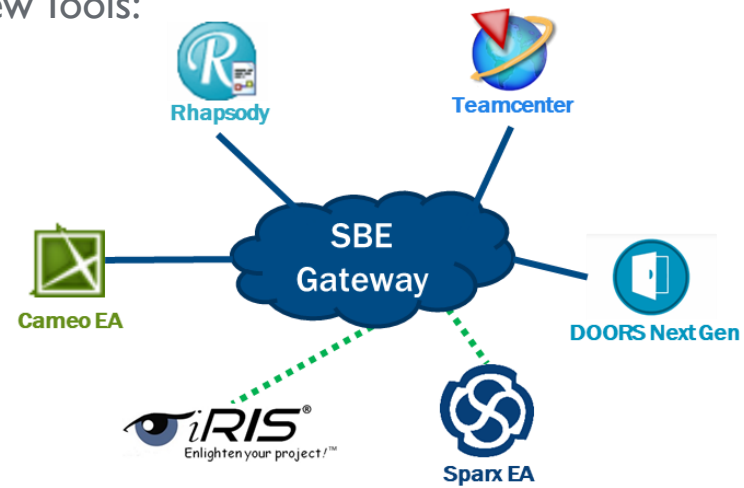
A Data Centric System Architecture Model Development Process Emphasizing Rapid Tempo and Quality

Chris Swickline
Heidi Jugovic
INCOSE International Symposium
Jun 25, 2022 - Jun 30, 2022

SAIC

Barriers to Building a Well-Formed Architecture Model

- New Tools:



- New Languages and Frameworks:

UAF OMG UNIFIED ARCHITECTURE FRAMEWORK®

OMG SYSTEMS MODELING LANGUAGE SysML™

UML®

- The digital transformation of systems engineering depends upon the creation of well-crafted, consistent, and complete descriptive and executable system models.
- Skilled modelers are in short supply
- Growing new modelers requires coaching and guidance
- Many ways to build a bad model
- Models are huge and complex to review (manually) (10^5-10^6 elements)

A robust model development process is needed to build useful models quickly and well.

Research Questions for a System Architecture Model (SAM) Development Process

- The following research questions drove development:
  - What system architecture products should be created within the SAM?
  - How does one use SysML to create those systems architecture products?
  - What order should those systems architecture products be developed, and more importantly what are the dependencies across them?
  - How does one enforce consistency across architecture products?
  - How does one enforce consistency across the team to develop a coherent SAM?
  - How does one prevent floundering and increase the tempo of development within the SAM?
  - How does one structure the data such that the model is able to support analysis

Existing processes for constructing a SAM continue to focus on diagrams as opposed to data and do not meet our full set of needs

Philosophical Themes of Our Approach

Theme	Description
Data-Centricity	The process/approach values data and the relationships between data above diagrams, nomenclature, visualization etc. The purpose of building a descriptive model is to manage the complexity of data which describes the system in a readable, analysable, and sustainable way.
Defect Reduction	A central goal of this process, and DE/MBSE at large, is to prevent defects introduced early on in the systems engineering process from lingering and festering into serious issues requiring costly corrections later in the systems lifecycle.
Architectural Consistency	A major issue with Document Based Systems Engineering (DBSE) is the inability to keep data from one view of the architecture consistent with others. This approach leverages the use of an integrated SysML model to enforce consistency across various aspects of the architecture.
Architectural Separation of Concerns	The operational problem, engineering problem, and engineered solution are all aspects of the system architecture, however should be clearly distinguishable and curated with separately while maintaining realization based traceability.
Style Commonality	The process/approach encourages common style methods are used by all contributing developers, and leverages automated validation to facilitate scaling team size and reducing the learning curve during on boarding.
Modeling Efficiency	Quality and timeliness are often competing requirements in technical work. This process/approach increases the tempo of development and provides a clear path to completion through process definition, maturity alerts, and a modelled system example for comparison.

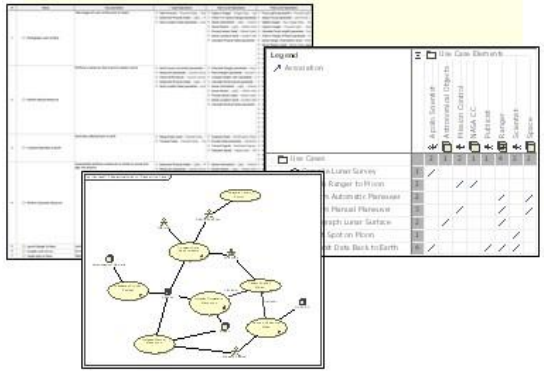
Development Process Assumptions

- The following assumptions served as a basis for development:
  - Top down architecture decomposition
  - The system boundary is definable
  - The process is limited to support for descriptive model development
  - Multiple engineers will develop the SAM simultaneously in parallel
  - The SAM is constructed iteratively and through refinement
  - The team has familiarity with SysML and their modeling tool (Cameo, Rhapsody, etc.)
  - The customer has provided some sort of top level requirements document and/or other source content which provide pedigree

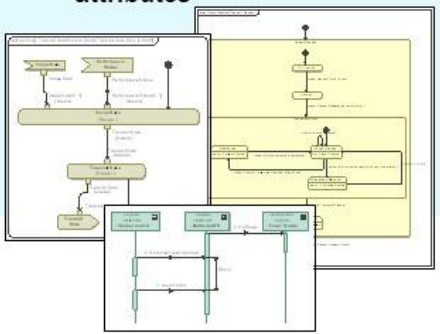
Tailoring to the process may be required if any of these assumptions do not hold for a given program

Architectural Separation of Concerns

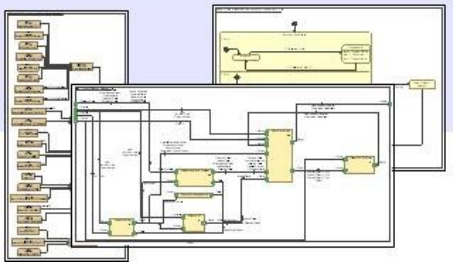
- **Behavioral Architecture**
  - Defines the **operational role** of the system
    - Use cases & use case diagrams
    - Associated actors
    - Top level activity diagrams integrated with the Logical Architecture



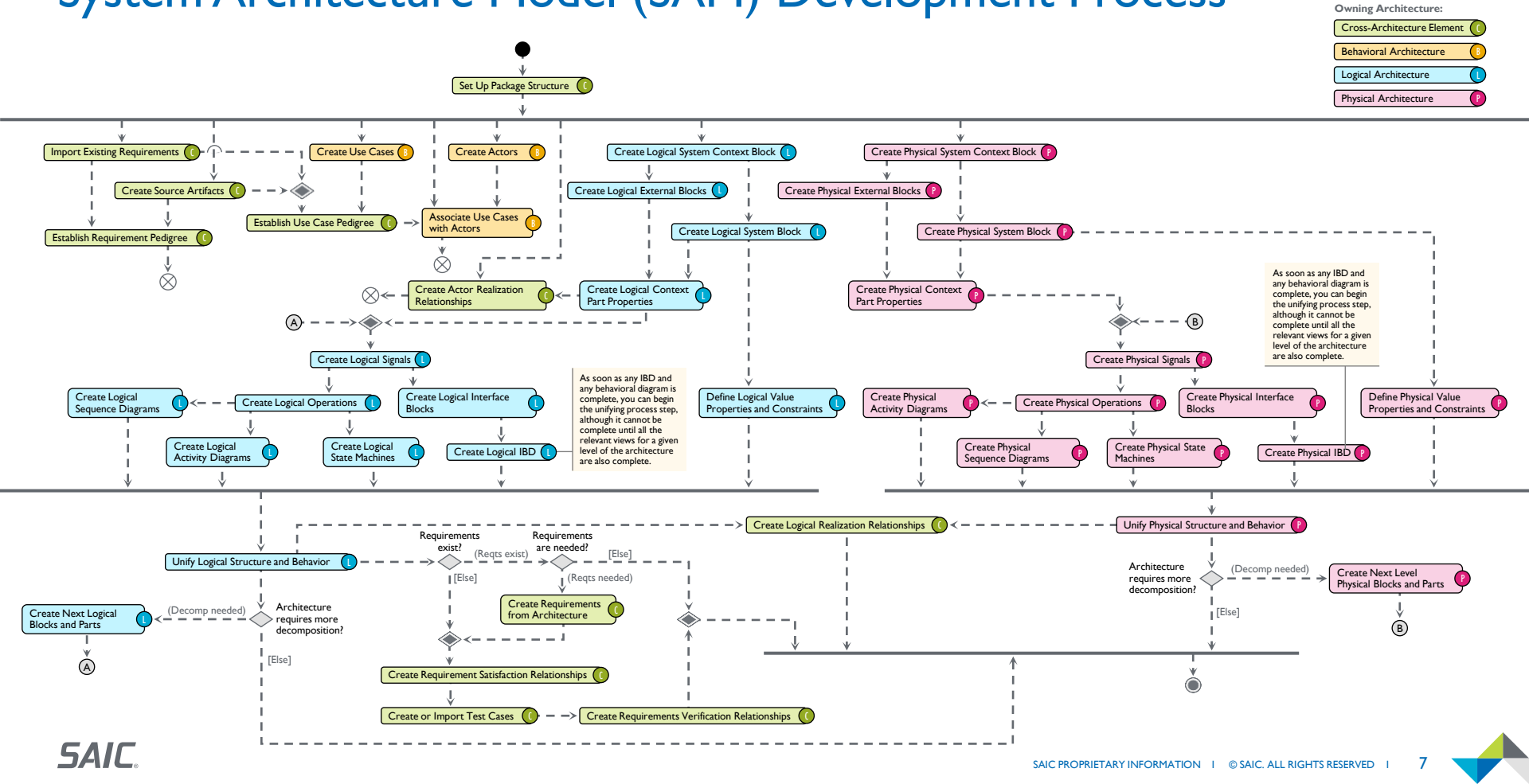
- **Logical Architecture**
  - Defines the **problem space**
    - Behavioral decomposition through activity, sequence, & state machine diagrams
    - Structural decomposition through block definition and internal block diagrams
    - Interface definition through signals
    - Value properties defining needed attributes



- **Physical Architecture**
  - Defines the **solution space**
    - Behavioral decomposition through activity, sequence, & state machine diagrams
    - Structural decomposition through block definition and internal block diagrams
    - Interface definition through signals
    - Value properties defining predicted/actual attributes
    - Realization of the Logical Architecture

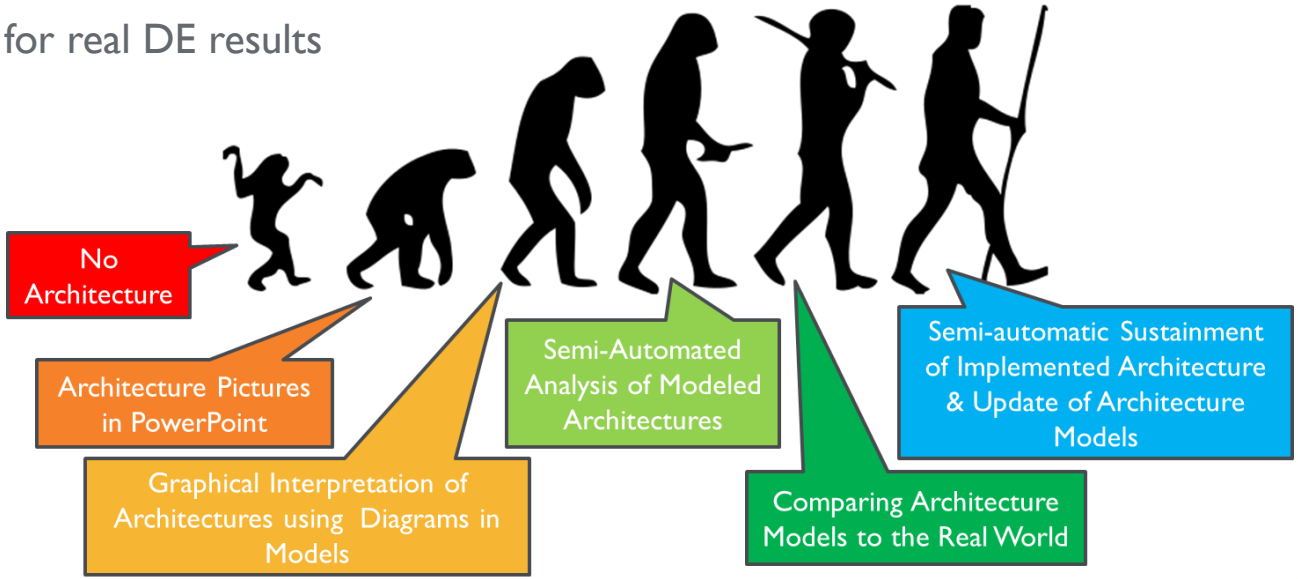


System Architecture Model (SAM) Development Process



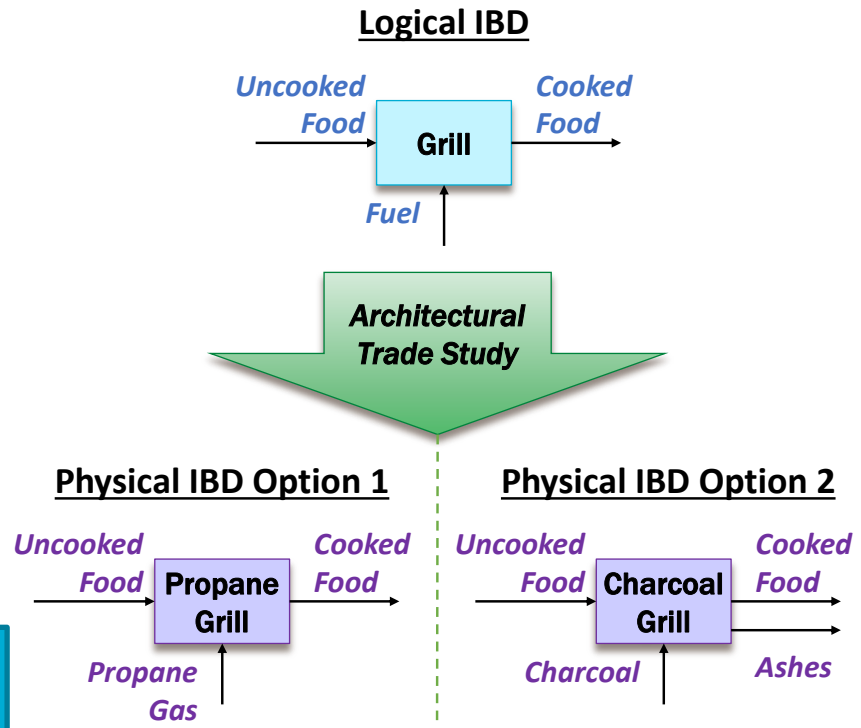
A Strictly Defined and Efficiently Enforced Model Style

- Enables automated or semi-automated **analysis** of the model data and the use of inference
- Enables the use of tool-based internal model **validation** capability
- Enables the automated analysis of the architecture for internal **consistency**: i.e. behavior vs structure, logical vs physical, nested architecture flows vs end-to-end flows
- Gets us in the green zone for real DE results



Key Characteristic: Dotted-Line Relationships Between Architectures

- Discriminates between need and solution supporting trades and decision making
- Enables federated approaches
  - I.e. customer defines the logical architecture and vendor provides the physical architecture
- Supports the use of COTS/GOTS/NDA solutions without corrupting the definition of need
- Enables parallel development of Logical and Physical Architectures for legacy systems



Provides a means to clearly and consistently model separate operational concerns, needs definition and solution definition

Unification Across Views and Internal Consistency

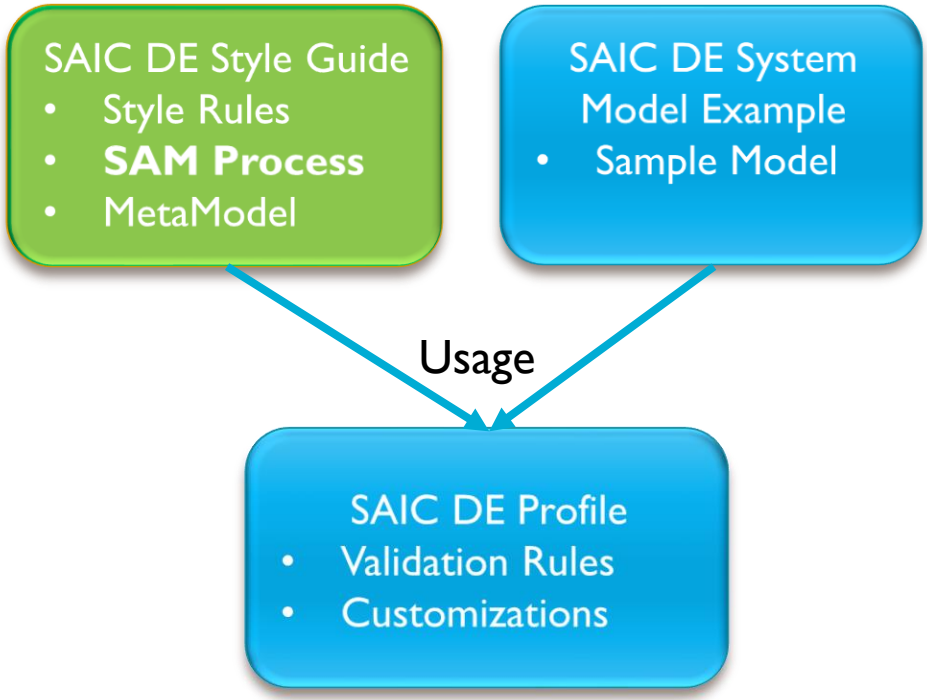
- Realization relationships between behavioral, logical, and physical architectures
- Mapping interfaces implied by behavioral views to interfaces defined in structural views
  - Object flows on Activity Diagrams
  - Messages on Sequence Diagrams
  - Signal Event transition triggers on State Diagrams
- Satisfying Requirements via the logical architecture
- Establishing Pedigree: formalizing sources of model data
- Leveraging extensive model syntax validation

The process focuses on coherently building the data to describe the system

Requirement Type	Architecture Element	Explanation
Functional	Operation	Operations are used within the SAM to represent functions, or what the system/component does.
Performance	Value Property	Value Properties define how well a system/component does something (a function). Value properties must also have "Value Types" and "Units" assigned accordingly. NOTE: Because performance requirements provide additional refinement of functional requirements, SysML "Refine" relationships between them are required.
Design Constraint	Value Property	Design constraints bound the architecture, sometimes in non-quantifiable terms. Value properties capture the system/component attributes which document these bounds.
Interface	Item Flow	The content of required interfaces are captured as SysML "Item Flows" within the structural portion of the architecture, which convey the signals defining more specifically what is passed over the interface.
	Proxy Port	For hardware oriented interface requirements (i.e. cabling), SysML "Proxy Ports," typed with "Interface Blocks", are used for satisfaction.

Availability and Tailoring

- SAIC Validation Tool: Non-proprietary, ITAR approved, and releasable from SAIC: https://www.saic.com/digital-engineering-validation-tool
- Traceability between the SAM process and the style rules allows users to identify which style rules are needed to support the portions of the process which are relevant to that program.
- Customizable: Import the rules selectively to create tailored, fit-for-purpose validations suites



All of the systems engineering industry benefits from quickly constructed high quality system models.

For More Information

MBSE Jobs: https://jobs.saic.com/pages/mbse

Digital Engineering: http://www.saic.com/digital-engineering

Contact Us: DigitalEngineering@saic.com