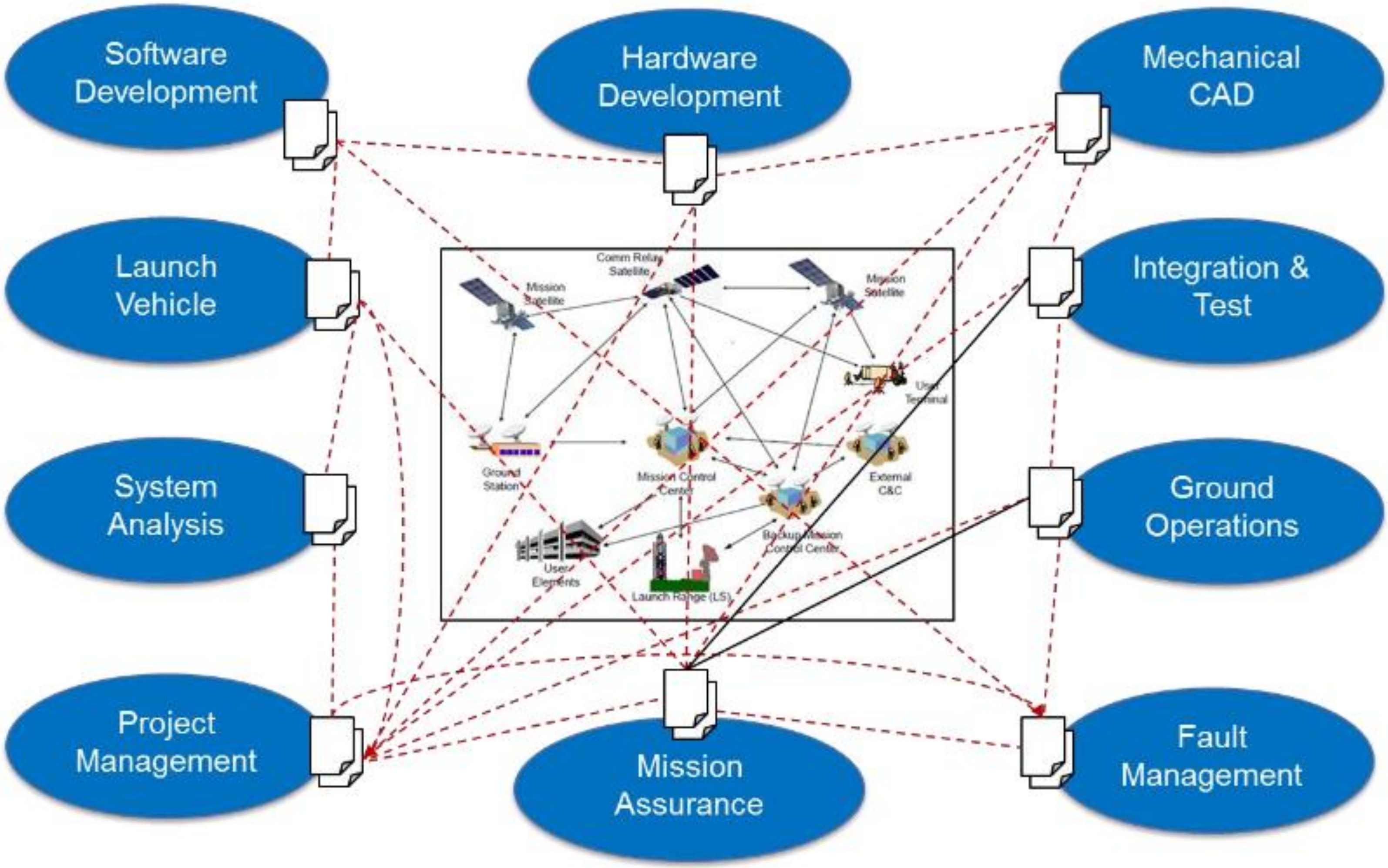
So, where do we seek understanding?

Understanding through Views



Complex system ➔ many teams ➔ many models = many views

Understanding through Documents



When Architecturing Doesn’t Happen…

Winchester Mystery House, San Jose, CA

Source: Library of Congress/Wikimedia Commons

- Doors leading to nowhere
- 2-inch high steps
- Windows overlooking other rooms
- Columns installed upside down
- Doors opened into walls
- Fully furnished decorated rooms walled-off
- Stairways leading to the ceiling

- Blueprints available? *None!*
- Mrs. Winchester never had a master set of blueprints, but did sketch out individual rooms on paper and even tablecloths (according to legend)!

Considerations

- Architectures are not unique
  - More than one architecture can satisfy the needs
  - Constraints by human biases, legacy, and available resources can lead to a single option
- Architectures provide enough detail to,
  - Describe properties of the solution to the problem
  - Describe the technical and organizational risks, impacts, and inter-dependencies
  - Confirm that a solution fulfills the functional, technical, and business requirements
- Architecture provides enough analysis and context to address all stakeholder concerns
- An architecture can be implemented (realized solution) in more than one way

General Methods for Architecting

Methods	Basis	Examples
Normative	Solution basis	Building codes, government regulations
Rational	Procedural basis	Data analysis, structured techniques, object-oriented, text-book
Participative	Stakeholder basis	Tiger teams, Brainstorming, Delphi sessions
Heuristic	Lessons learned	Rules of thumb from experience, qualitative judgment from examples

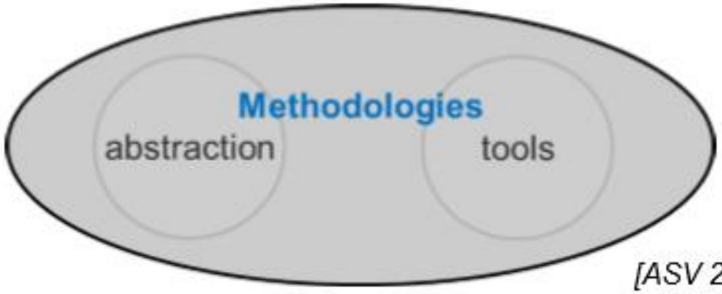
- Science-based, deductive methods:
  - Normative: hard rules are provided and success defined by compliance to rules
  - Rational: defined from objectives utilizing optimizations and formal techniques
- Art-based, inductive methods:
  - Participative: solution from group consensus
  - Heuristics: based on lessons learned and "soft" rules developed from experience

Architecting is a decision-making process, not diagrams or documents

An Approach to Architecting

- **Abstraction and decomposition**
  - *Eliminate unnecessary details with respect to the goal at hand*
  - *Break system development into semi-independent parts ("divide-and-conquer") and separation of concerns (i.e. "what" vs. "how", computation vs. communication)*
  - *Incremental refinement: include details while preserving properties*

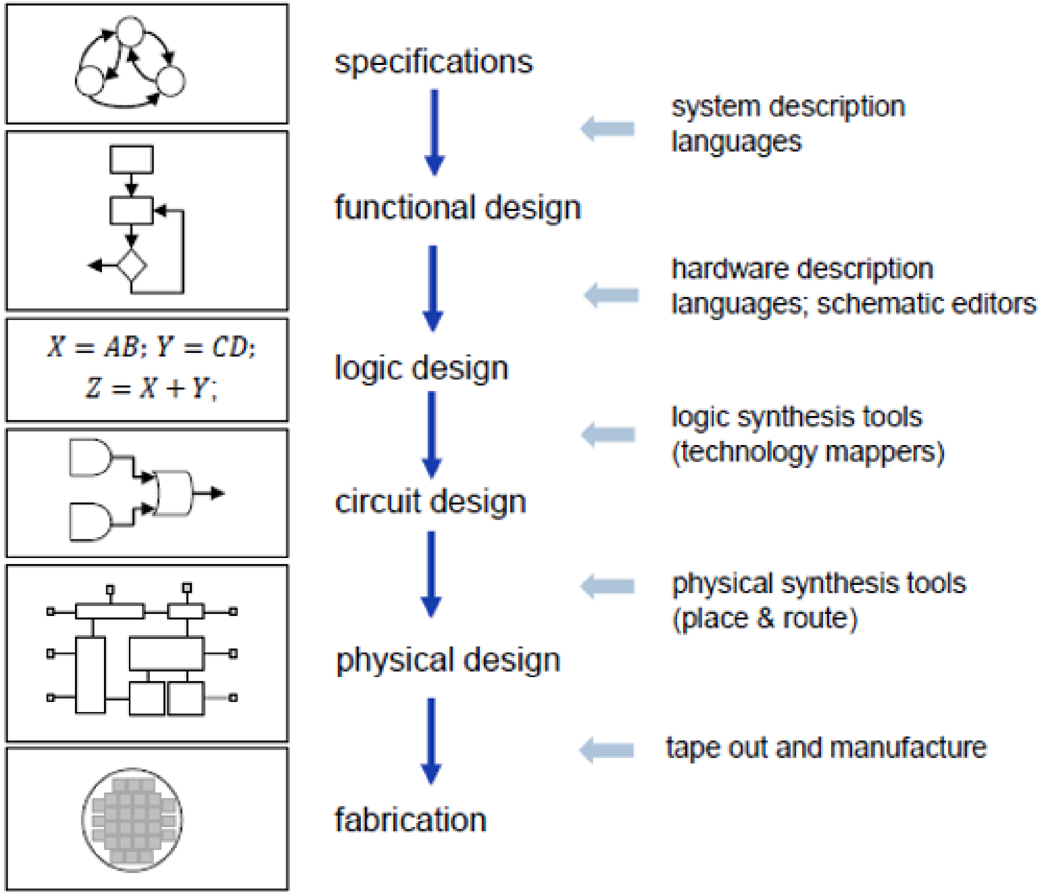
- **Construction**
  - *Constrain the design space and define transformations from high-level abstraction to final implementation*
  - *Defines verified, strongly encapsulated components with well-defined interfaces (enabling reuse)*



[ASV 2010]

Dealing with today's system design challenges requires more than just developing new tools. It requires understanding principles of design, necessary changes to design methodologies, and supply chain dynamics [ASV 2008]

Example: Architecting VLSI *Systems*



Decomposition:
break system development into semi-independent parts ("divide-and-conquer") and separation of concerns (i.e. "what" vs. "how", computation vs. communication)

Abstraction:
eliminate unnecessary details with respect to the goal at hand

Construction:
constrain the design space and define transformations from high-level abstraction to final implementation, supported by design automation tools

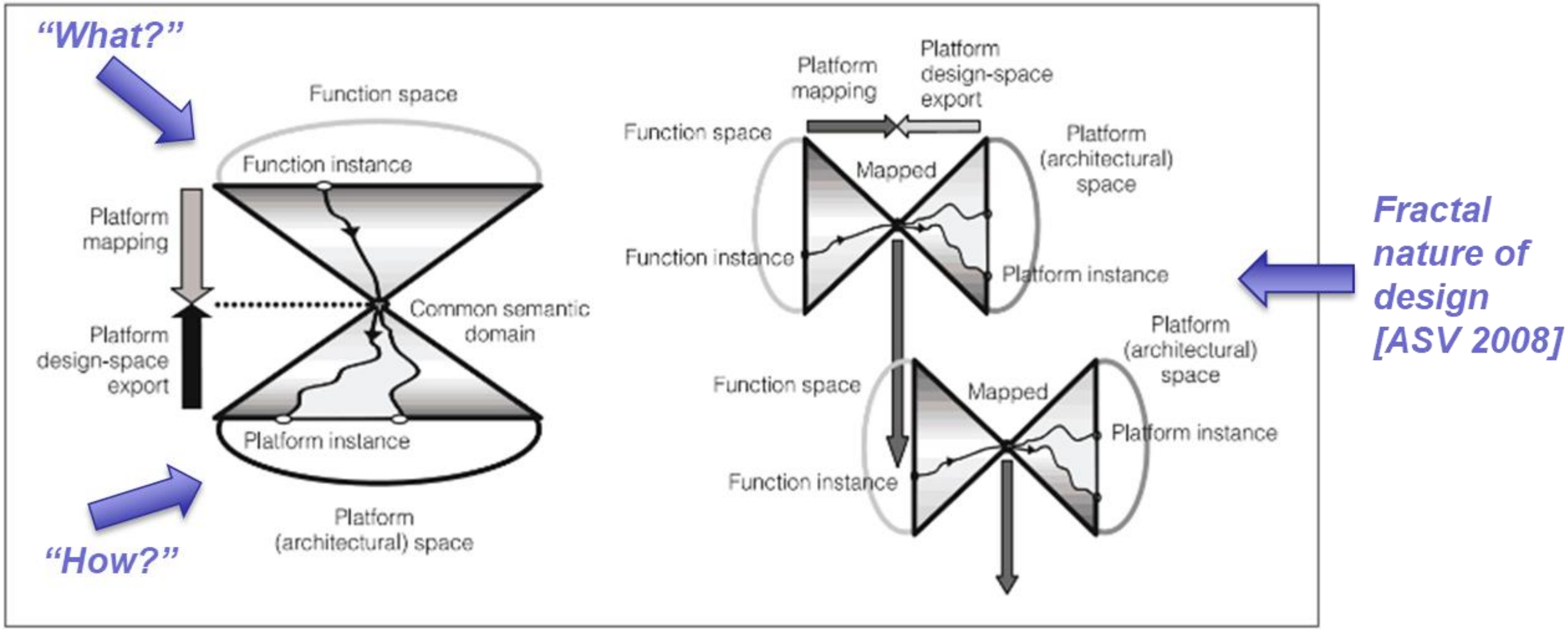
This methodology illustrates a structured means by which specification is transformed to implementation in electronic design automation; also applied in embedded software engineering, automotive systems, synthetic biology, and building automation

VLSI = Very Large Scale Integrated circuits

Platform-Based Design Approach

Key Principles of Platform-Based Design (PBD):

- 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



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

Example in Hardware Design

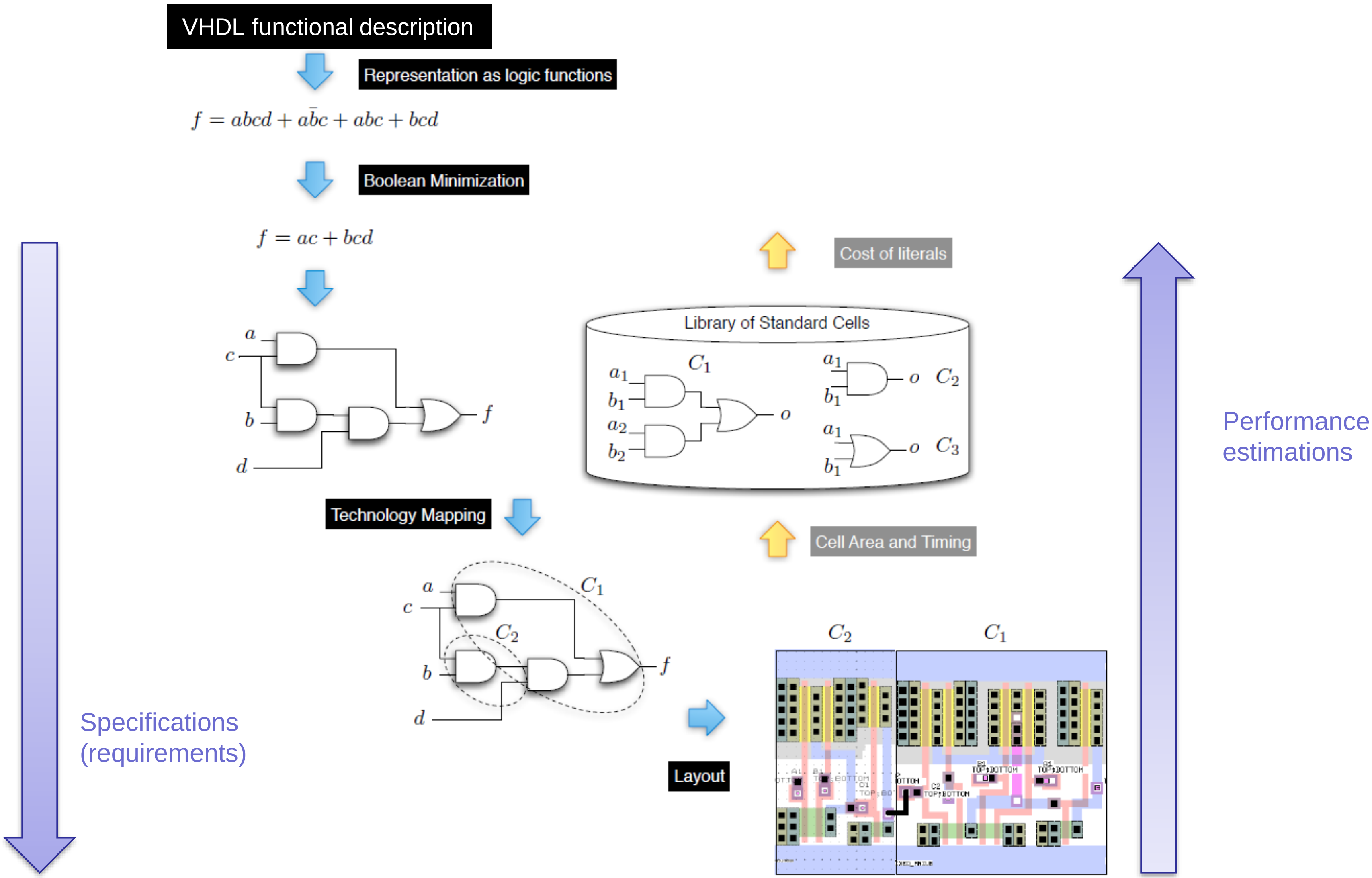
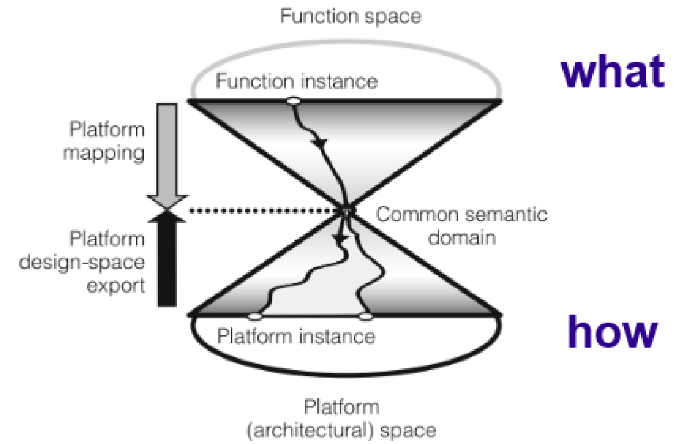


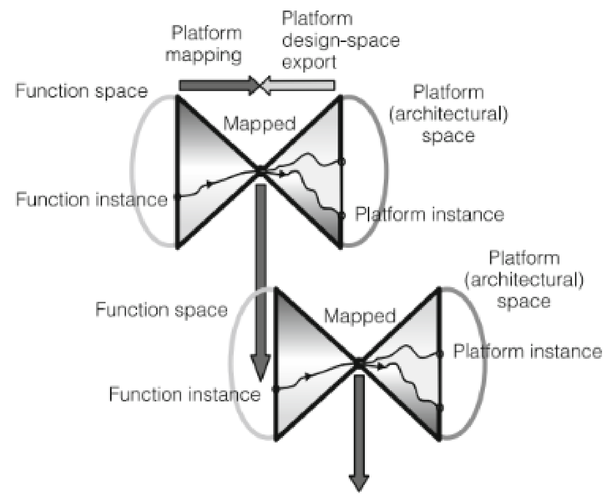
***Image Source**: Alberto Sangiovanni-Vincentelli*

Platform-Based Design Approach

- A meet-in-the-middle design method
  - *Platform: an abstraction layer that hides the details of several possible implementation refinements of the underlying layers*
- Function model
  - *Provides an abstraction of **what** the system is supposed to do*
- Architecture component model
  - *Provides an abstraction that describes **how** the function is realized*
- Mapping
  - *Process by which function and architecture meet*
  - *Propagates constraints from above to meet performance estimations from below*



successive refinements



[A. Sangiovanni-Vincentelli, "Is a unified methodology for system design possible?", 2008]

Component-Based Design Pattern

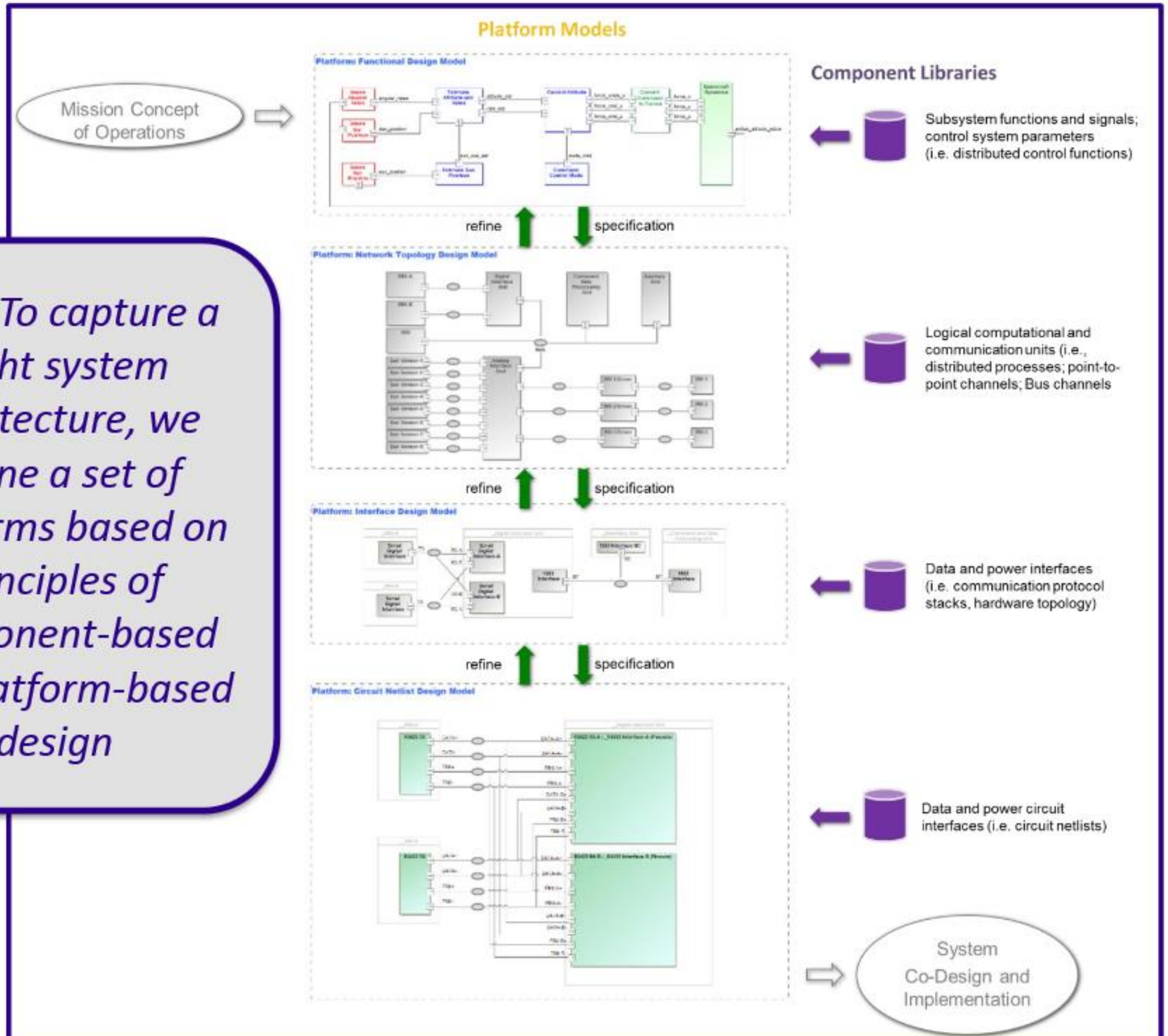
Abstract syntax

Abstract semantics

Concept	Description	Examples
signal	represents messages, flows, signal traces	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

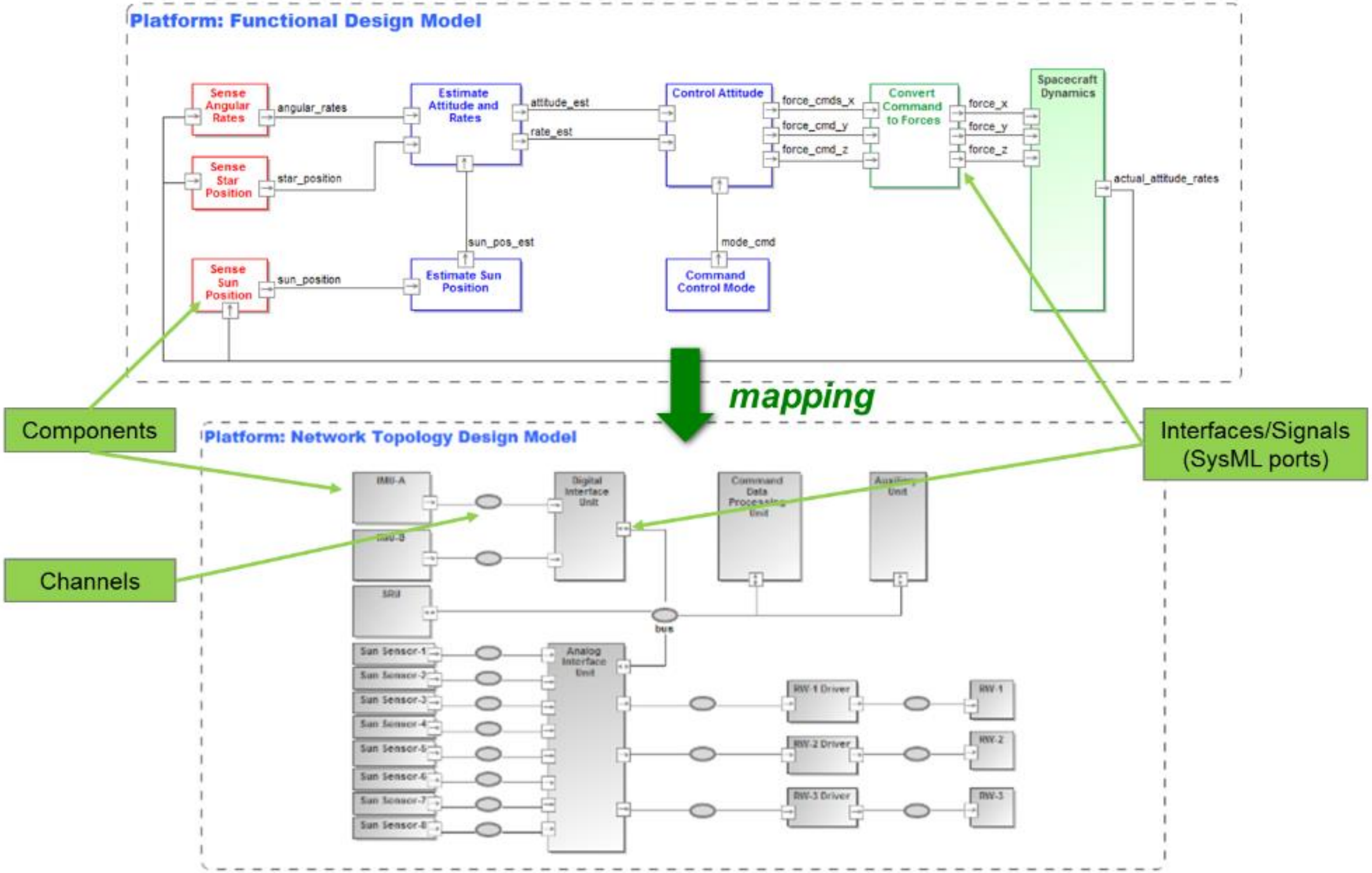
Key principles [component-based design]: strongly encapsulated design entities (components) with rigorous interface specifications ⇒ reusable, replaceable (modular), minimal dependencies between components, separation of communication and computation

Example Application: Platform Models for Spacecraft Interface Design



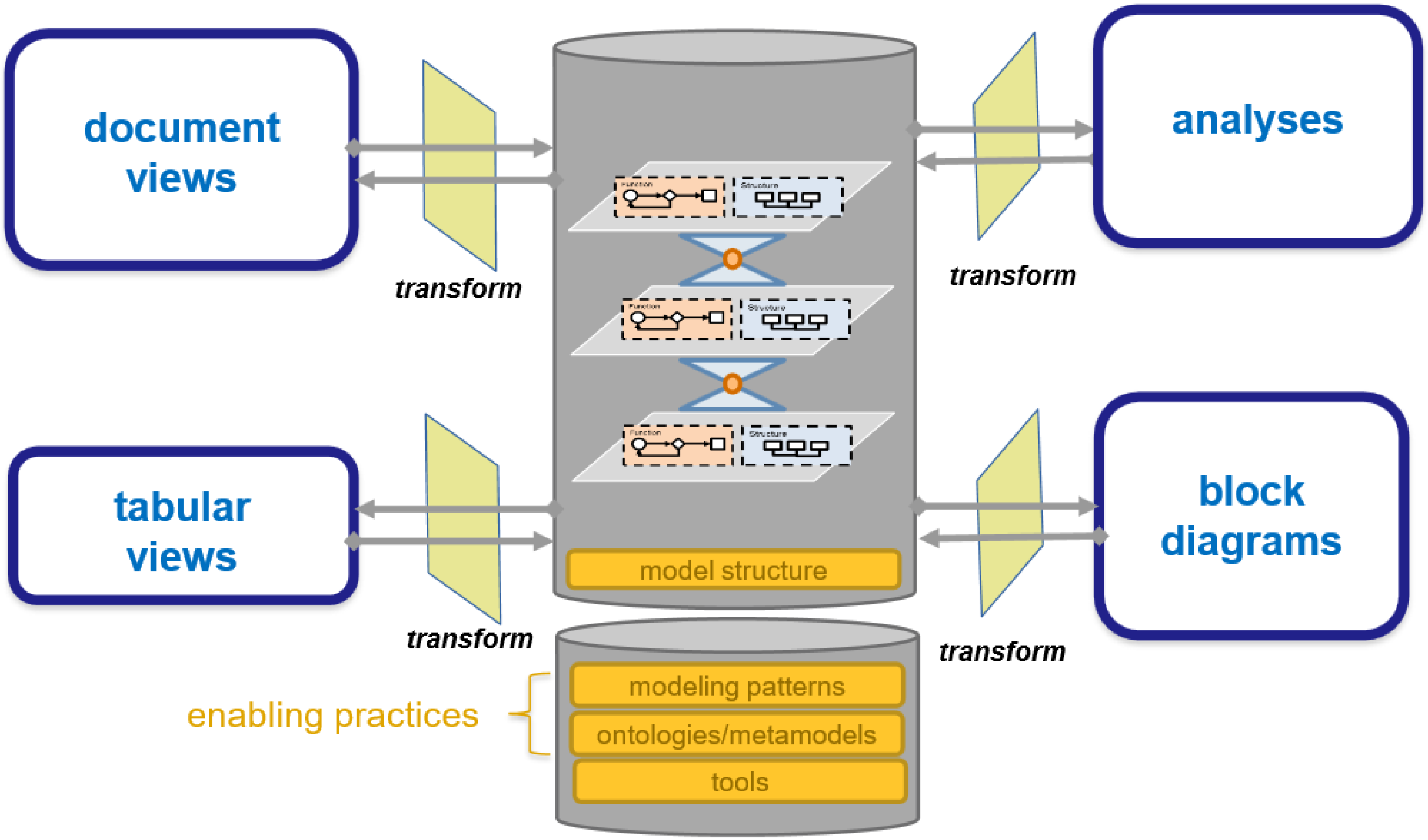
***Goal:** To capture a flight system architecture, we define a set of platforms based on principles of component-based and platform-based design*

Example Application: Platform Models for Spacecraft Interface Design



In this example, the Systems Modeling Language (SysML) was used to capture platform models

Integrating Data Through the System Architecture Model



The architecture model provides a framework - a structure where information is stored – for integrating data and import/export data to external tools for analysis

Application of Modeling, Analysis to System Architecting

- Separate areas of concern
  - *Function from component realization; computation from communication; data from view*
  - *Enables optimization and independent management of irrelevant features/functionality*
  - *Easier to read, understand, and communicate*
- Use existing data sources and workflows to establish levels of abstraction
- Use model elements that are intended to be used for a specific level of abstraction
- Favor composition over inheritance to maximize reuse
  - *Inheritance limits reuse due to parent/child dependency*
  - *Composition enables modularity, enhances reuse*
- Coupling between abstractions (platforms) is implemented through explicit mapping relationships to maximize reuse and modularity

Summary

- Modeling framework is constructed upon platform-based and component-based design principles to unify modeling, design, and analysis

- Benefits of approach:
  - *Provides structure through a unified system architecture*
  - *Partitions models along key articulation points in design process → enables explicit design decisions, assumptions, and constraints*
  - *Ensures consistency of information that characterizes a system*
  - *Enhances traceability*
  - *Enables semantic knowledge representation and analysis*
  - *Supports model transformations for external analyses and end user views*

References

- [ASV 2008] A. Sangiovanni-Vincentelli, "Is a Unified Methodology for System-Level Design Possible?," in *IEEE Design & Test of Computers*, vol. 25, no. 4, pp. 346-357, July-Aug. 2008.

- [ASV 2010] A. Sangiovanni-Vincentelli, "Corsi e Ricorsi: The EDA Story," in *IEEE Solid-State Circuits Magazine*, vol. 2, no. 3, pp. 6-25, Summer 2010.

- [McKelvin 2015] M. McKelvin, R. Castillo, K. Bonanne, M. Bonnici, B. Cox, C. Gibson, J. P. Leon, J. Gomez-Mustafa, A. Jimenez, and A. M. Madni. “A Principled Approach to the Specification of System Architectures for Space Missions”, 2015.