INTRODUCING

The Case for Integrated Model Centric Engineering

Model-Based Enterprise Summit 2019

Maged Elaasar, PhD (elaasar@jpl.nasa.gov)
IMCE Program Chief Architect
April 1-4, 2019

Jet Propulsion Laboratory
California Institute of Technology

Paper authors

Maged Elaasar
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA, USA
elaasar@jpl.nasa.gov

Nicolas Rouquette
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA, USA
Nicolas.F.Rouquette@jpl.nasa.gov

Steve Jenkins
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA, USA
J.S.Jenkins@jpl.nasa.gov

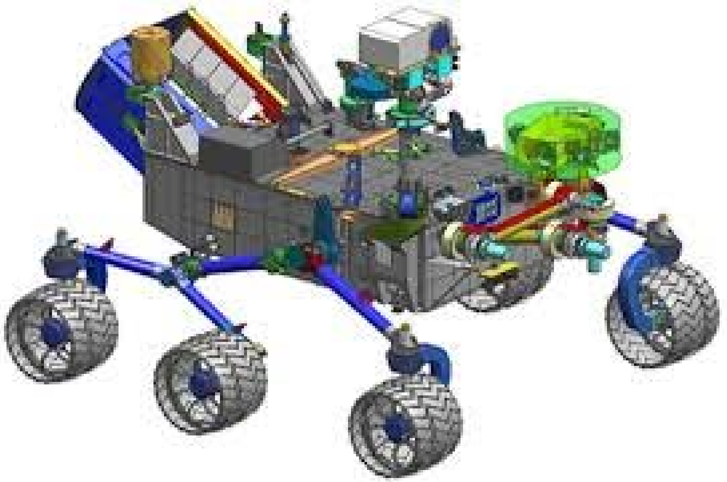
Sebastien Gerard
Laboratory of Research on Software
Intensive Technologies (LIST), CEA
Paris-Saclay, France
Sebastien.GERARD@cea.fr

Outline

1. Introduction: Challenges of MBSE
2. Characteristics of Modern MBSE Practice
3. Architectural Principles for IMCE
4. Impact of Adopting IMCE Practice
5. Related Works: INCOSE and OMG
6. Conclusions and Future Work

System model

- A system model is a simplified version that represents some details of interest about the system while suppressing others

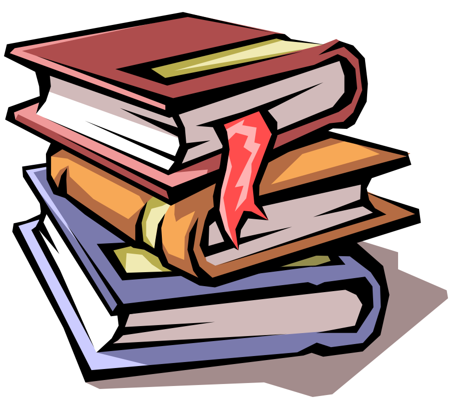


A model of the Mars rover

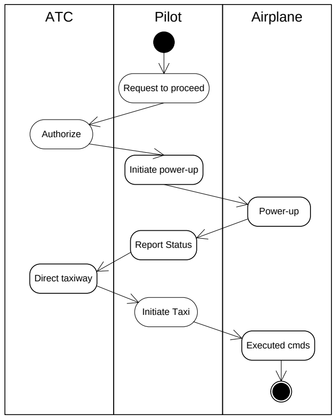
The real Mars rover

Model Based Systems Engineering

- Shared understanding of system requirements and design
- Assists in managing complex system development
- Improved design quality by reducing errors and ambiguity
- Supports early and on-going verification & validation to reduce risk
- Provides value through life cycle (e.g., training)
- Enhances knowledge capture



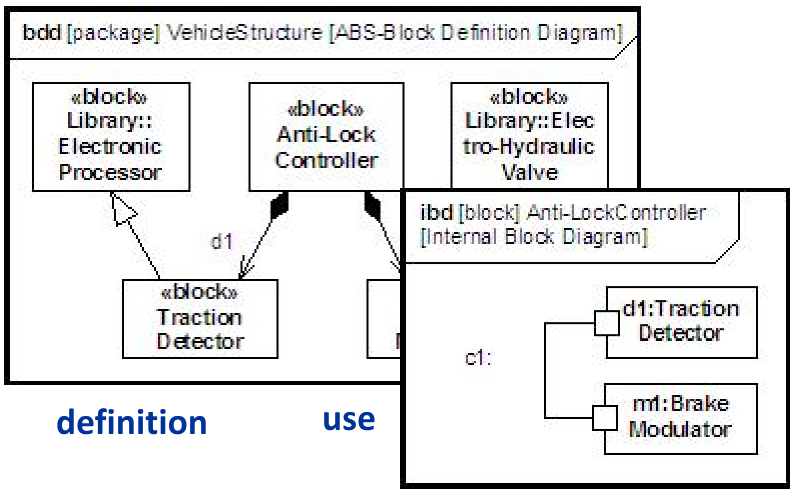
Document-based



Model-based

SysML: System Modeling Language

1. Structure



definition

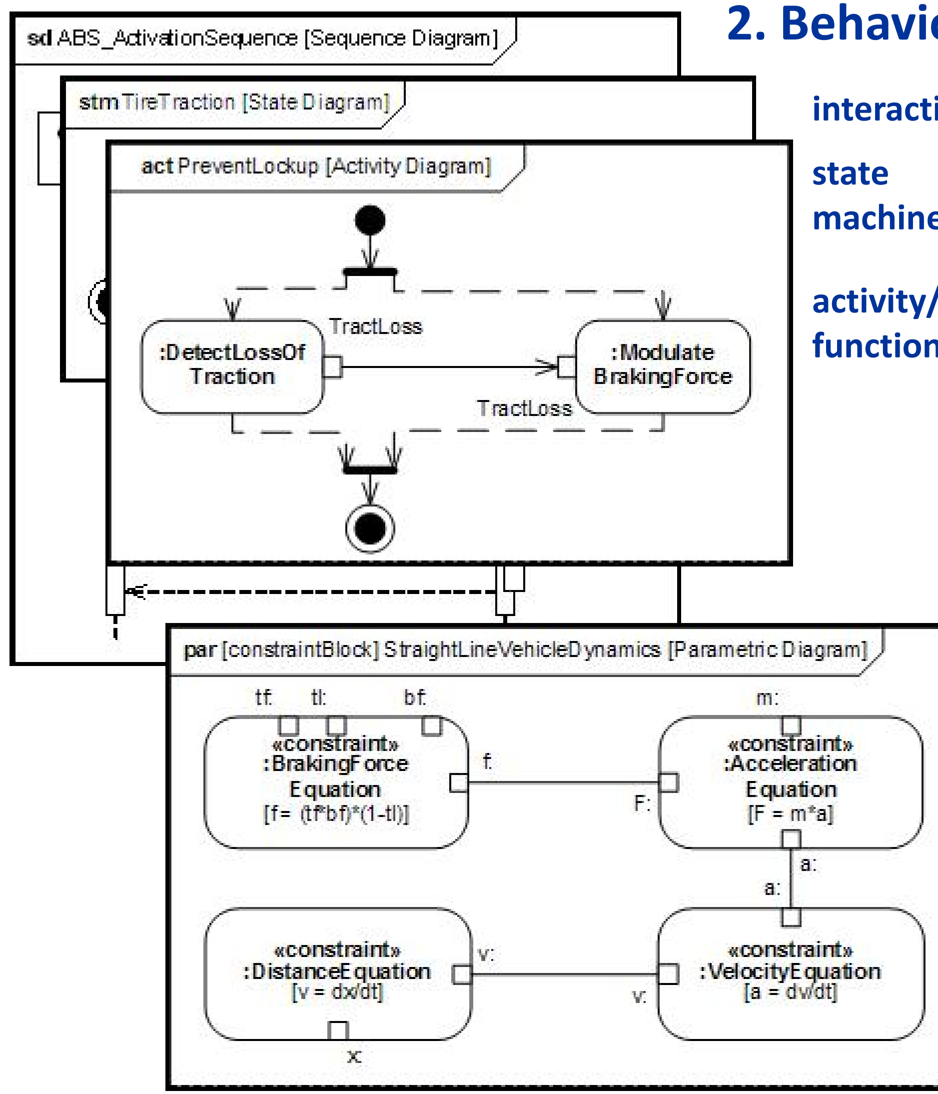
use

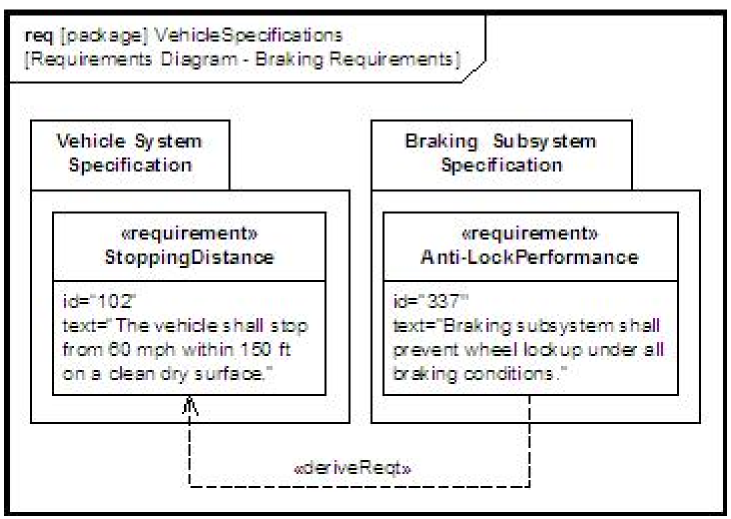
2. Behavior

interaction

state machine

activity/ function





3. Requirements

4. Parametrics

Introduction: Challenges of MBSE

1. No support for logical analysis
2. Methodologies not well supported
3. Poor support for model organization
4. Poor support for model integration
5. Poor support for model CM

Challenge 1: No support for logical analysis

- Until recently, SysML did not have any formal semantics
- Today some viewpoints have execution (operational) semantics
  - Foundational UML Subset (fUML)
  - Precise Semantics of UML State Machines (PSSM)
  - Precise Semantics of UML Composite Structures (PSCS)
- Still there is no support for logical analysis such as
  - Consistency (no contradiction)
  - Satisfiability (no incompatible constraints)

Challenge 2: Methodologies not well supported

- A methodology provides vocabulary, viewpoints, and guidance
- Example methodologies: OOSEM, State Analysis, Arcadia, CESAM
- Methodologies cannot to be effective unless they have:
  - An extensible modeling language to express vocabulary
    - SysML’s profile mechanism has limited extensibility, does not hide complexity
    - Using separate DSLs is possible but complicates integration
  - Custom tooling to support viewpoints and guidance
    - Some SysML tools allow custom viewpoints but still hard to do
    - Tool-supported guidance and automation rarely exist

Challenge 3: Poor support for model organization

- System model often needs be organized into a logical aggregate of model fragments assigned to different collaborating teams
- Model fragmentation often needs to be done around domain, discipline, and/or organization boundaries
- Existing MBSE tools provide poor support for model organization
  - SysML only allows fragmentation across containment relationships
  - Most DSLs do not allow one element to be described in multiple fragments

Challenge 4: Poor support for model integration

- Model fragments are often managed in different SE tools
- Fragment integration is usually poorly supported
  – Implicit correspondences need methodology-specific identify criteria
  – Explicit correspondences not possible due to differences in representations
  – Some standard efforts (ex: OSLC) to address this exist but are insufficient
  – Tool-specific solutions (ex: Syndia, ModelBus) exist but not methodology aware

Challenge 5: Poor support for model CM

- Most SE tools use their own proprietary CM system
  - Identifying a baseline across model fragments is hard
  - Managing cross-fragment linking is hard
- Challenges exist with using file-based CM systems
  - Lack of use of canonical (XMI) format for interchange (delta noise)
- Challenges exist with using database-base CM systems
  - Lack of a standard mapping to databases
  - Lack of content-based git-like commits

Characteristics of Modern MBSE Practice

- **Clarity**: the need for a precise language to describe system architecture and analysis in order to reach consensus among collaborating parties
- **Rigor**: adoption of mathematical principles in describing systems and performing analysis
- **Traceability**: preserve the association between an authority and its assertions, the rationale for a particular assertion, and links to other models that contributed to that assertion
- **Repeatability**: formalize the specification of analyses, including their dependencies, so that once an analysis has executed once, it can execute again with minimum overhead and cost
- **Durability**: information produced or needed by the SE analysis is stored in an accessible location permanently and become immutable
- **Efficiency**: employ automation through computation to augment, and in some cases, replace human processes

Integrated Model Centric Engineering

IMCE is a practice of MBSE that addresses the aforementioned challenges and strives to improve the desired characteristics by adopting a set of architectural principles:

1. Adopt Linguistic Rigor
2. Support Decisions with Analysis
3. Analyze at the Right Level of Abstraction
4. Define Patterns to Manage Complexity
5. Adopt CI/CD to Discover Issues Early
6. Use Content-Based CM
7. Use Deterministic Serialization of Model Content
8. Record Provenance of Model Content
9. Define and Verify Process Invariants
10. Define Artifact Organization Strategy
11. Account for Variations Explicitly

Principle 1: Adopt Linguistic Rigor

- Identify architecture stakeholders and their concerns
- Define architectural viewpoints that frame these concerns
- Define controlled vocabulary for each viewpoint that is
  - Sufficiently small to facilitate learning for proficient communication
  - Sufficiently expressive to convey concerns precisely

Principle 2: Support Decisions with Analysis

- Specify why a particular design is preferred (rationale)
- Specify how it addresses some concern analytically (explanation)
  - Should link to outcomes and values meaningful to stakeholders
  - should calculate the merit of the solution (how much conformance to concern)

Clarity	Rigor	Traceability	Repeatability	Durability	Efficiency

Principle 3: Analyze at the Right Level of Abstraction

- Employ decomposition to transform a problem in one domain into a set of related problems along discipline lines
- Analyze each problem separately in its own terms
- Make a principled connection between disciplines in the form of analysis

Principle 4: Define Patterns to Manage Complexity

- Complex systems are described by large numbers of simple facts
- These facts could be recurring coarse-grained patterns
- Describe these patterns separately then specialize them in the complex system description

Clarity	Rigor	Traceability	Repeatability	Durability	Efficiency

Principle 5: Adopt CI/CD to Discover Issues Early

- Use Continuous Integration (CI) and Continuous Delivery (CD) to
  - Regularly exercise analysis automation and provide early feedback on the analytical consequences of engineering decisions
  - Determine whether the predicted characteristics of the resulting design remain within acceptable margins for stakeholder concern criteria

Clarity	Rigor	Traceability	Repeatability	Durability	Efficiency

Principle 6: Use Content-Based CM

- Configuration manage all system model fragments and analysis results
- Use content-based CM (uses hashes of content as ids) in order to easily
  - Determine whether any two versions of content differ even in small details
  - Determine if a particular snapshot of content exists in the CM repository

Clarity	Rigor	Traceability	Repeatability	Durability	Efficiency

Principle 7: Use Deterministic Serialization of Model Content

- Use deterministic serialization to guarantee that a semantically-distinct model has one and only one syntactic serialization
  - Avoids creating unnecessary deltas in the CM system
  - Allows distinguishing content based on syntactic differences only

Clarity	Rigor	Traceability	Repeatability	Durability	Efficiency

Principle 8: Record Provenance of Model Content

- For each analysis, record the provenance of all inputs, outputs and computation logic used
- For chained analysis, recording provenance at every step of creating a chain of custody for all information back to its origin
- Combined with content-based CM, this capability guarantees semantic traceability for all changes made between versions of models in CM

Clarity	Rigor	Traceability	Repeatability	Durability	Efficiency

Principle 9: Define and Verify Analysis Invariants

- Specify unambiguous preconditions and postconditions for each analysis computation to ensure results are trustworthy
  - forces clear thought about the definitions of analysis steps
  - leads to deeper understanding of the engineering process
- Explicitly verify these conditions at each step of analysis, integrated into a CI/CD system, to ensure that violations of asserted conditions are detected reliably and as early as possible

Clarity	Rigor	Traceability	Repeatability	Durability	Efficiency

Principle 10: Define Artifact Organization Strategy

- A SE process is going to produce large numbers of artifacts to be stored and indexed for later access
- Devote thought to the organizational scheme for these artifacts, giving due consideration to affinities regarding (among others):
  - Concerns
  - Provenance
  - Access Control

Clarity	Rigor	Traceability	Repeatability	Durability	Efficiency

Principle 11: Account for Variations Explicitly

- Establish a baseline for the integrated system model, a distinguished branch in the CM system that is expected to change in an orderly fashion subject to a review process supported by analysis
- Maintain distinct variants in separate CM branches that can be merged with the baseline when desired
- Support performing a trade study, in which multiple possibly-incompatible design options are considered and evaluated according to some preference criteria, using variant branches

Clarity	Rigor	Traceability	Repeatability	Durability	Efficiency

Related Works

- **INCOSE: Integrated Data as Foundation for Systems Engineering**
  - Outlines the MBSE challenges
  - Discusses how they should be addressed from a data-centric perspective
  - Discuss how organizations should transition towards implementation
  - Defines a maturity scale for data-centric MBSE adoption

- **OMG: two relevant request for proposals**
  - *SysML v2*: enables more effective application of MBSE by improving precision, expressiveness, interoperability, consistency and integration of the language concepts relative to SysML v1
  - *APIs and services for SysML v2*: supports construction, query, viewpoint management, analysis, CM, and transformation of SysML v2 models

Conclusions and Future Works

- We identified key challenges of the MBSE practice today
- We highlighted desired characteristics of a modern MBSE practice
- We described the architectural principles of an IMCE practice that addresses the key challenges and improves the desired characteristics
- We are working on a reference architecture for a platform that supports an IMCE practice
- We are working on realizing the reference architecture as a platform called **OpenCAESAR** (https://opencaesar.github.io/) that is designed with state of the art technologies
- We are building a set of systems engineering applications on top of the platform to showcase its capabilities
- We are working closely with flight projects at JPL to transform their systems engineering practice into an IMCE one using the platform