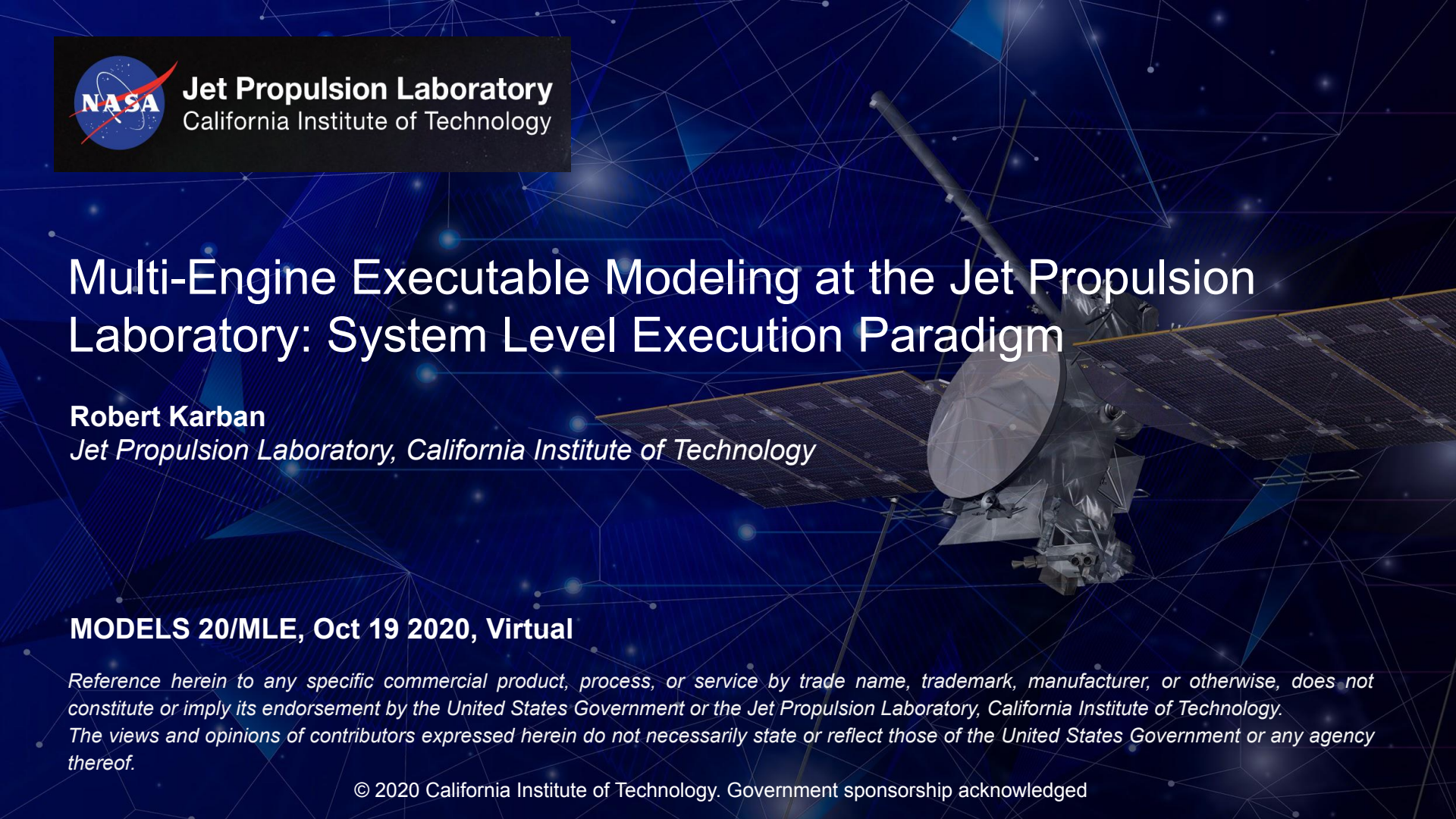


Jet Propulsion Laboratory
California Institute of Technology

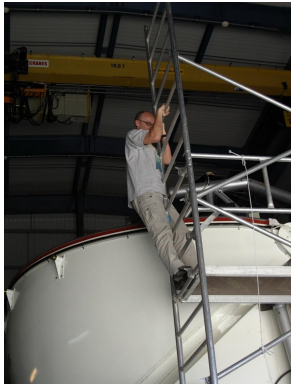
Multi-Engine Executable Modeling at the Jet Propulsion Laboratory: System Level Execution Paradigm

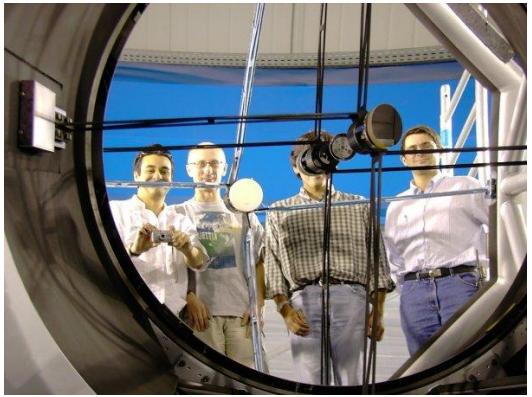
Robert Karban
Jet Propulsion Laboratory, California Institute of Technology

MODELS 20/MLE, Oct 19 2020, Virtual

Who is Robert?

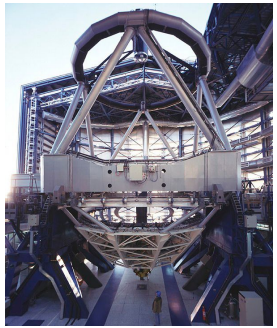
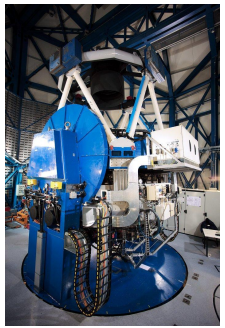
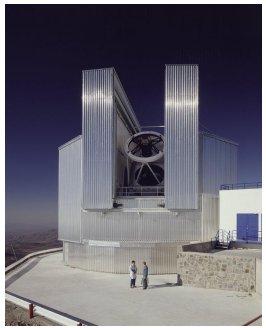
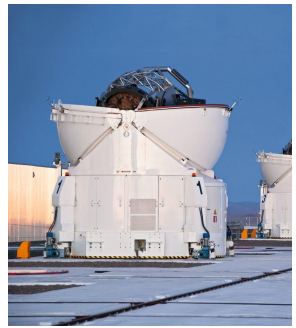
- CAE Systems and Software Environment Chief Engineer at JPL
- Member of INCOSE, SysML RTF, SysMLv2
- Formerly Control System/Software Engineer and Architect at:
  - European Southern Observatory (ESO) – Germany, Chile
  - CERN – Switzerland/France
  - Siemens Healthcare - Austria
- M.Sc. Computer Science (Austria)



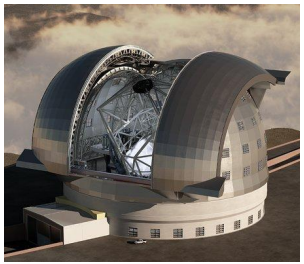
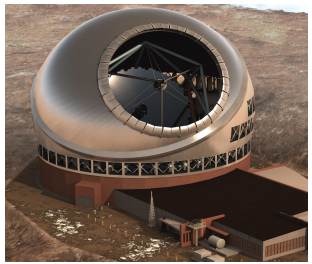



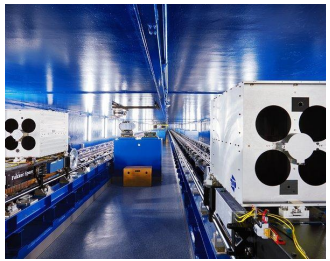

Telescopes and Instruments Robert worked on

- 4 x Very Large Telescope – 8m
- 4 x Auxiliary Telescope – 1.8m
- 1 x Very Large Interferometer – 130m - 200m
- 2 x Survey Telescope – 4m
- 55 x Radio Telescope – 12m - 15m
- 1 x 3.6m Telescope – 3.6m
- PRIMA and APE instruments
- 1 x NTT – 4m
- 1 x ELT – 39m
- 1 x TMT – 30m

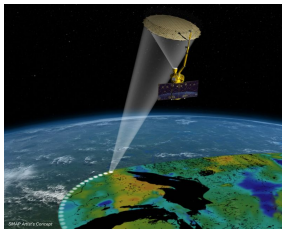


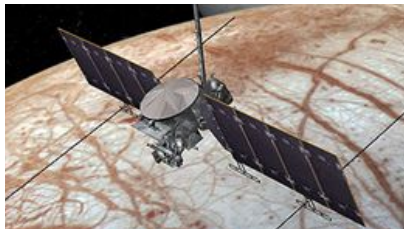




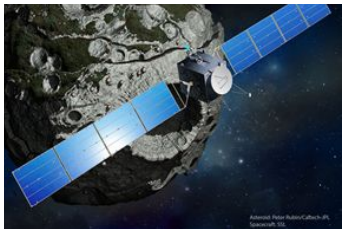


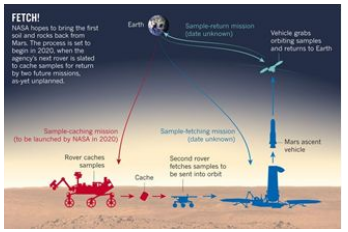
Space Missions Robert contributed to

- Europa Clipper
- Europa Lander Mission Concept
- Mars 2020
- Mars Sample Return
- Psyche
- SMAP



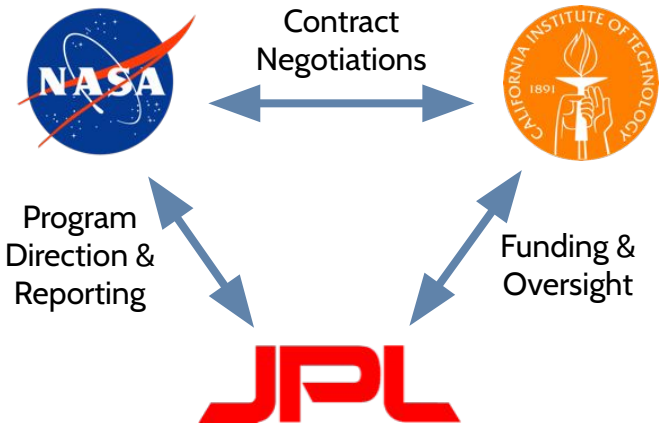


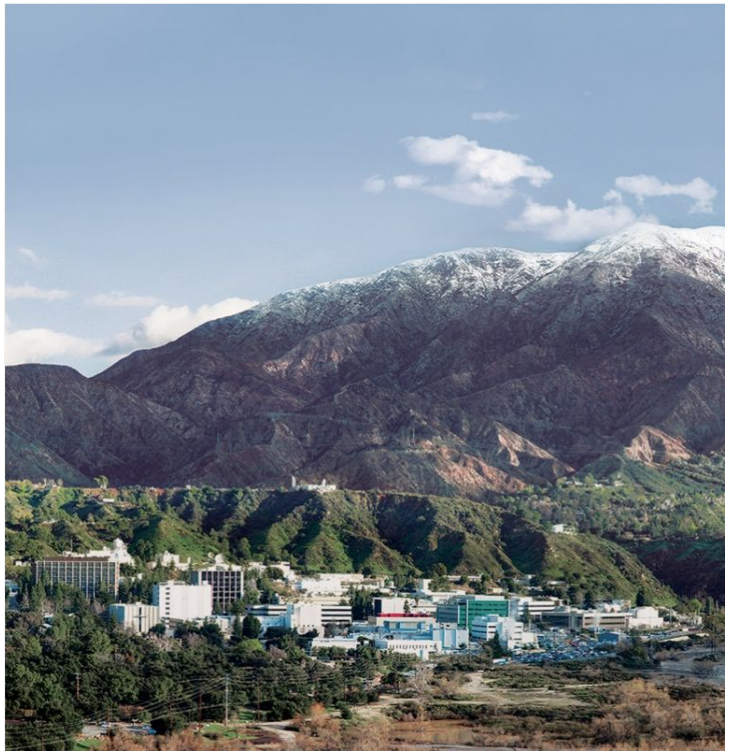




NASA Jet Propulsion Laboratory

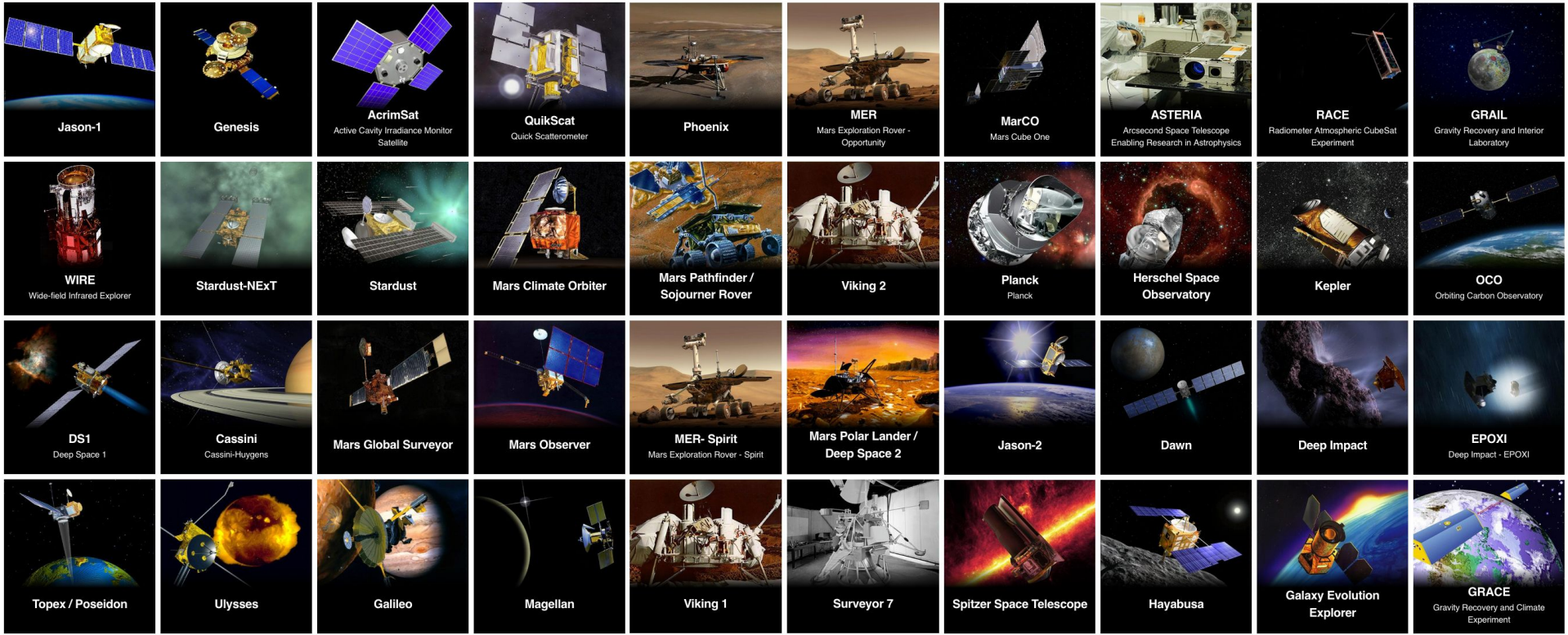
- Located in Pasadena, CA
- NASA-owned *"Federally-Funded Research and Development Center"*
- University-operated
- ~6,000 employees





JPL Campus in Pasadena, CA

Past Missions



Source: https://www.jpl.nasa.gov/missions/

Current Missions

Source: https://www.jpl.nasa.gov/missions/

Topic: The practitioner perspective

System and Software – the driver for executability

Why an executable Model (at System Level)?

D*escribe* a domain or system under study or to *specify* a (business, software and/or hardware) system to be built.

- Gain system **understanding** before or while building the system
- Specify parameters and behavior based on requirements and **validate** them in a System Model
- Express requirements more **precisely**
- **Analyze** how the as-designed system reacts to user interaction, predefined testing data and execution scenarios, timing and dynamic resource budgets

Why an executable Model (at System Level)?

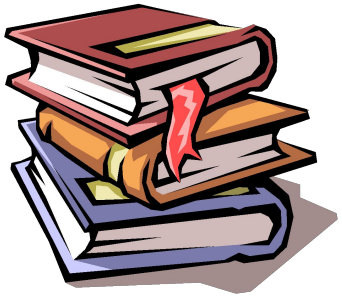
- **Explore** desirable and undesirable behaviors of a system
- **Check** consistency and logic of design
- **Understand** roles of the different components
- Enable (semantic) **consistency** across different **transformed** artifacts such as code, documentation, simulation, and analysis

Model Based Systems Engineering (MBSE)

- MBSE is the formalized application of modeling techniques to support system requirements, design, analysis, verification, validation and documentation activities
- MBSE expresses a system using a Systems Modeling Language (e.g. SysML, Modelica)
- MBSE is often applied with a method like Object Oriented System Engineering Method (OOSEM)

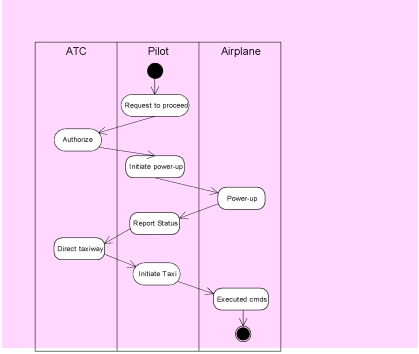
Vision from early 2000's

Present



- Specifications
- Interface requirements
- System design
- Analysis & Trade-off

Future



Moving from Document centric to Model centric

Topic: Archeology

Experiences with Executable Models at System Level

1990 Geode at Alcatel – Verilog France

- Based on **SDL'88** (SDL-92, SDL-2000, SDL-2010)
- Abstract grammar, concrete grammar and a full formal definition, uses ASN.1 data types
- Graphical editor, translator between **textual** representation (SDL-PR) to **graphical** representation (SDL-GR), semantic checker, simple **report** generators, some **code generation** (C).
- **System is specified as a set of interconnected abstract machines which were extensions of finite state machines** (FSM)
- Used for specifying **telecom systems**
- openGeode is still around

1994 Foresight at CERN - Nu Thena Systems

System design environment

- Capture **system requirements and designs** into **graphical executable** system models.
- Models could be **analyzed and simulated** to ensure requirements correctness
- **Discrete Event simulator**, modeling network communications and distributed computer systems
- Underlying textual action language (derived from Ada)
- Proprietary language features and provider of tools for the environment and language variant.
- Used for **simulations of particle physics experiments** with 10000s of channels

Not around anymore

1997

2007

SCXML
2015 (2010 draft)

fUML
2011

PSCS
2015

PSSM
2019

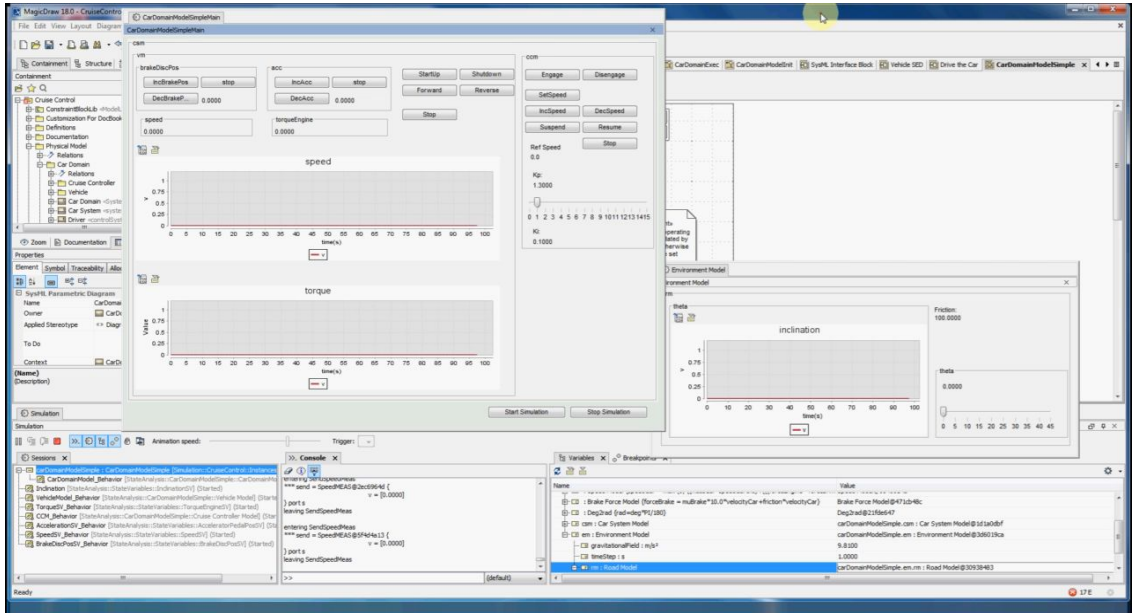
2011 (2008) - 3DS CAMEO SIMULATION TOOLKIT

Model execution framework and infrastructure (2020):

- Model **debugging** and animation
- **Multi-engine**, pluggable engines, languages and evaluators
- Engines are independent, **integration** is part of the toolkit
- User Interface **prototyping**
- Model driven **configurations** and test cases
- Open **Standards based**
  (W3C SCXML, OMG PSSM, OMG fUML, OMG SysML)

Cruise Control Model (2011)

- Modeling for control systems with collaborating state machines
- Demonstrate **integration of state machines, parametric, activity, interaction** models for a car and its environment



Topic: Current state of practice

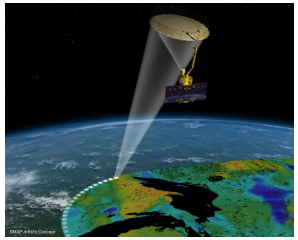
Today's solutions and issues

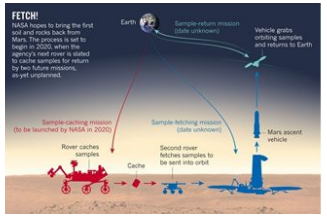
Selected JPL Applications

- Capture
  - Operational Scenarios
  - Fault Protection
  - Flight Software Behavior
- Model
  - Activities, State Machines, Parametrics, Sequence diagrams
- Produce
  - Interface Control Documents
  - Functional Design Documents
  - Analysis reports (e.g. Timelines)
  - Code

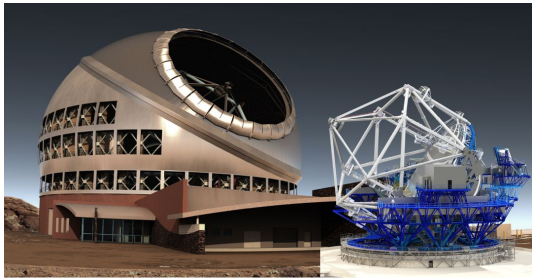
Thirty Meter Telescope, SMAP, Psyche,

Europa Lander Mission Concept, Mars Sample Return





TMT applies “Hybrid” Systems Engineering Approach



Traditional SE

- Clear, defined deliverables
- Easily accessible
- Shallow learning curve
- Simple traceability

MBSE

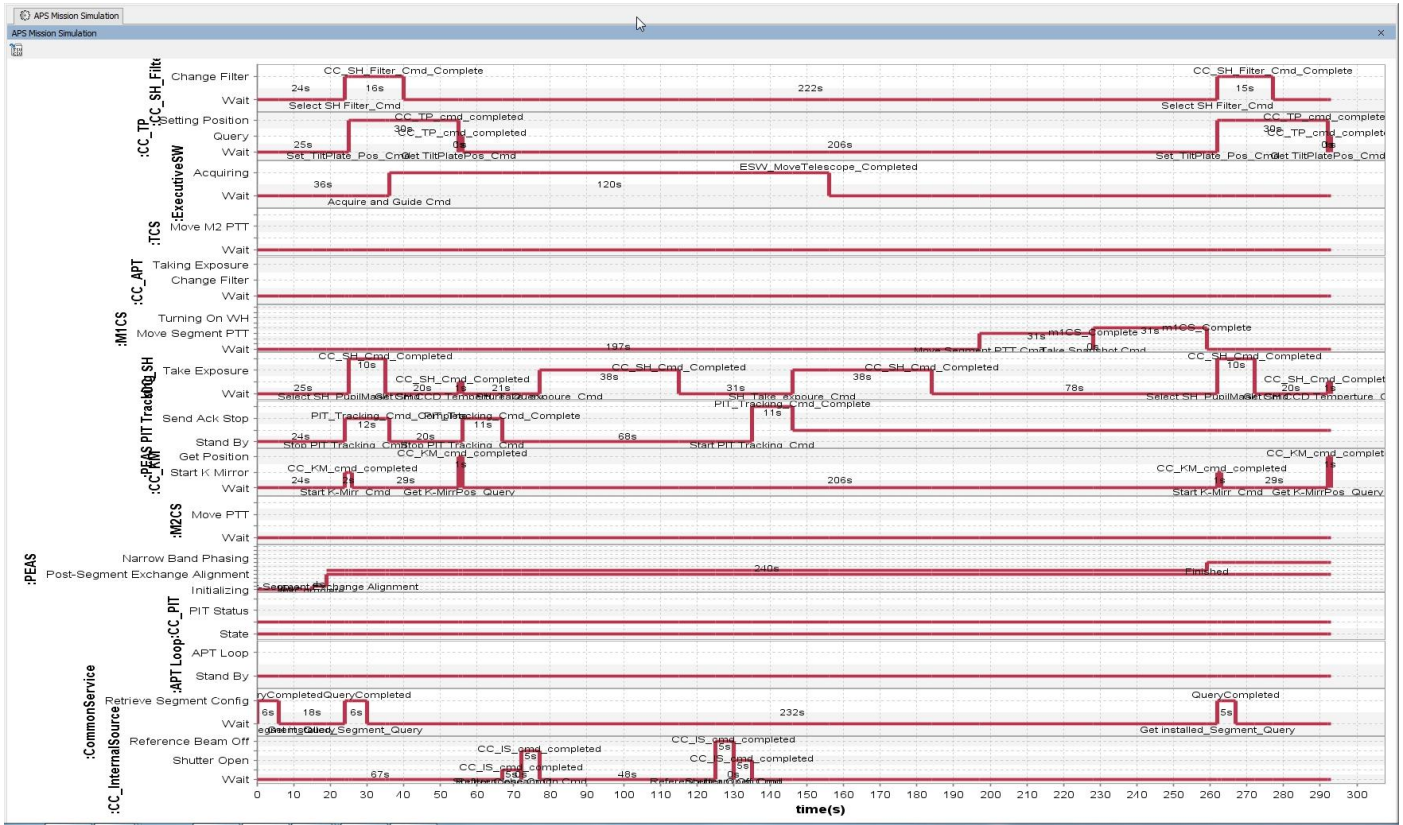
- Understanding behaviors of a system
- “Rich” capability to represent complex systems

Exploit the advantages of each approach

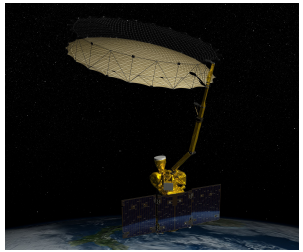
TMT MBSE Objectives

- Capture operational behavior, Identify involved subsystems, Identify interfaces and interactions among subsystems
- Develop/refine timing requirements for algorithms and external interface commands
- Create an **executable SysML model** with state machines and activities to specify behavior
- Use the model to **analyze the system design and verify requirements** on power consumption, mass, duration, pointing errors, ...
- Produce **engineering documents**
- Use **standard languages and techniques, and COTS tools where practical** to avoid custom software development

Sample TMT timeline



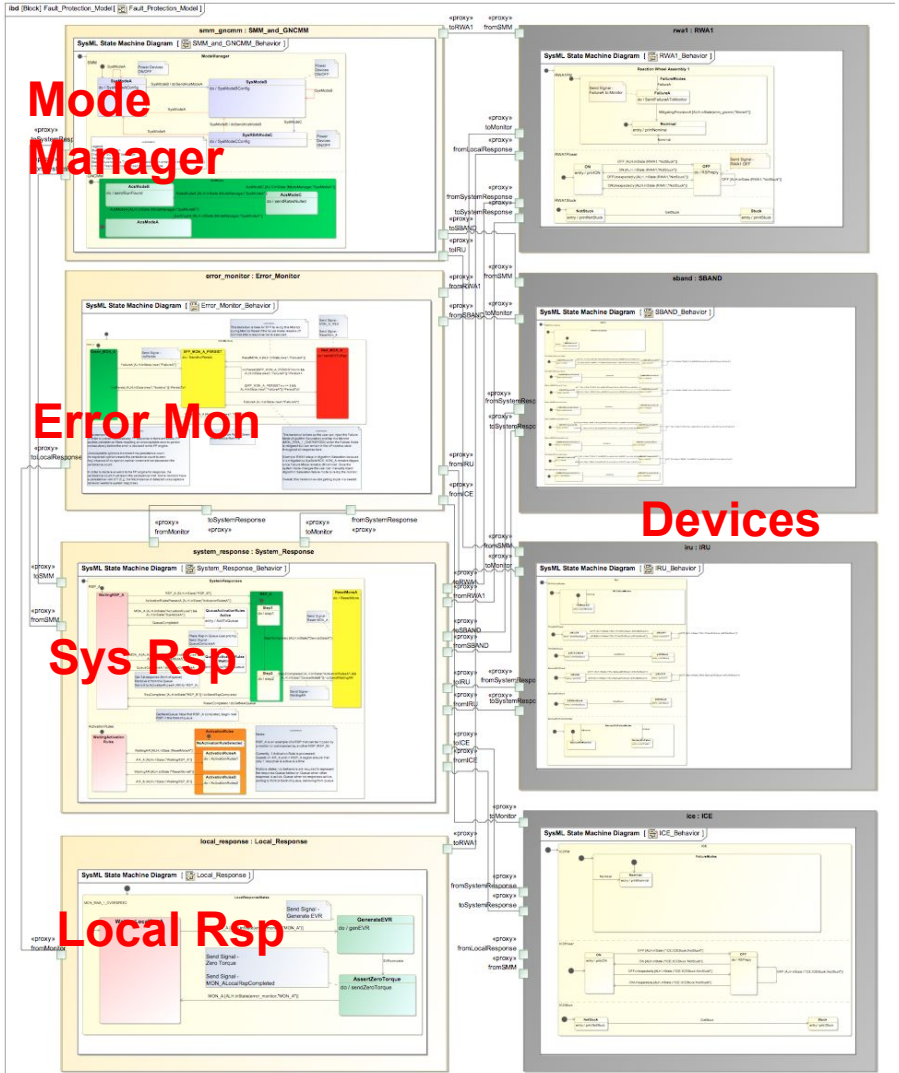
SMAP SysML Model



- Model of SMAP Fault Protection logical design in terms of structure and behavior.
- Behavior is modeled as network of collaborating Statecharts.
- Design Patterns for Fault Protections are identified.
- Model Checking

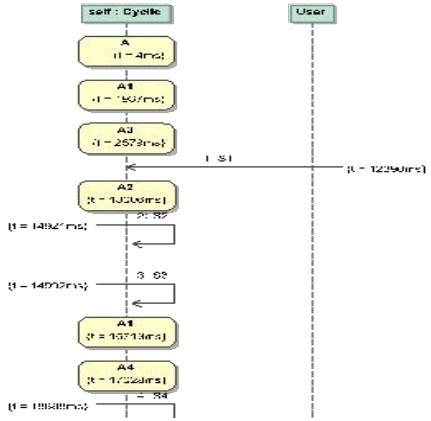
Size:
Error Monitors 6 (of 150)
Responses 8 (of 75)

of State Machines 8
# of States 184
# of Regions 68
# of Transitions 307
# of Guards 181
# of Signals 134
# of Connectors 21
# of Ports 42
# of Actions/Activities 173



Automatic Generation of Sequence Diagram

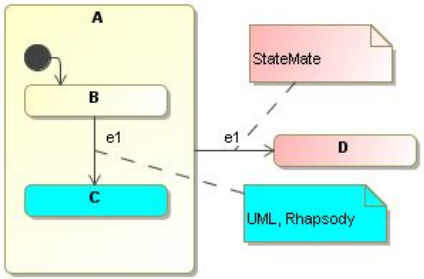
- Post-run behavior verification
- Goal is to run a handful of test cases for the different failure modes
- Verify that the correct responses that occur and that proper actions were taken to mitigate the fault



Practitioner challenges with Adoption

Semantic variations across standards and tools. For example:

- Regions, Run to completion, Null transitions, Time
- Ambiguous or incomplete language specification
- Tool problems - Standard Test Suite



- Bug or feature of specification or tool?
- User misunderstanding?
- Model wrong?
- Impacts model re-use

Select a (standard) execution semantic and stick to it in the whole tool-chain.

3-D Print - Model rendering on the target

- Execute model **directly** vs. generating code
- **Preserve** behavior and **render** it into the actual system
- Transformation of format
- Similarly to shaping a 3-D printed item you enable **modifying Behaviour@Runtime**
  W3C SCXML is a step in that direction
- Libraries and **execution engines** vs.
  arbitrary model, arbitrary transformation, arbitrary code

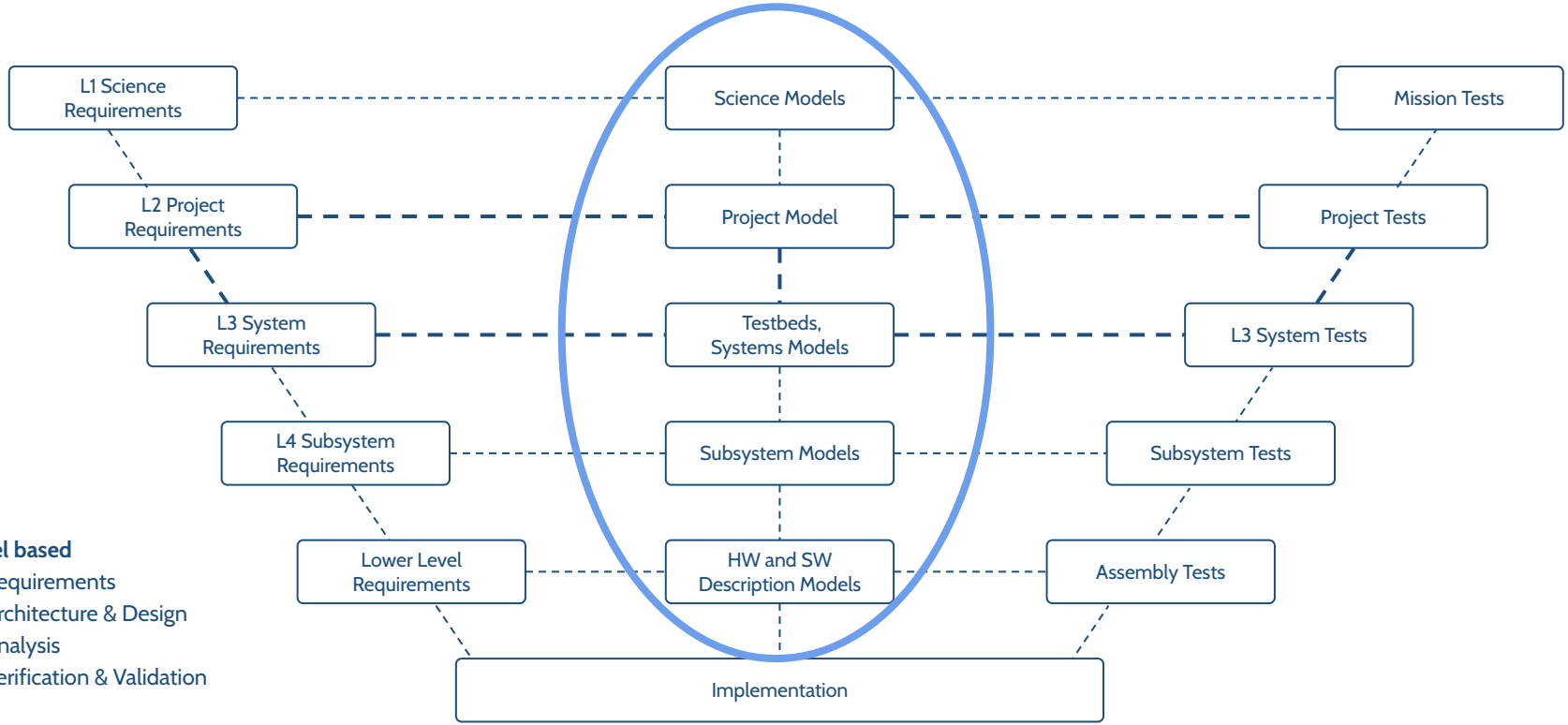
Topic: Enabling Technology

What is the future holding?

fmi SysML v2 jupyter CONDA

OpenMBEE

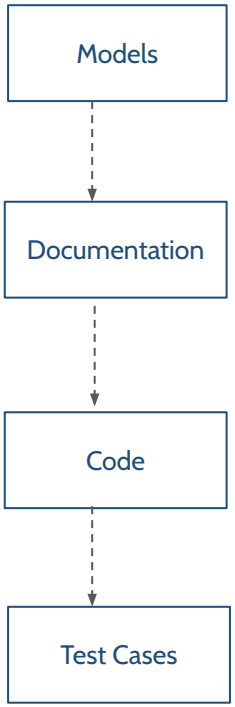
Model Driven V

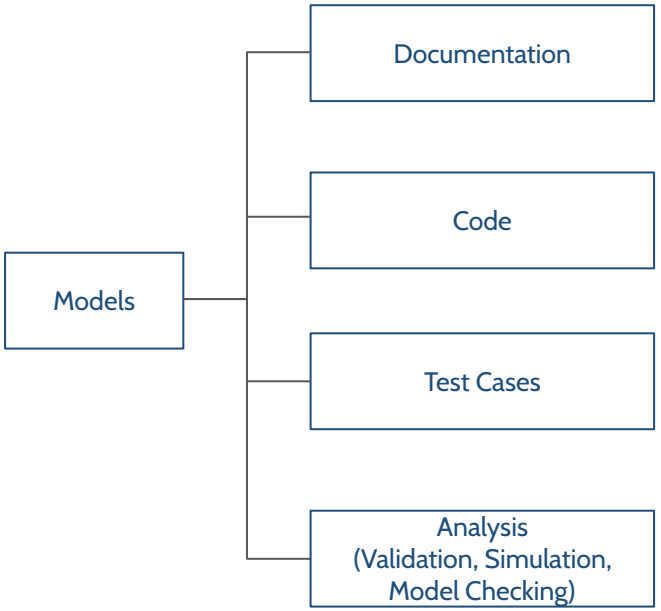


Model based

- Requirements
- Architecture & Design
- Analysis
- Verification & Validation

Model Driven instead of Document Driven – from System to Software

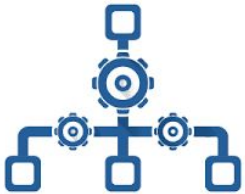




Semantic consistency across all transformed artifacts

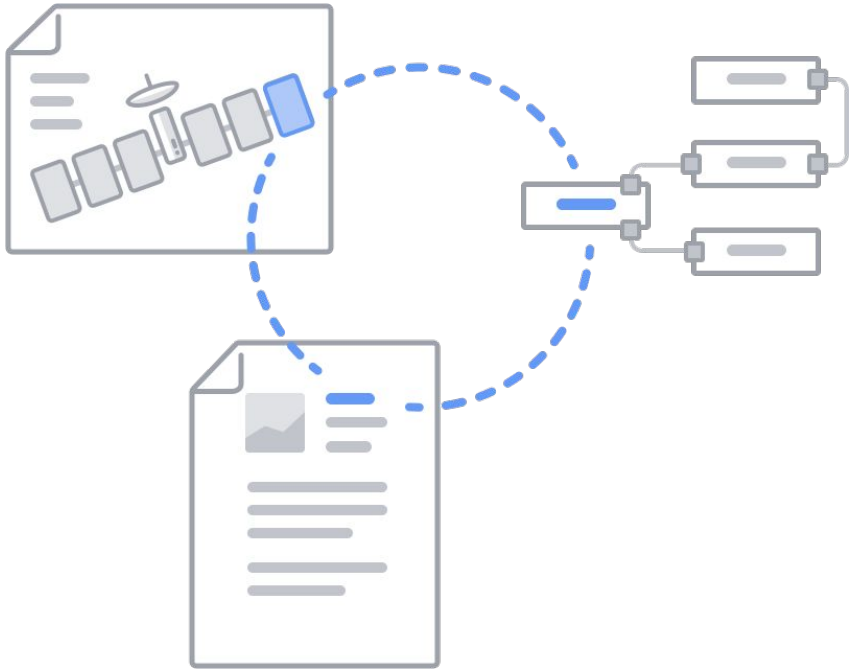
Objectives of an End-to-End Goal Based Executability Framework

- Produce a system **model** of the control architecture enabling **autonomy**
- Capture an **executable set of requirements**
- Clarify the understanding of **control architecture and operations**
- Perform **analysis** of operational scenarios and component behavior
- Enable **transformation**



Open Model-Based Engineering Environment

- OpenMBEE is a **community** for open source modeling software and models
  - Open source software activities
  - Open source models
  - Open source exchange of ideas
- Participants and adopters:
  JPL, Boeing, Lockheed Martin, OMG, NavAir, Ford, Stevens, Georgia Tech, ESO, ...
- > 400 members



Linked Data Documents with OpenMBEE

OpenMBEE Vision

- Augment Jupyter's **multi-language analysis** capabilities with modeling and connected engineering
- Enable novel data-driven analyses with advanced capabilities, as a **service**
- Unlock value through **commoditization**
- **Standards** powered engineering **platform** using SysML v2 + API & Services

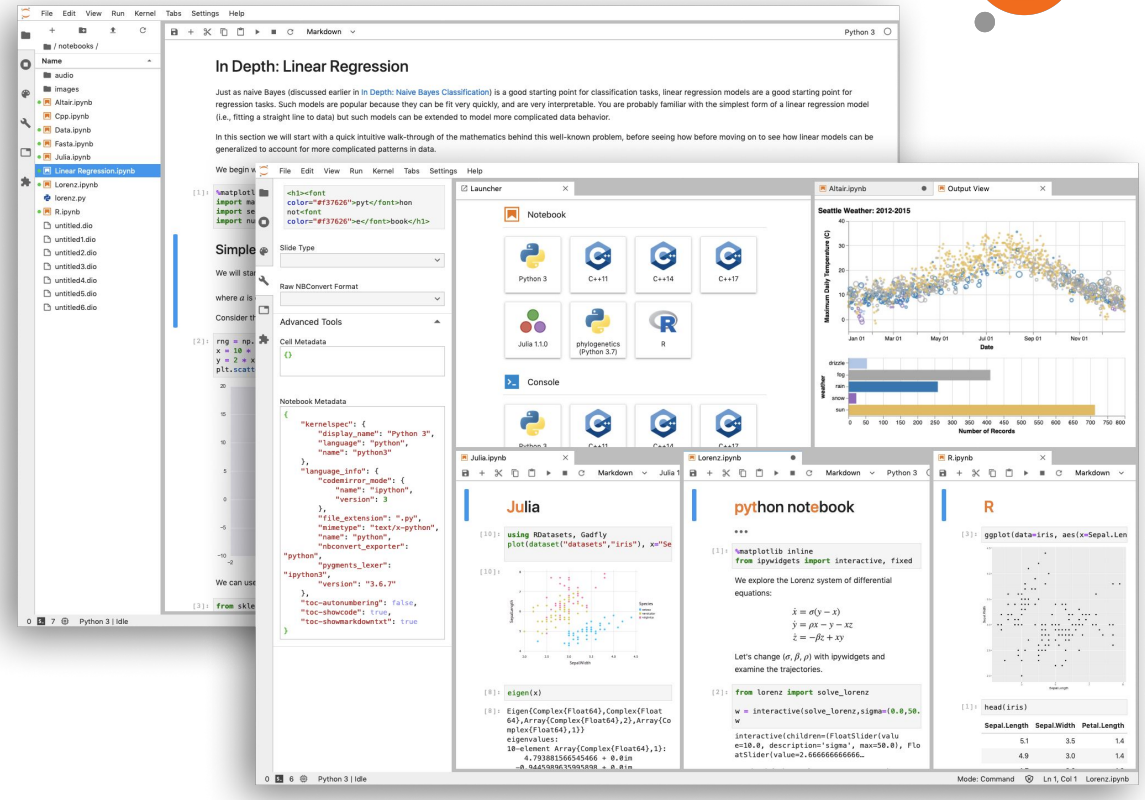
Join the community in making this vision into a reality @ openmbee.org

Global Engineering Ecosystem

Jupyter as Analysis and Visualization hub for Models

Interactive, exploratory, browser-based computing environment for:

- engineering
- data science
- scientific computing
- ML/AI
- and so much more...



SysML v2 Key Elements

- New metamodel that is not constrained by UML
  - Grounded in formal semantics
- Robust visualizations based on flexible view & viewpoint specification and execution
  - graphical, tabular, textual
- Standardized API to access the model

In [1]:

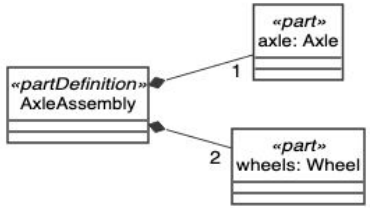
```
1  import ISQ::TorqueValue;
2
3  part def Axle;
4
5  part def Wheel {
6      part lugbolt {
7          attribute torque : TorqueValue;
8      }
9  }
10
11 part def AxleAssembly {
12     part axle : Axle[1];
13     part wheels : Wheel[2];
14
15     connection : Mounting connect axle to wheels;
16 }
17
18 connection def Mounting {
19     end axle: Axle[1];
20     end wheel: Wheel[*];
21 }
22
23 attribute def FuelCmd;
24
25 action def providePower (
26     in fuelCmd : FuelCmd,
27     out wheelTorque : TorqueValue[2]
28 );
29
30 part axle : AxleAssembly;
```

...

In [2]:

```
1  %viz AxleAssembly --style lr
```

Out[2]:



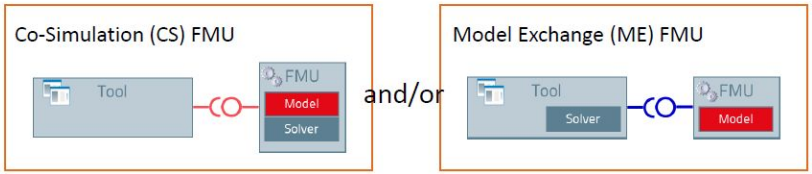
Functional Mock Up Interface

- Supported by more than 100+ tools (https://fmi-standard.org/)
- Custom IP protection
- Cost-effective deployment
- Compiled models
- Parameters can be changed
- Structure cannot be changed

fmi FUNCTIONAL MOCK-UP INTERFACE

Tool agnostic model encapsulation

The **FMI (Functional Mock-up Interface)** standard allows for the creation of tool agnostic models, **FMU's (Functional Mock-up Units).**



Conda

- Package and **dependency management** for models
- Enable sharing **distribution** of models as self-contained packages
- Model **re-use**

The Next Generation Systems Engineer's Dream Car

OpenMBEE

JSysMLv2yter

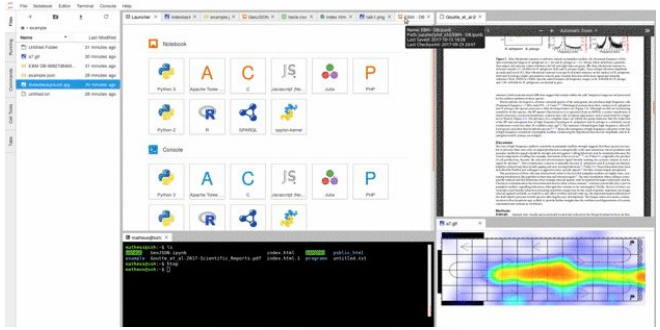
fmi

CONDA

Questions?

Backup

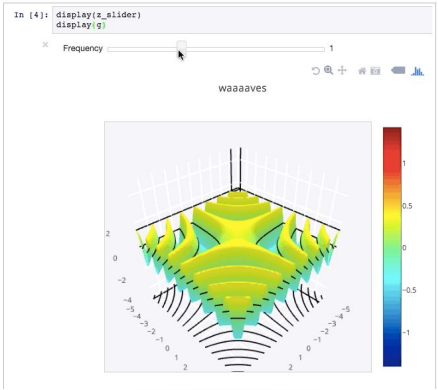
JupyterLab Features



Rich engineering narrative

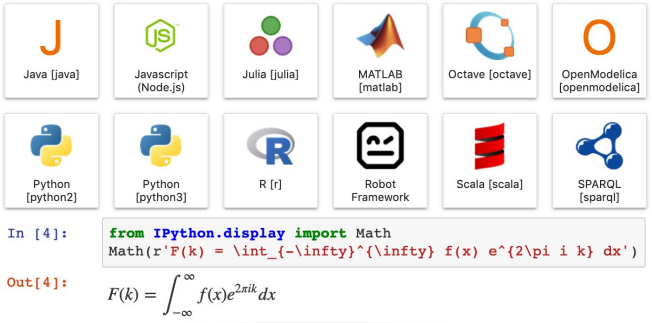
Feature rich and integrated computational environment

Notebooks as reproducible and executable reports



Interactive & visual

Live code, narrative text, equations (LaTeX), images, visualizations, audio, video, widgets



Widely popular & supported

~100 programming languages supported

Millions of public notebooks on GitHub to find inspiration