



**Jet Propulsion Laboratory**  
California Institute of Technology

# Taming Monsters with Dragons

Towards a Model-Based Product Development Process from Early Concepts to Engineering Implementation

**Robert Karban and Myra Lattimore**

*Jet Propulsion Laboratory, California Institute of Technology*

---

*Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.*

*Any permissions have been obtained and that proper credit of third party material has been cited.*

*The views and opinions of contributors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.*

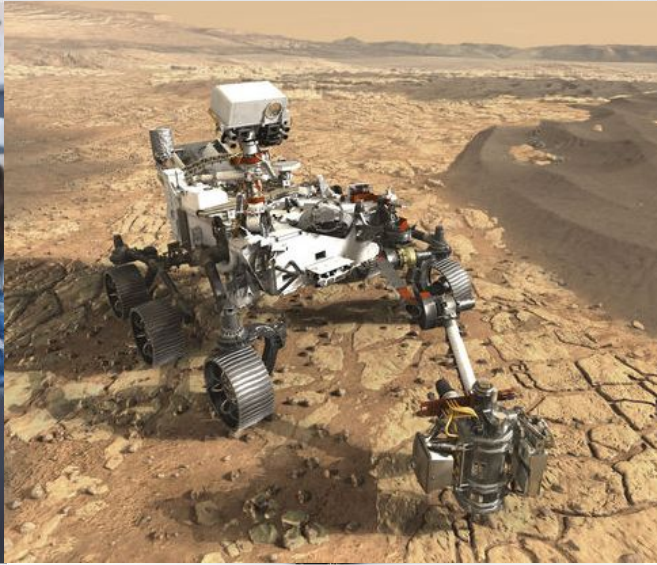
© 2023 California Institute of Technology



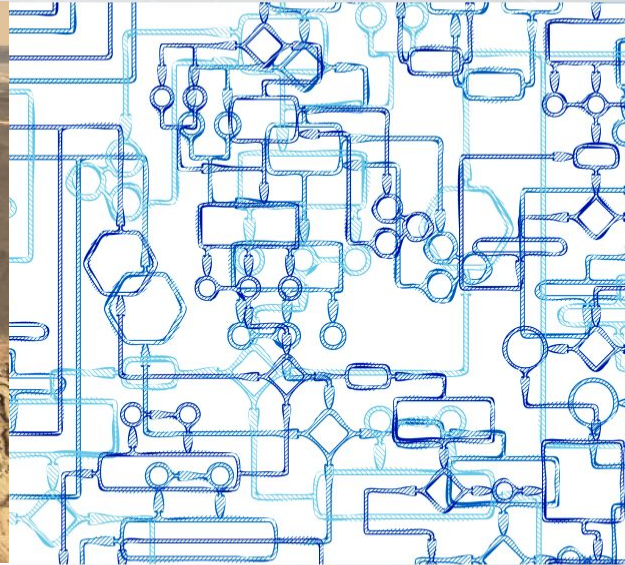
**Systems Engineers guide the  
concurrent collaborative design  
of complex technical systems**



**Leadership**



**Architect and Design  
Cyber-Physical Systems**

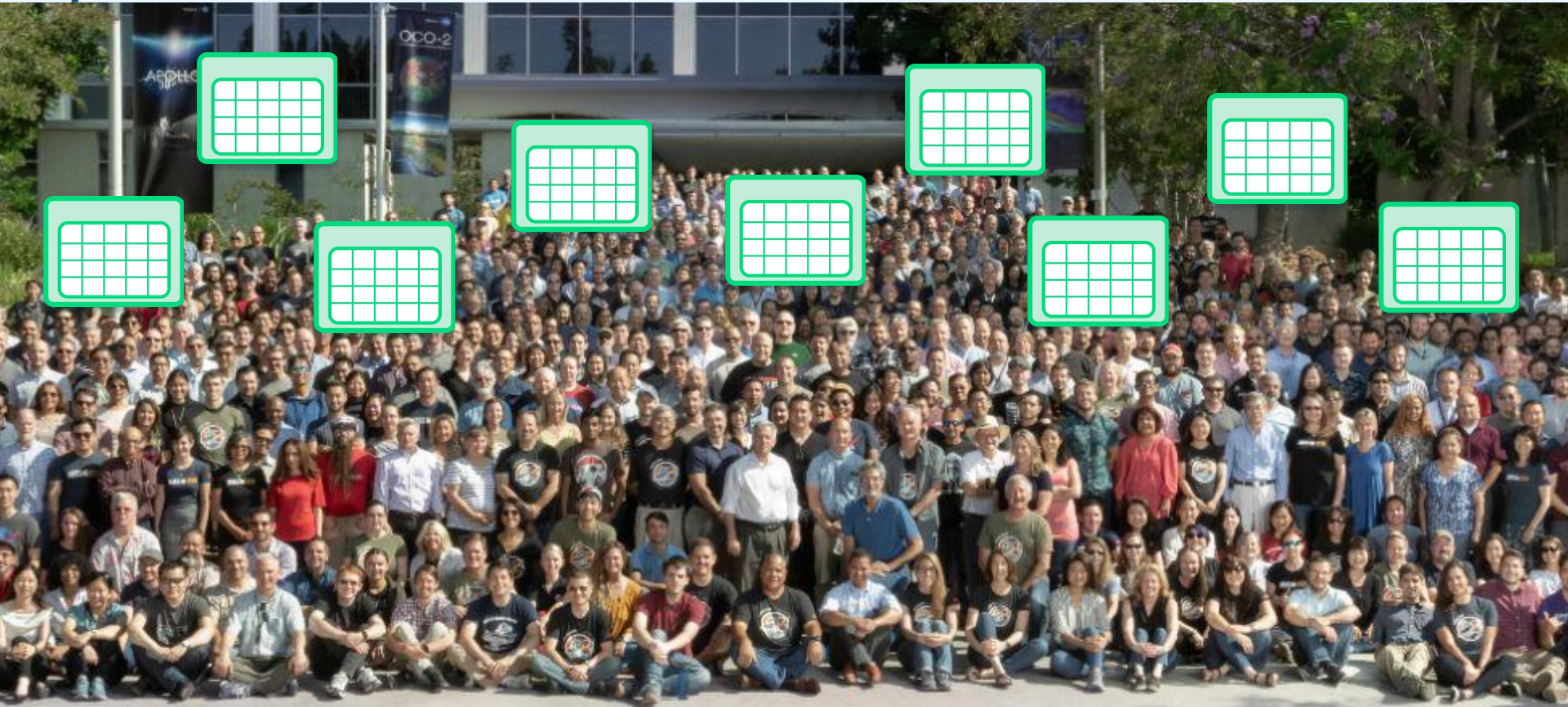


**Manage Complexity**

**Project teams are large. Lots of people work on one project...**



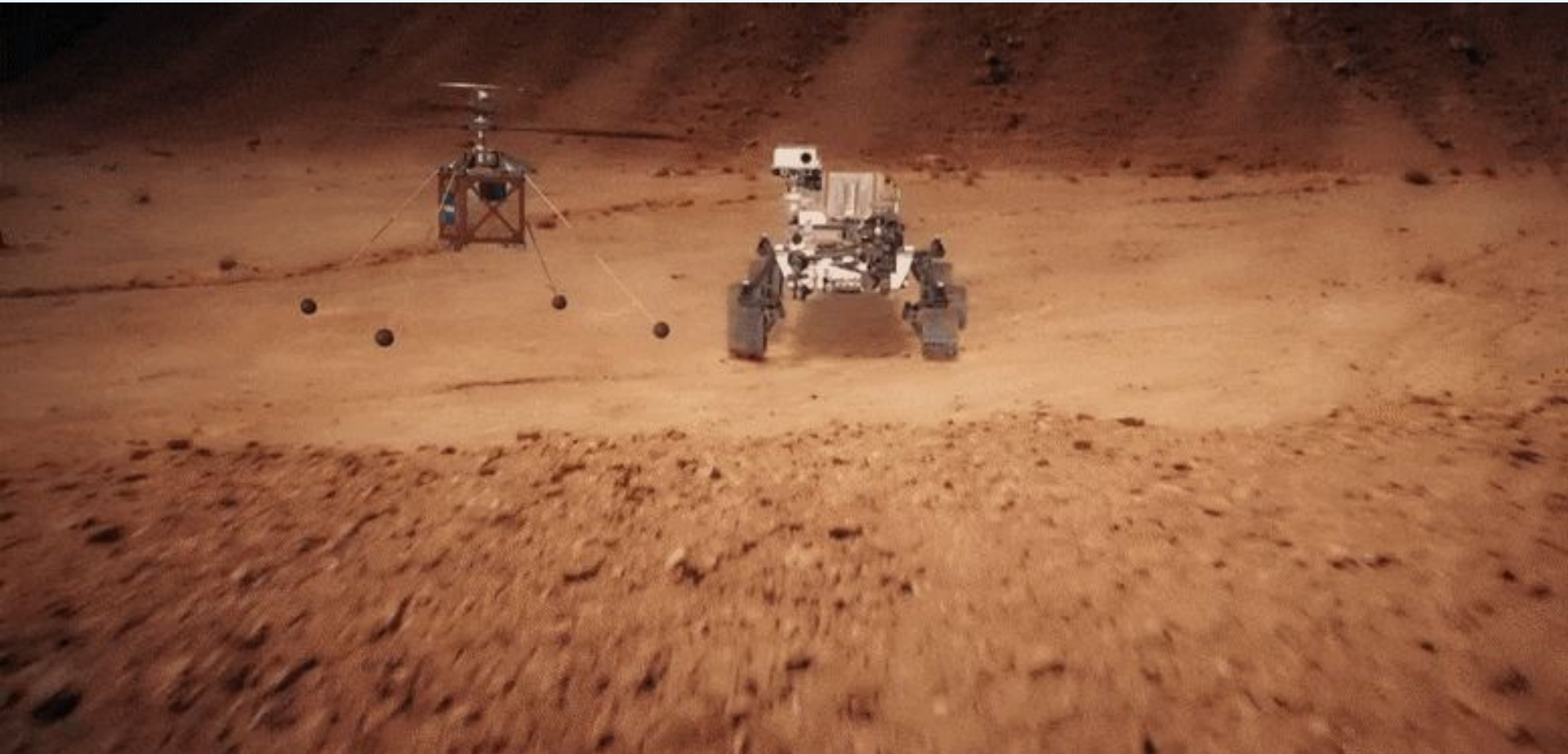
**...and they communicate their project needs often through spreadsheets.**



**To eventually do this...**

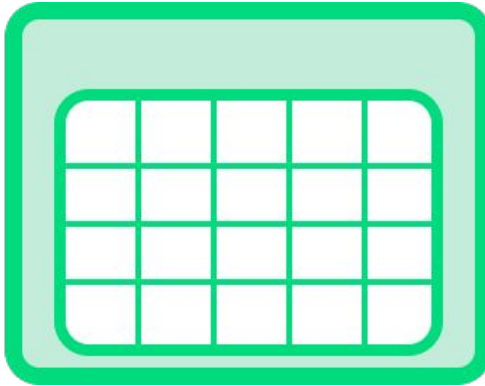


**And this. But it takes a lot to get here.**



# How can the systems environment help to bridge this gap?

So how do we get from this...

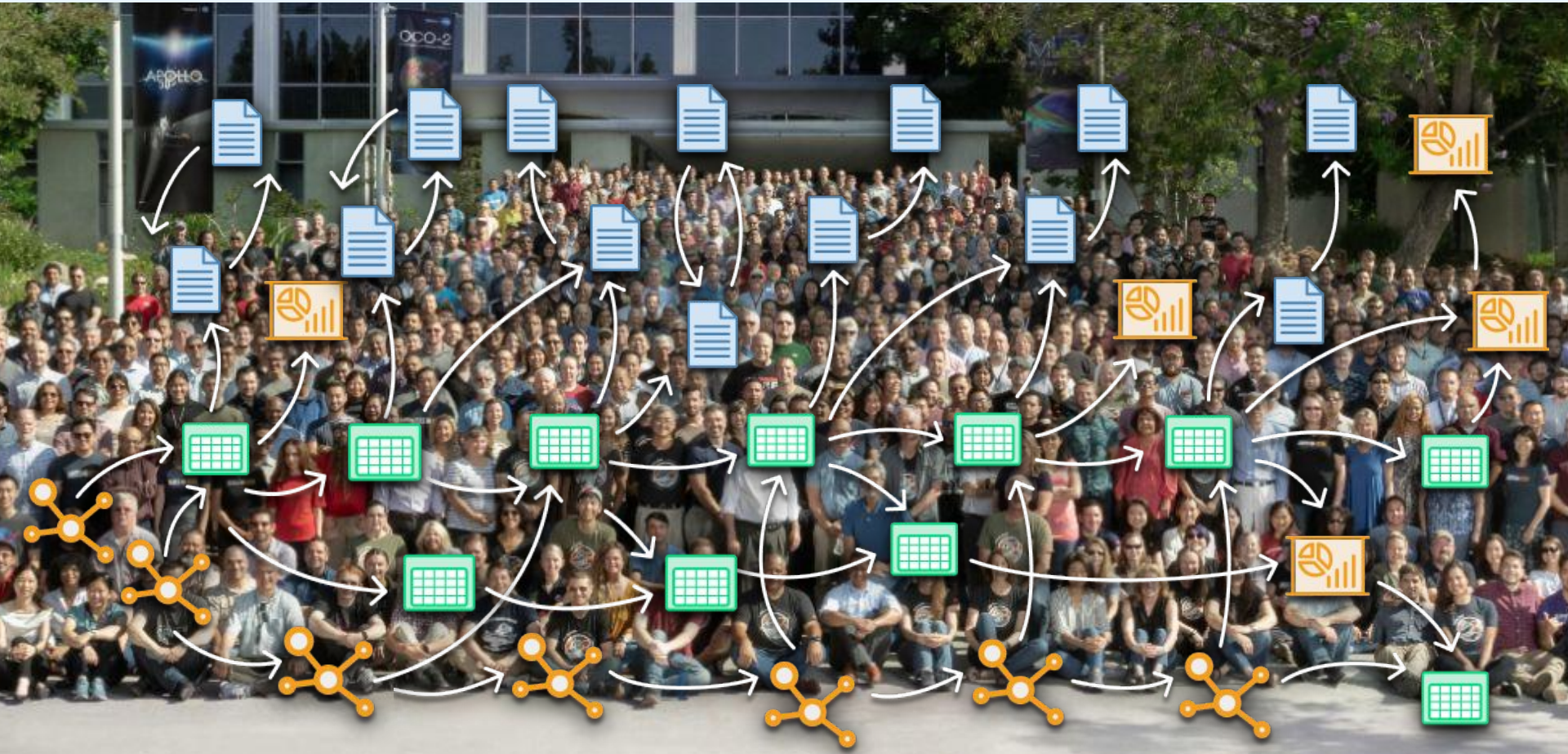


to this?

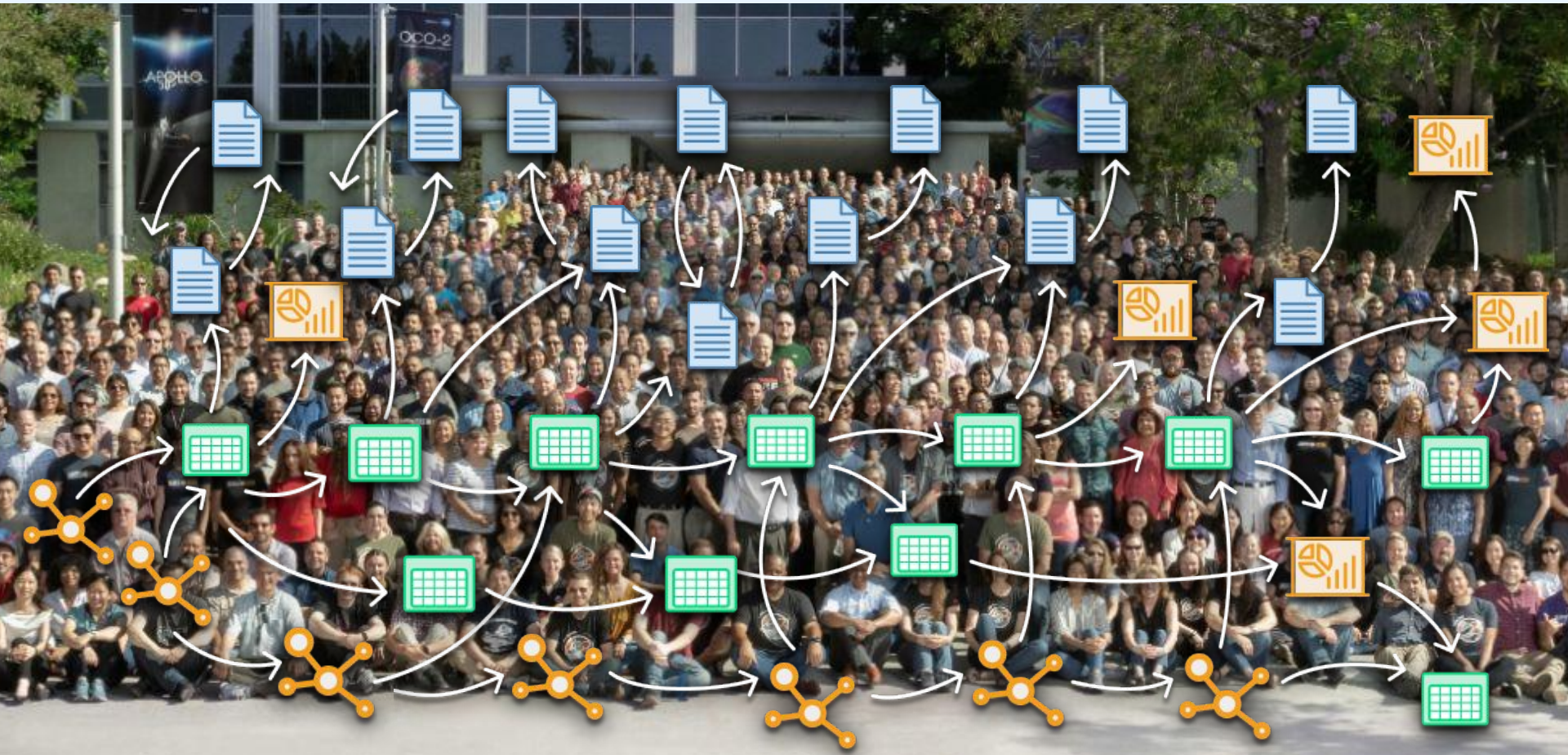




**It takes people, and requirements, and spreadsheets, and documents, and processes, and workflows, and tests...**



**Unfortunately, all of these things create silos of information that lead to miscommunication and duplicate work.**



**A project starts simple.**



**Engineers iterate on their models.**



**They add it to a spreadsheet to track it over time.**



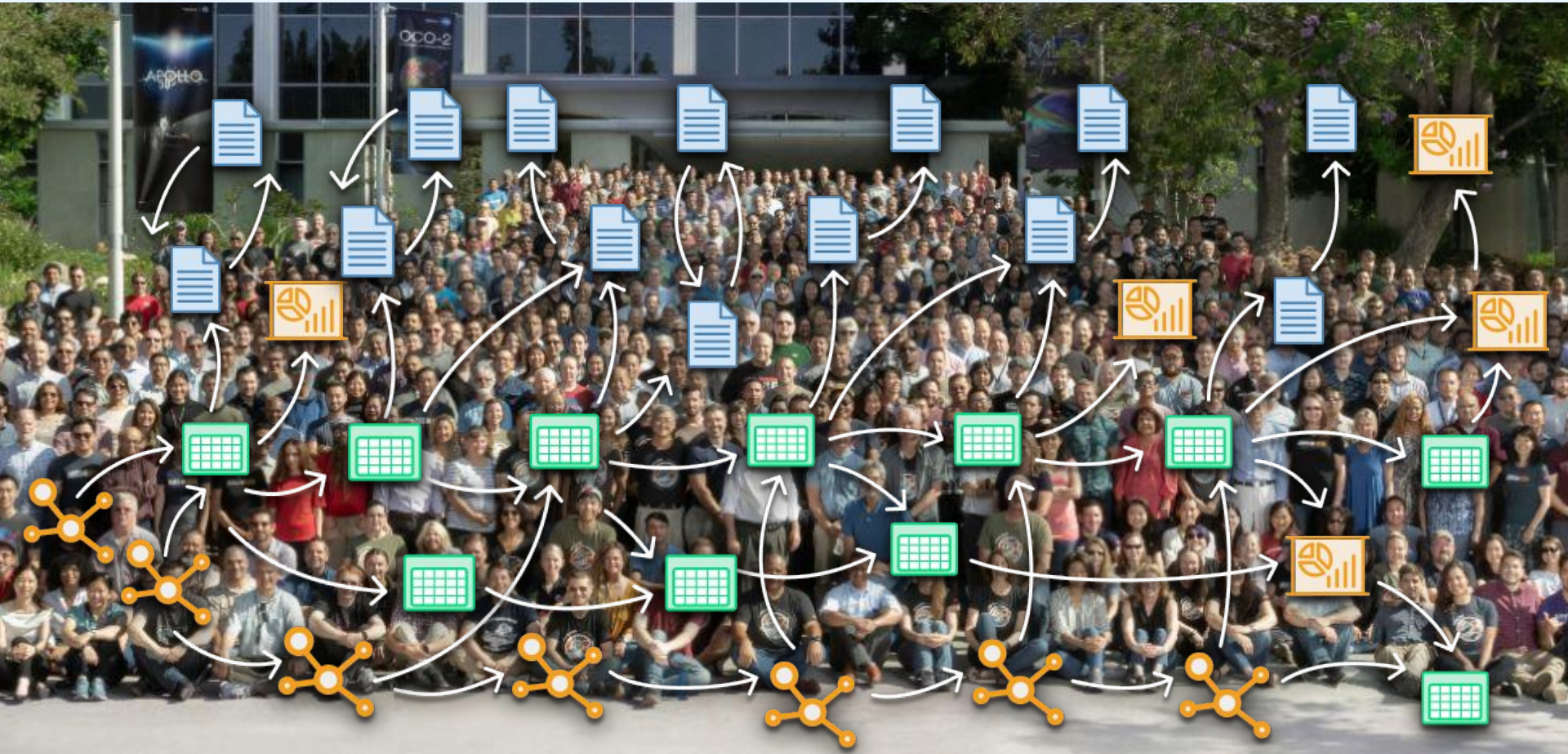
**They add it to a document.**



**And get input from others.**



And it gets complicated pretty quickly...

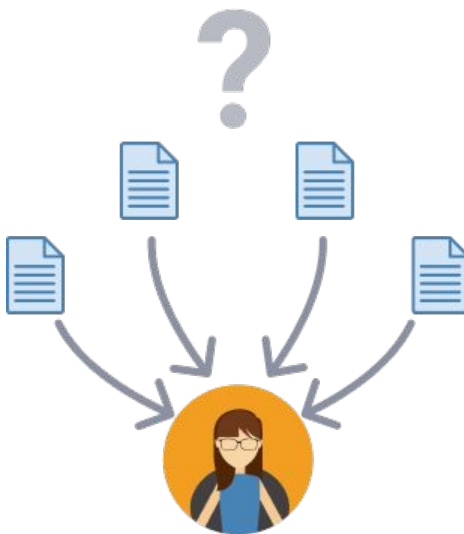




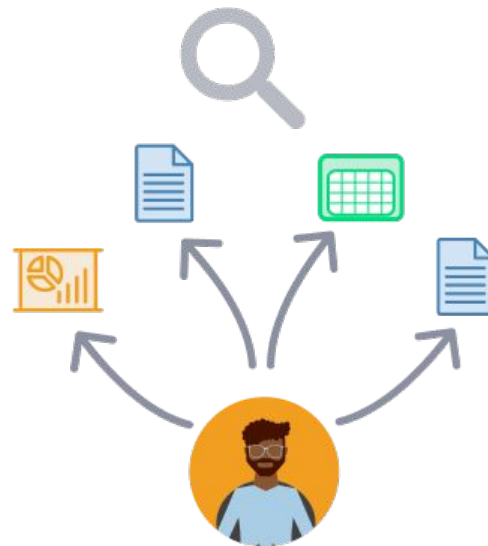
# Bad Ratio of Real engineering vs. overhead



Repetitive Data Entry

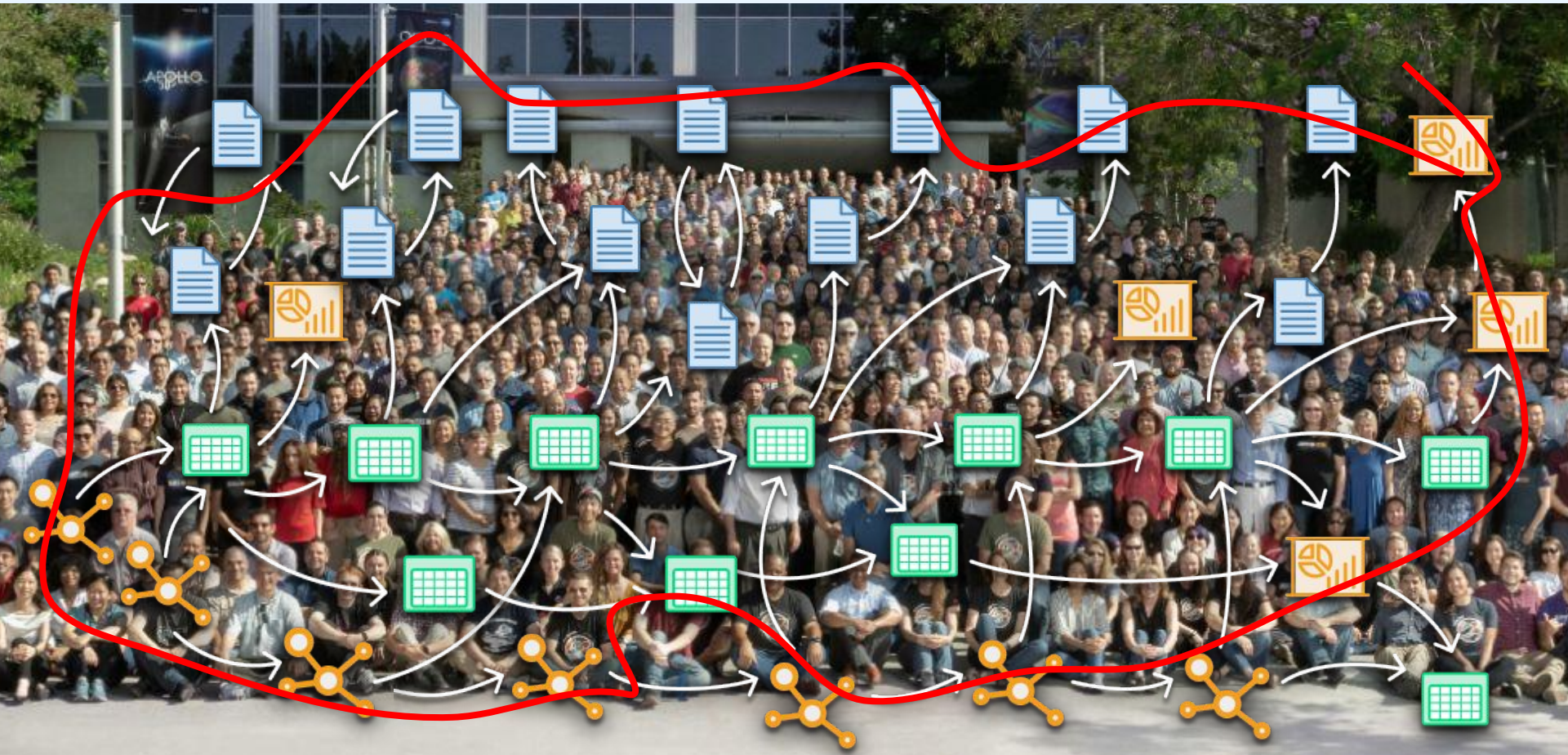


Version Confusion



Constant Searching

# How do we connect all of these things together?



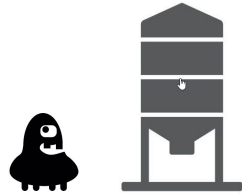
# Dragons beating Monsters

Systems engineering the development process

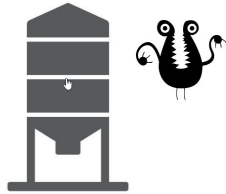
# Monsters occur and reproduce

---

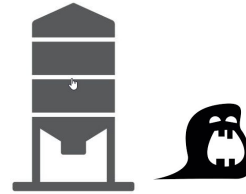
- Ever growing complexity of spacecraft
  - More functions
  - More hardware
  - More software
- Observation that silos (the Monsters) occur, for example:



Flight Software



Ground Data System

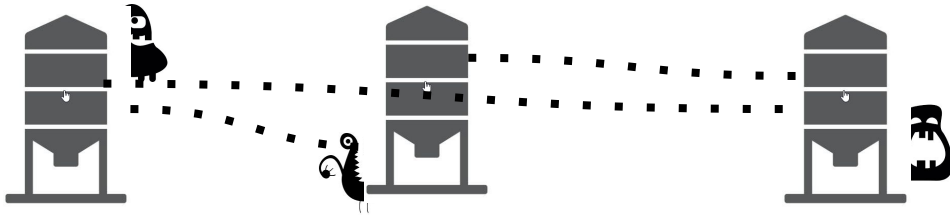


Remote Engineering Unit

# Monsters have Implicit connections

---

- Hard to understand relationships between silos, e.g. how to check requirements against test
- Difficulty when something changes and perform impact analysis across the board
- Monsters can be resolved by using **models** to capture relationships – turns monster into a solution



Flight Software

Ground Data System

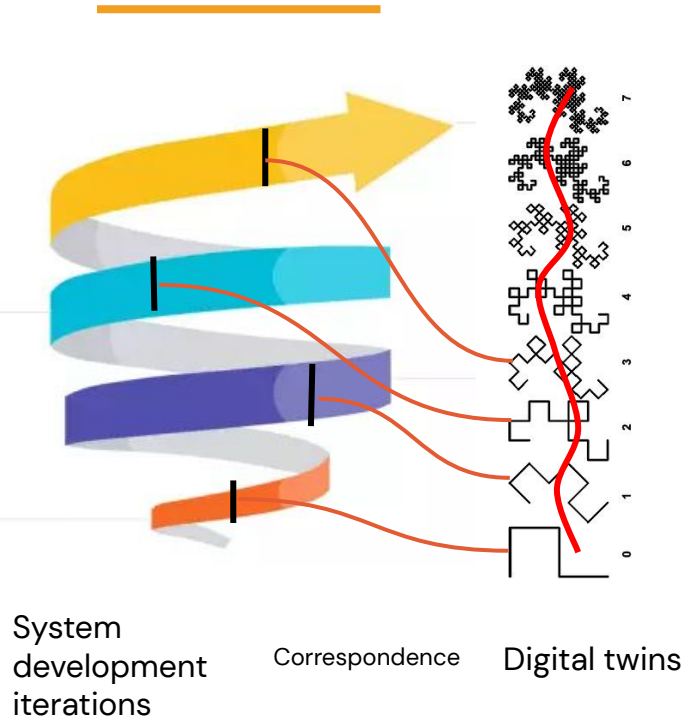
Remote Engineering Unit

# Monsters live in a discontinuous space

---

- **Overhead** to connect everything manually limits breaking up the silos
- **Disconnected** areas, implicit (e.g. in Excel).
- Qualification process does not work systematically

# Digital twin pipelines evolve with system development



**Digital twin** - progression of detail, fidelity and clarity of what we build - the product, and produce companion products of system, e.g. model of system under control

**Progress towards implementation until we have high enough fidelity so we can build the system - software defined systems**

**Qualification** - property of twin,  
Connect twins only when qualified

**Lifecycle axis** - how twins connect up  
Progression of pipelines to complete the qualification of the system for flight, e.g. MBSE with Model checking & Simulation  
Each of fractal branches is a digital twin product

# Follow an evolutionary pattern to keep things connected

---

- **Expand pipelines** evolving them as a digital twin
- Manage **Change packages**
- **Fractal approach**: Each qualification step remains valid over time or elaborated, e.g. scenarios in simulation are still valid for SW testing later on; levels of requirements
- **Spiral build** of system representation



# Make process of systematically connecting information a commodity

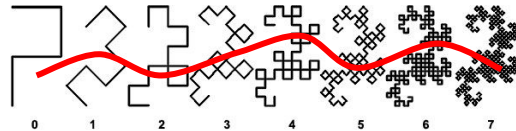
---

- Project **spiral of system development** into **fractal pipelines** with increasing detail and fidelity
- Have a degree of **qualification comprehensiveness** for each twin (increment in the spiral)
- **Each pipeline builds on top of the other** – progress in terms of fidelity

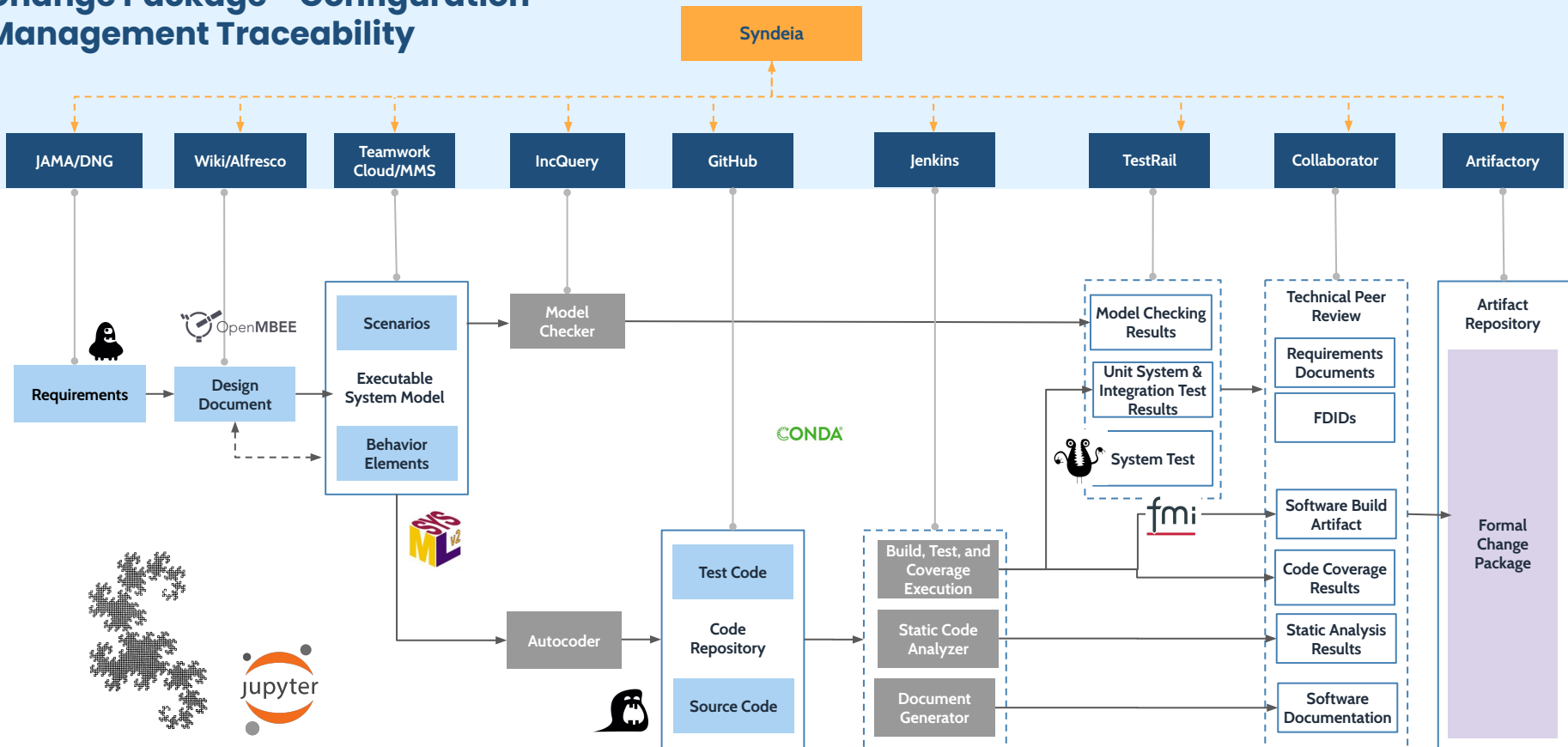
# Dragon: Formal Qualification of Systems Modeling and Software

---

- **Repeatable, executable representation** of the system and its relationships at different levels of fidelity
- **Explicit** qualification record
- Relationship between representation – the **progression**
- **Systematic** process for developing the system



# Change Package - Configuration Management Traceability



## DevOps - Continuous Integration - Simulation

Process: Issue Management • Continuous Integration • Process Orchestration

JIRA

Jenkins

XLRelease



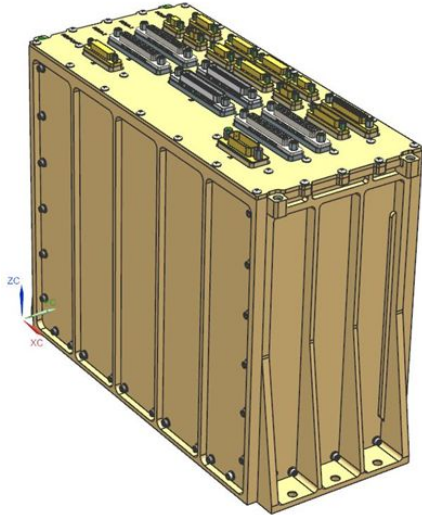
# Example of Dragon

Europa Clipper REU Use Case as a starting point

# Europa Clipper Remote Engineering Unit (REU)

## Capabilities:

- Manage and develop test suites and cases
- Trace requirements to test suites
- Analyze and report traceability matrices
- Explicitly traced and querible artifacts
  - Requirements
  - Test cases
  - Log cases
  - Test results
  - Generated reports



## Test Management

The screenshot displays the Test Management interface for the Europa Clipper - Remote Engineering Unit (REU). The page title is "Europa Clipper - Remote Engineering Unit (REU) Testing Status". The navigation menu includes "OVERVIEW", "TODO", "MILESTONES", "TEST RUNS & RESULTS", "TEST SUITES & CASES", and "REPORTS". A banner for "Free testing webinars" is visible. The main content area shows a test case titled "[REU-TEST-001-001] Example Test Case 001". Below the title, there is a table with the following data:

Type	Priority	Estimate
Compatibility	High	None
Automation Type	None	

Below the table, there is a "Steps" section with a table of test steps:

Step	Expected Result
1 [REU-TEST-001-001-001] Example Step 1	Setup Event Confirmation
2 [REU-TEST-001-001-002] Example Step 2	Enable Instrument Confirmation
3 [REU-TEST-001-001-003] Example Step 3	Instrument Data Event Confirmation

## Requirements Traceability

The screenshot displays the Requirements Traceability interface. The page title is "2.3 Consolidated Requirements and Test Cases Table". The main content area shows a table titled "Remote Engineering Unit (REU) - Consolidated Requirements and Test Cases Table". The table has two columns: "DNG LS" and "TestRail Case". The table contains the following data:

DNG LS	TestRail Case
618477.L5-REU-POR assert duration	POR Assert Length / Ready Test
618478.L5-REU-External input reset behavior	POR Latch Test - Switch Case 1
618324.L5-REU-POR visibility and persistence	POR Flag Set via 1553 Case
616487.L5-REU-POR default RTI rate	RTI Default Rate Check Case

# Nexus/OpenMBEE ViewEditor

The Nexus product is a collaborative platform for Systems Engineers to automatically generate documents and integrate data from other repositories in a coherent fashion

- Connects data to web-based engineering documents on Confluence Wiki
- Manages updates and changes to connected information
- Supports the document collaboration, review, and export process
- Provides document and requirement visualization
- Maintains the single, authoritative source in a collaborative environment

The screenshot displays the Nexus/ViewEditor interface. On the left is a navigation pane with a 'PAGE TREE' listing various documents and sequences, including 'Europa Clipper Sequence FDID Example' and 'Sequence: Flight System Requirements'. The main area shows a document titled 'Sequence: Flight System Requirements' with a 'Connected Data Table with DNG Requirements' section. This section includes a 'Query Type' dropdown set to 'Appendix Flight Systems Requirements' and a table of requirements.

ID	Requirement Name	Requirement Text	Key/Driver Indicator	Affected Systems	Maturity	Child Requirements
1040145	Sequence Restart	The Spacecraft shall be capable of restarting control programs at a ground specified point within the control program.	•		Baseline	• 905575:L4-AVS-Sequence Restart Validation • 905574:L4-AVS-Sequence Restart
1040146	Sequence Math	The Spacecraft shall support the following mathematical operations within control programs: Addition, Subtraction, Multiplication, Division.	•		Baseline	• 905573:L4-AVS-Sequencing Math Operators
1040147	Sequence Load and Execute	The Spacecraft shall be capable of separately loading and executing a sequence as separate actions.	•		Baseline	• 591577:L4-AVS-Sequence validation checks on activation • 785540:L4-AVS-Reserved sequence engines
1040148	Sequence Loops	The Spacecraft shall provide the capability to iteratively loop through sections of a sequence.	•		Baseline	• 693486:L4-AVS-Conditional sequencing logic
1040149	Sequence Wait	The Spacecraft shall provide the capability to have a sequence delay itself a programmatic amount of time.	•		Baseline	• 693486:L4-AVS-Conditional sequencing logic

# Nexus helps authoring Model Based engineering documents

- Insert rich hover references to either DNG requirements or other Content.
- Live tables for Requirements and Dictionary data
- Configuration Managed Data updates

FDID Example and Front Page

## Resources Used

is a moment ago

ordered list of commands that are executed together in order and with that timing.

ed in a sequence, and some command, sequence and accounting of this process. **756164**

**DNG Requirements** Wed, Jan 12, 2022, 7:02 PM

ID  
756164

Requirement Name  
Control Programs/ Sequences Contain Commands

Requirement Text  
are a series of commands in a specific The Spacecraft shall support all flight software commands and sequence directives within sequences.

Connected Data Table with DNG Requirements

Version: Thu Dec 09 2021 @7:02:10 PM

Query Type: Appendix Flight Systems Requirements

Parameter Value

System VAC: Sequencing

Maturity: Baseline

ID	Requirement Name	Requirement Text	Key/Driver Indicator	Affected Systems	Maturity	Child Requirements
1040145	Sequence Restart	The Spacecraft shall be capable of restarting control programs at a ground specified point within the control program.	*		Baseline	<ul style="list-style-type: none"><li>905575L4-AVS-Sequence Restart Validation</li><li>905574L4-AVS-Sequence Restart</li></ul>
1040146	Sequence Math	The Spacecraft shall support the following mathematical operations within control programs: Addition, Subtraction, Multiplication, Division.	*		Baseline	<ul style="list-style-type: none"><li>905573L4-AVS-Sequencing Math Operators</li></ul>
1040147	Sequence Load and Execute	The Spacecraft shall be capable of separately loading and executing a sequence as separate actions.	*		Baseline	<ul style="list-style-type: none"><li>591577L4-AVS-Sequence validation checks on activation</li><li>764650V1-A-AVS-Sequence</li></ul>

CAE Connected Engineering Document

Document Data Status

New updates are available every Friday at 10:00 PM

Data Type	Version	Tables
DNG Requirements	Fri, Sep 10, 2021, 8:01 PM	2

Pages / CAE CED UAT Home

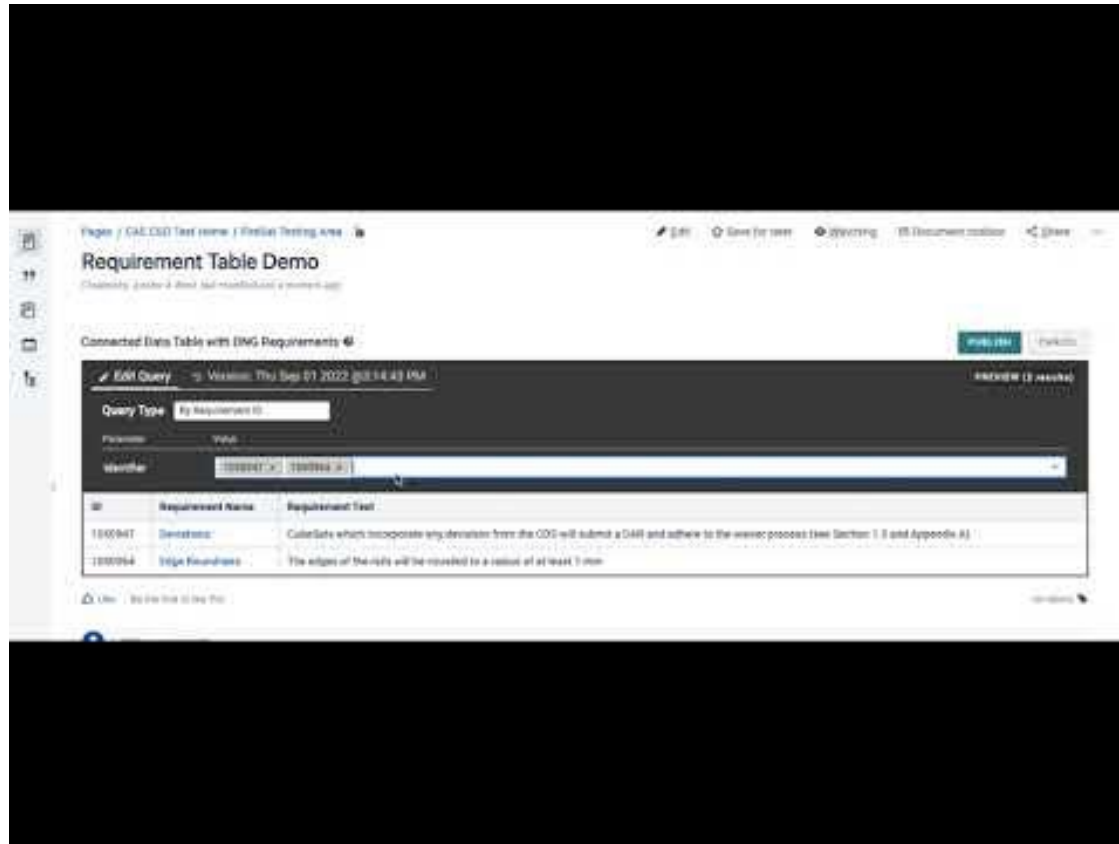
## Europa Clipper Sequence FDID Example and Front Page

Created by Marie Y. P. Gomez, last modified by Dore T. Lam on Oct 29, 2021

Here you will see an example of how a document may be populated. The following children pages are examples of sub-pages of a CED Document.

- 1. Purpose and Scope
- 2. Related Documents
- 3. ConOps and Scenarios
- 4. Hardware Description and Resources Used
- 5. Functional Behavioral Specification
- 6. Commands
- 7. Telemetry, Event Reports, Data Products
- 8. Parameters

# Demo: FDIDs as linked documents



The screenshot displays a software interface for a 'Requirement Table Demo'. At the top, there is a navigation bar with options like 'Edit', 'Save for later', 'Refreshing', 'Document outline', and 'Share'. Below this, the title 'Requirement Table Demo' is shown, along with a subtitle 'Chemistry: gascon & daniel (ed) modifiable & connectable'. A section titled 'Connected Data Table with BING Requirements' features a 'VIEW DATA' button. The main area is a query editor showing a 'Query Type' of 'By Requirement ID', a 'Filter' set to 'None', and a 'Where' clause containing 'ID IN (1000947, 1000094)'. Below the query editor is a table with two columns: 'Requirement Name' and 'Requirement Text'. The table contains two rows of data.

ID	Requirement Name	Requirement Text
1000947	Devices	GateSets which incorporate any detectors from the OGD will submit a DAF and adhere to the waiver process (see Section 3.8 and Appendix A3)
1000094	Edge Roundness	The edges of the ribs will be rounded to a radius of at least 1 mm



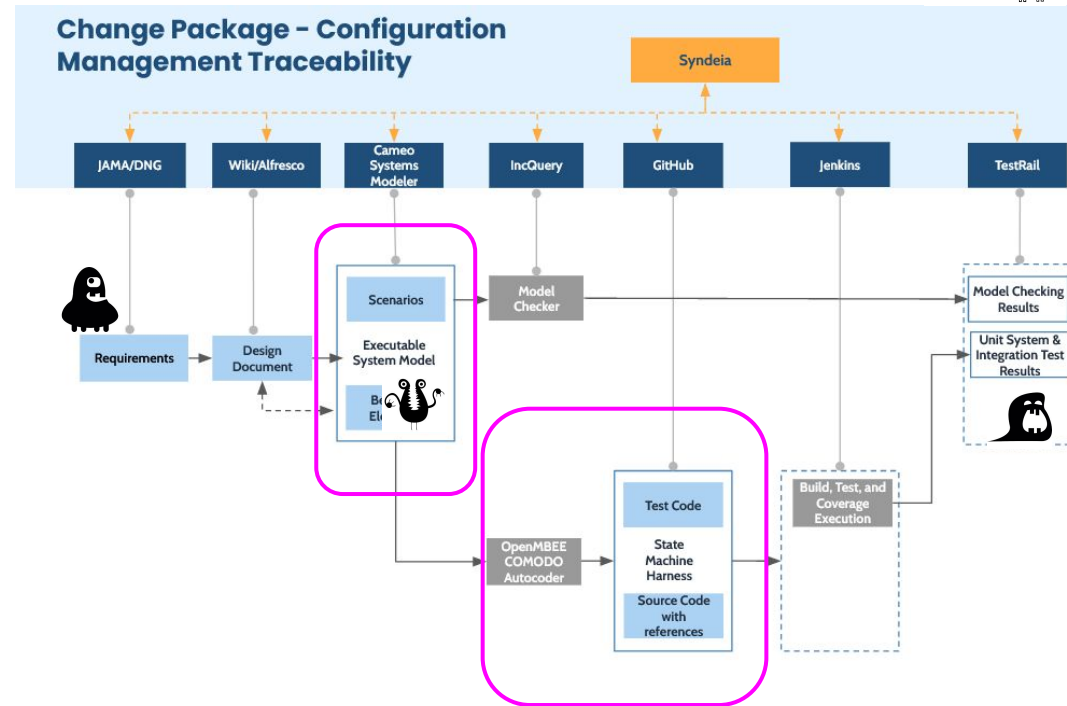


# Pipeline Prototypes

# Model Execution, Code Generation and Test

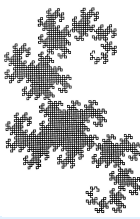


- FSW models revised to incorporate **executable simulation**
- Requirements drive **scenario tests** in the form of sequence diagrams
- Syndeia **traces requirements** and Testrail to systems model and scenario tests
- **Traverse the model** (state machine and sequence diagrams).
- **Generate** source code & test sequences.



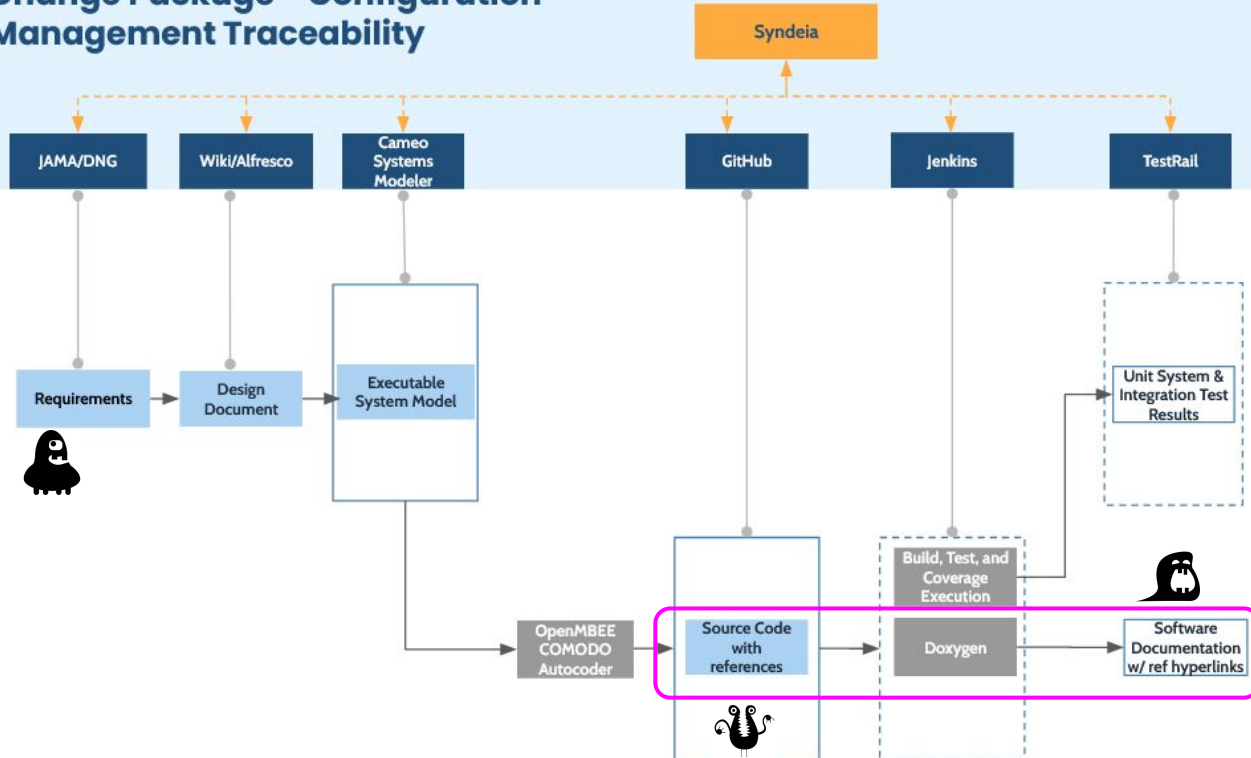


# Document Generation

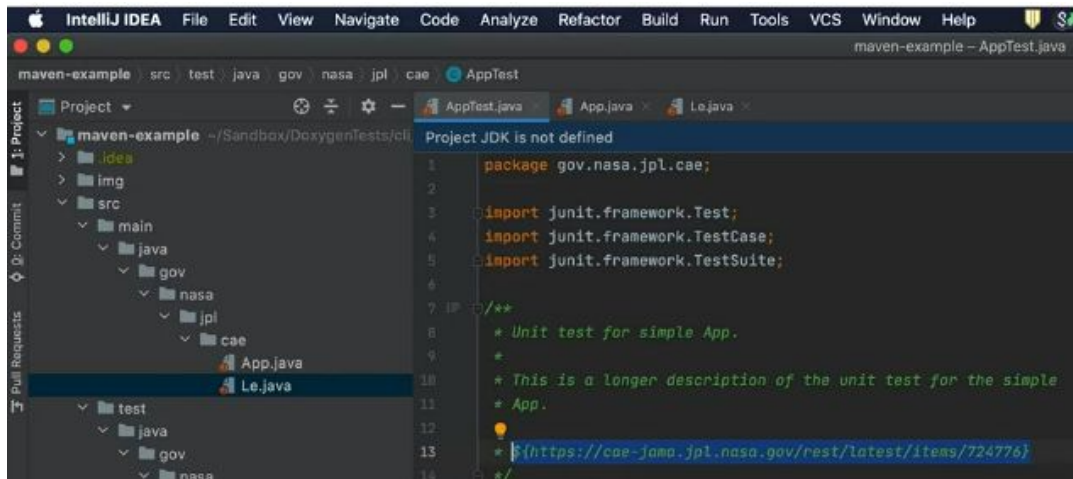


- **Detect reference** to requirement in source code files
- Create or update a **Syndeia relation** between the requirement and the source code file in GitHub
- **Display** the endpoints of this relation in the **documentation** with links to the requirement and the source code file in GitHub where it is referenced

## Change Package - Configuration Management Traceability



# Document Generation



Auto-include links into code

Generated linked documentation

## Maven example project 1.2.3

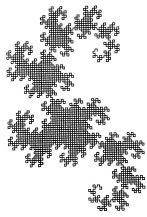
Test Project - CLI

The screenshot shows the generated linked documentation for the Maven example project. The documentation is organized into sections: Main Page, Related Pages, Packages, Classes, and Files. The Classes section is expanded, showing a list of classes including `int foo` and `int bar`. The `int foo` class is selected, and its detailed description is shown. The description includes a link to the REST API endpoint: `* This is a longer description of the unit test for the simple App.`

Repository	Link
GitHub File	AppTest.java
Requirement	Reset-to-A3R

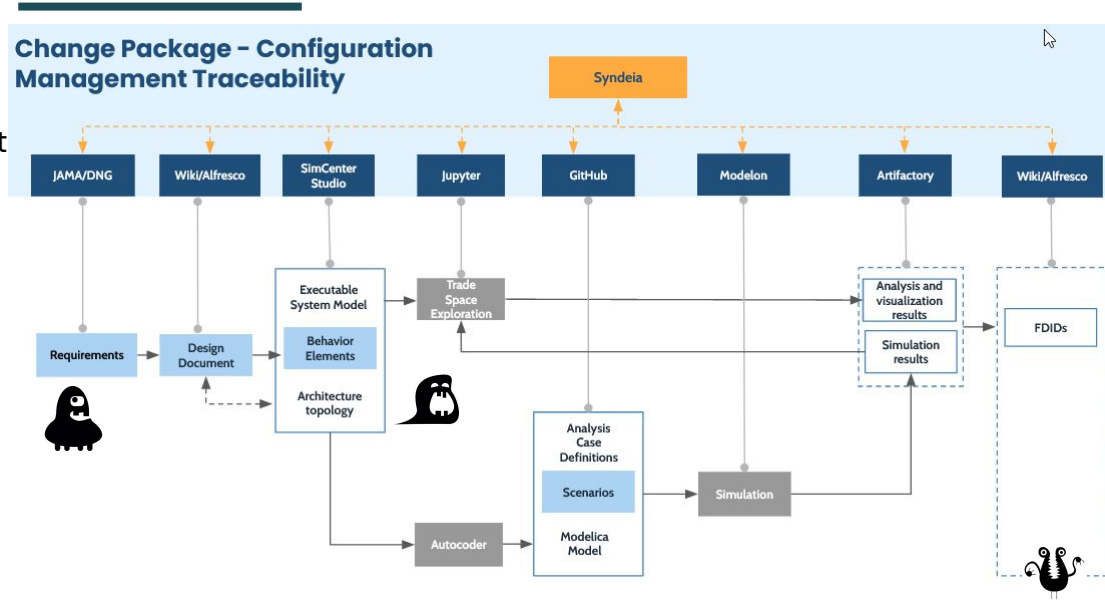


# Model Based Product Development



# Model Based Product Development Pipeline

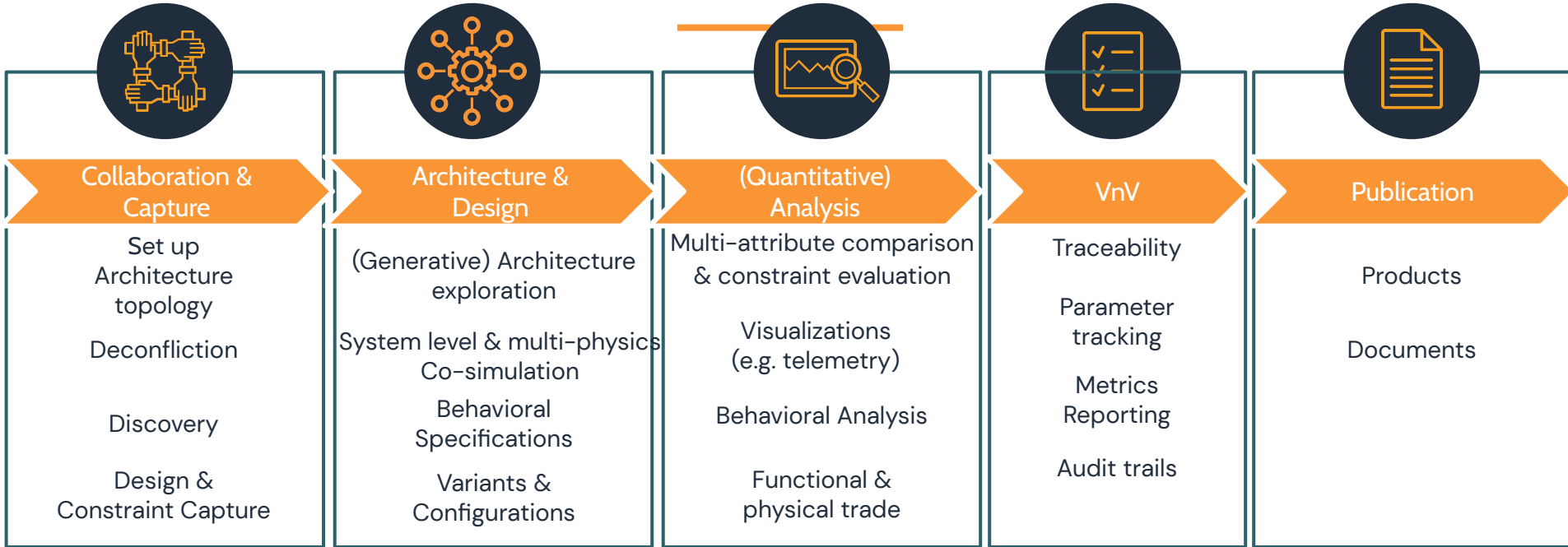
- **Cover product lifecycle** from early concept development to implementation
- Bring (executable) **systems/physics based modeling earlier** into the design
- **Evolve fidelity** of simulation as architecture matures
- Common, flexible, integrated, collaborative engineering **platform**
- **Track** parameters across product lifecycle
- Consistent set of models to **keep trade space open** longer before committing to hardware
- **Develop and validate** the architecture before buying parts
- Allow for **transition** into detailed **low level discipline specific** simulation capabilities
- Determine implementation risk





Iterative

# Workflow across product lifecycle



Issue Management

Continuous Integration

Process Orchestration



Process (Traceable, Auditable, Repeatable)





Iterative

# Collaboration & Capture



Collaboration & Capture

Set up Architecture topology

Deconfliction

Discovery

Design & Constraint Capture

```

system simple_rover:
  owns:
    Chassis
    Wheel
    PowerSupply
    Terrain

component Chassis:
  ports::group wheels:
    out::wheel_structure w1
    out::wheel_structure w2
    out::wheel_structure w3
    out::wheel_structure w4
    (out)::wheel_structure w5
    (out)::wheel_structure w6

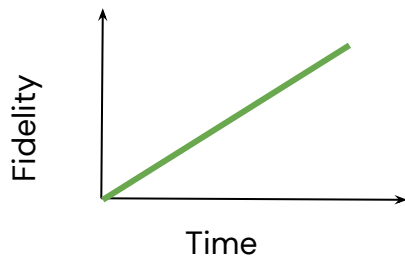
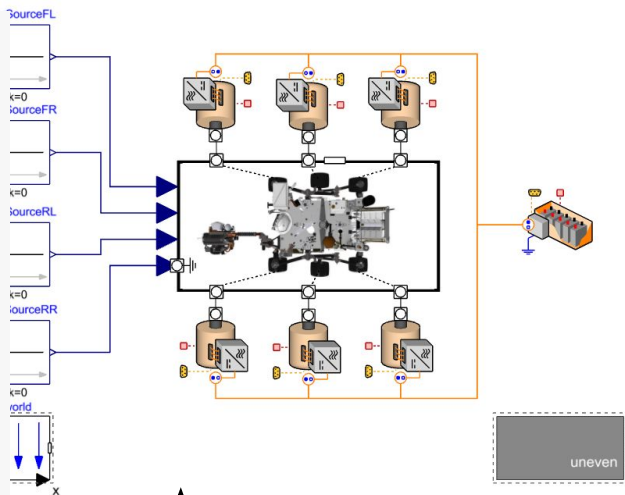
  ports:
    out::power_structure p1

component Wheel:
  properties:
    multiplicity = 4..6

  ports:
    in::wheel_structure w1
    out::terrain t1

component PowerSupply:
  ports:
    in::power_structure p1

component Terrain:
  ports::group terrains:
    in::terrain t1
    in::terrain t2
    in::terrain t3
    in::terrain t4
    (in)::terrain t5
    (in)::terrain t6
  
```



[Pages / ... / CAE Systems Environment Wiki and Discussion](#)

## Engineering Document

Created by Robert Karban just a moment ago

This document captures the design and analysis of a rover.

### MEL

Component	Mass
Chassis	500 kg
Wheel	10 kg
PowerSupply	50 kg

Architectures

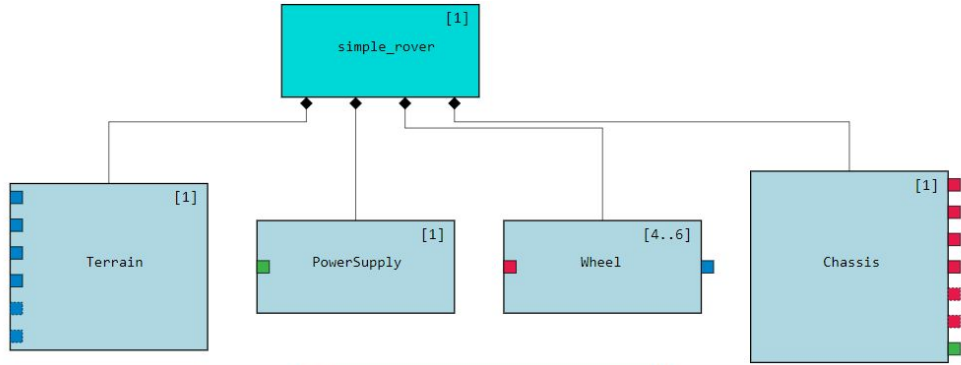
Analysis of attributes

System model

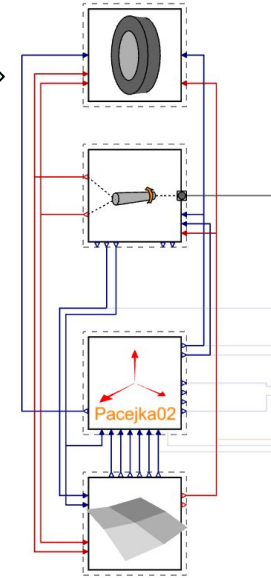
Multi-physics model

Wiki

# Architecture and Design

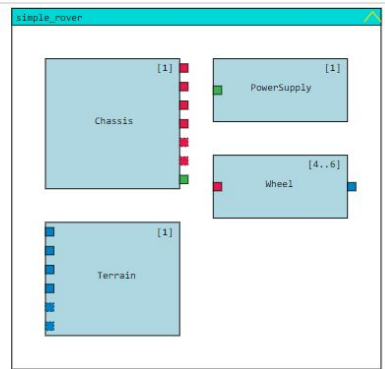


*Project functional into physical architectures*



```

component PowerSupply:
  ports:
    in::power_structure p1
component Terrain:
  ports::group terrains:
    in::terrain t1
    in::terrain t2
    in::terrain t3
    in::terrain t4
    (in)::terrain t5
    (in)::terrain t6
    
```



```

=====
Run: simple_rovers -
-----
Creating ensembles...
... done in 0.136seconds.
- 3 total ensembles
... Instantiating ensembles...
... 100.00% instantiated in 0.024 seconds
- 3 ensembles
... done in 0.170 seconds, maximum tree width = 0, maximum multiplicity = 6.
-----
Creating architectures...
- 3 unique architectures in 3.560 seconds
... done in 3.561 seconds.
    
```



## Architecture & Design

- (Generative) Architecture exploration
- System level & multi-physics Co-simulation
- Behavioral Specifications
- Variants & Configurations

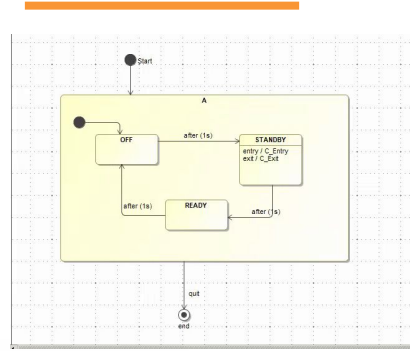
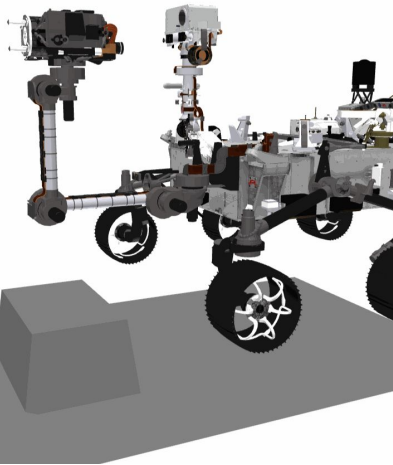
- Articulation of Design Space
- Determine required connectivity constraints

# Quantitative Analysis

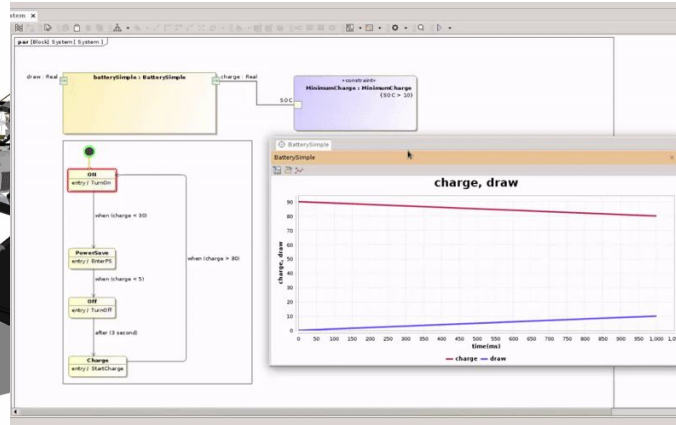
- Baseline Analysis
- Adding new Components
- Swapping Components

Guided Architecture Selection in an integrated engineering environment

System -  
Multi-physics  
Co-simulation



FSW  
simulation



Constraint  
evaluation



Quantitative Analysis

Multi-attribute comparison  
& constraint evaluation

Visualizations  
(e.g. telemetry)

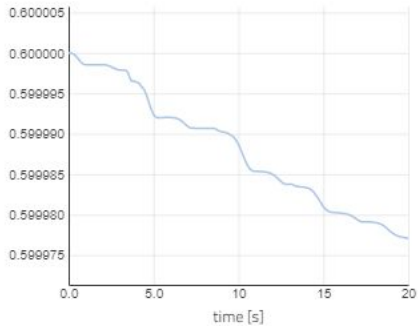
Behavioral Analysis

Functional &  
physical trade

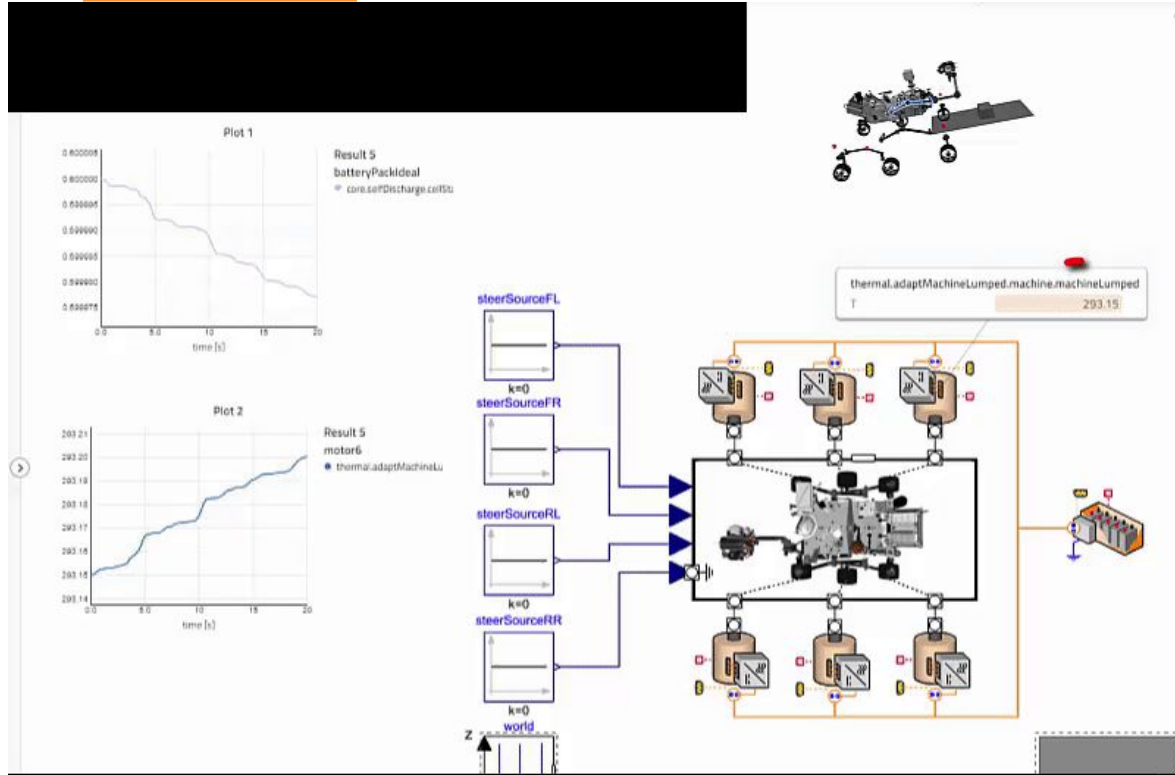
# Telemetry Visualization



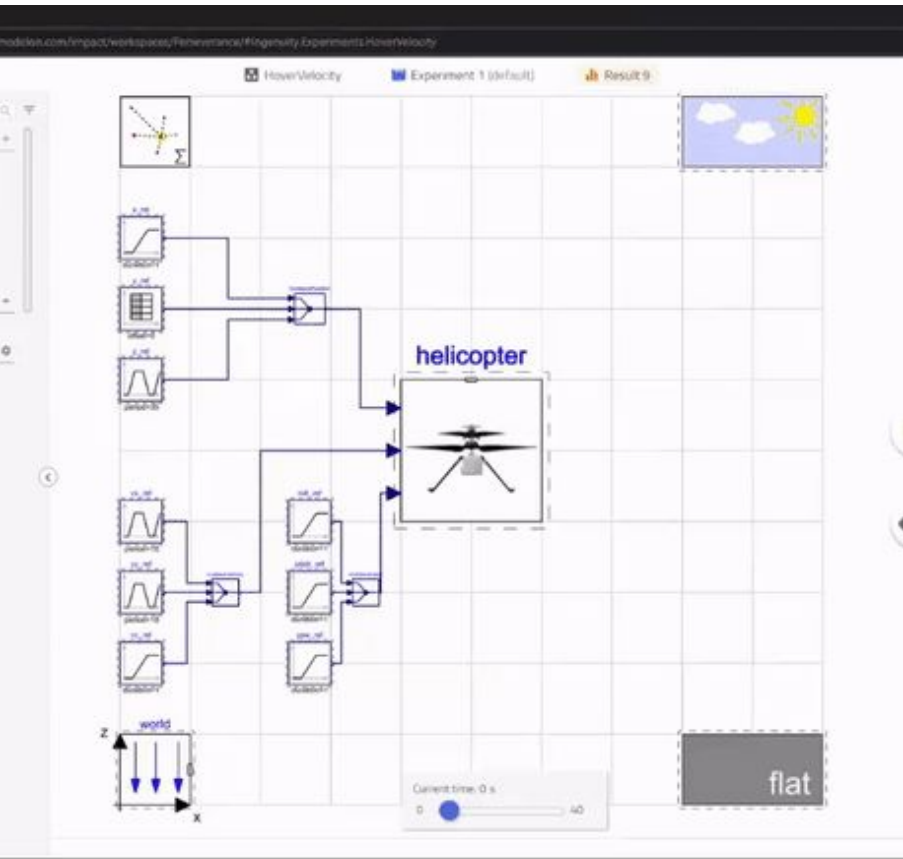
Plot 1



Result 5  
batteryPackIdeal  
● core.selfDischarge.cell5z



# Sophisticated Simulations



# Varying the physical architecture with Jupyter



For example  
Trade science  
return

by exploring

- Power topologies
- Sensing options
- Mass allocations

and  
Evaluate against  
constraints

**Create multi-physics simulation from architecture and run from Jupyter**

```
%%HTML


Previous
Next



## Simulation



In this example, Modelon is used to simulate the rover model. The rover model is constructed using the Modelica language and generic library elements from the Modelon library. The Modelon client and the Modelon server are used.



```

%%
%run getpass import getpass
token = getpass("Enter Jupyterhub API Key: ") #get token from https://en-jupyter.jpl.nasa.gov/hub/token
Enter Jupyterhub API Key: .....

%%
%run
import os
import requests

session = requests.Session()
r = session.get("https://en-jupyter.jpl.nasa.gov/user-redirect/",
headers={"Authorization": "token %s" % token})
}
if "login" in r.url:
    print("Disable token.")
elif "lab" in r.url:
    print("Server already running.")
else:
    #send "pending" msg to r.url:
    payload = {"profile": "shubml"}
    res = session.post(r.url, data=payload, headers={"Authorization": "token %s" % token})
    print("Server launching...")

Server already running.

%%
%run
import os
import requests

session = requests.Session()
# make "user-redirect" request to obtain cookie and open local user server
r = session.get("https://en-jupyter.jpl.nasa.gov/user-redirect/impact/lab",
headers={"Authorization": "token %s" % token})
}
if "login" in r.url:
    print("Disable token.")
elif "pending" in r.url:
    print("Server still launching. Please wait and try again.")
elif "open" in r.url:
    print("Server not running. Please launch the server.")
else:
    userurl = r.url.split("lab")[0]
    print("Connected to IMPACT API with URL: %s" % (userurl + "?token=" + token))
    from modelon.impact.client.spl.service import Context
    context = Context()
    context.session = session

from modelon.impact.client import Client
client = Client(userurl, context=context)

```


```

# Publication



Pages /... / CAE Systems Environment Wiki and Discussion

## Engineering Document

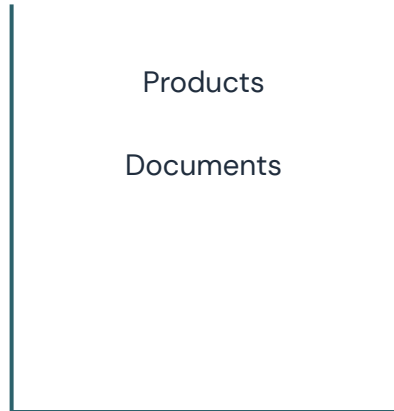
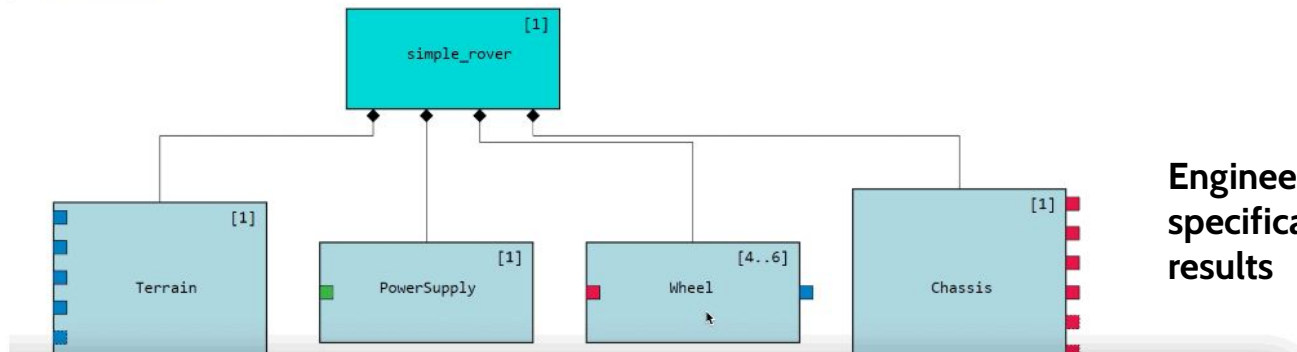
Created by Robert Karban, last modified by Myra A Lattimore just a moment ago

This document captures the design and analysis of a rover.

### MEL

Component	Mass
Chassis	500 kg
Wheel	10 kg
PowerSupply	50 kg

### Architectures



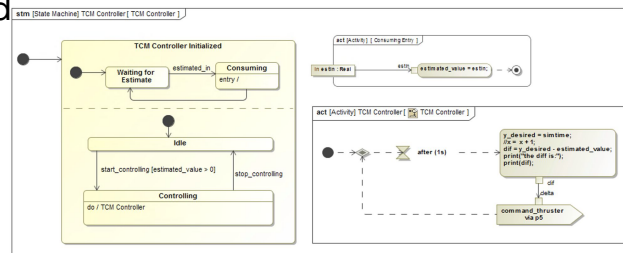
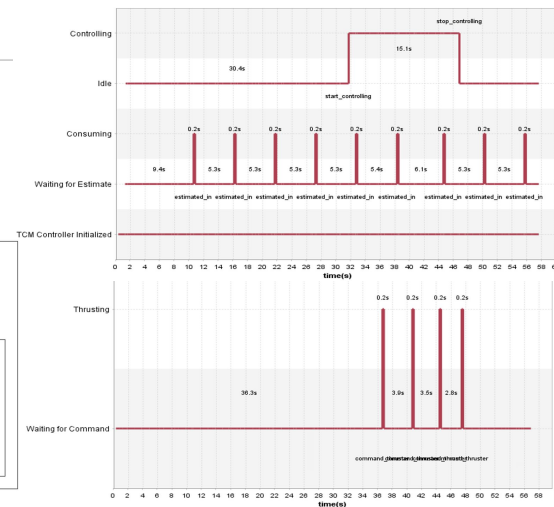
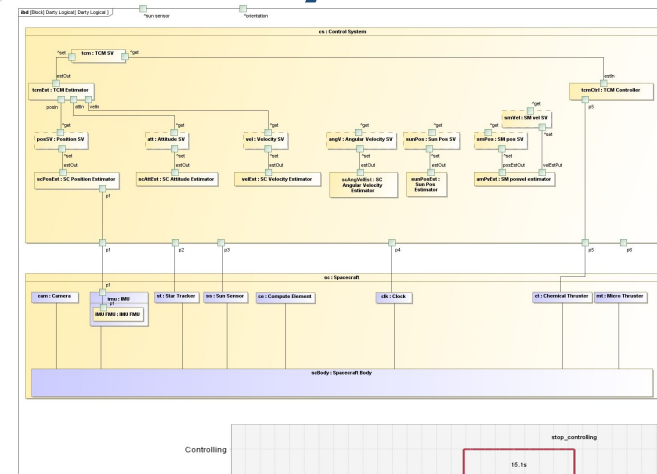
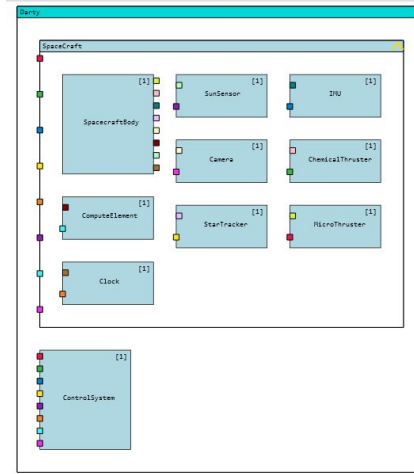
Engineering documents aggregate design specification, definition, and analysis results

# Autonomy Trades Prototype with "Darty"

- Prototype demonstrating the process with a DART like mission – crash into a small body
- Explore trade space to determine if autonomy is needed and which architectures require it

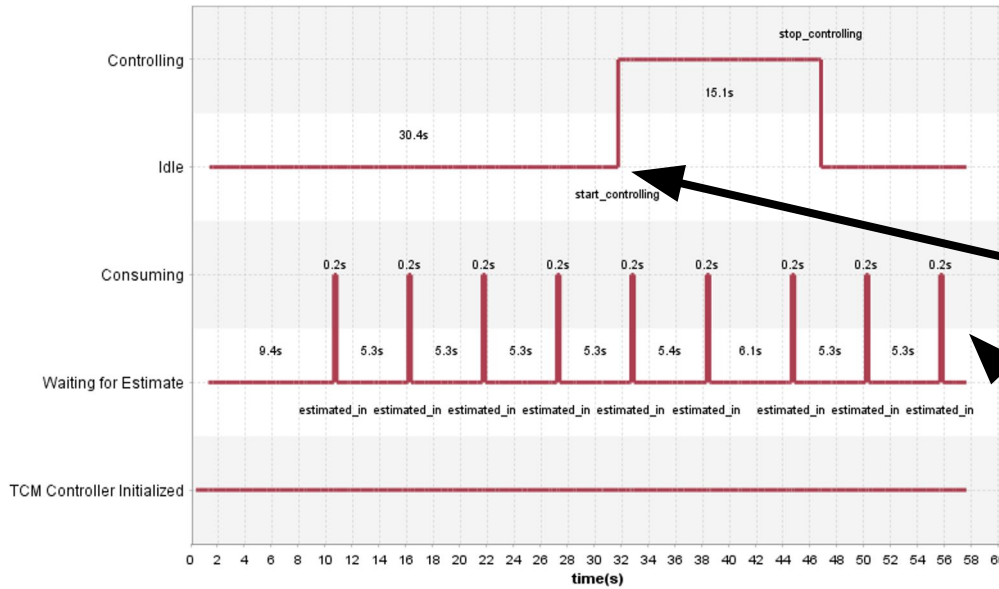
## Steps:

- Articulate architecture space with different types of thrusters
- Generate architecture structures as SysML models
- Elaborate required control system behavior with a goal-based architecture in an executable SysML model
- Capture multi-physics model of spacecraft in Modelica
- Co-simulate System level behavior and multi-physics with different scenarios
- Determine when trajectory correction maneuvers require autonomy



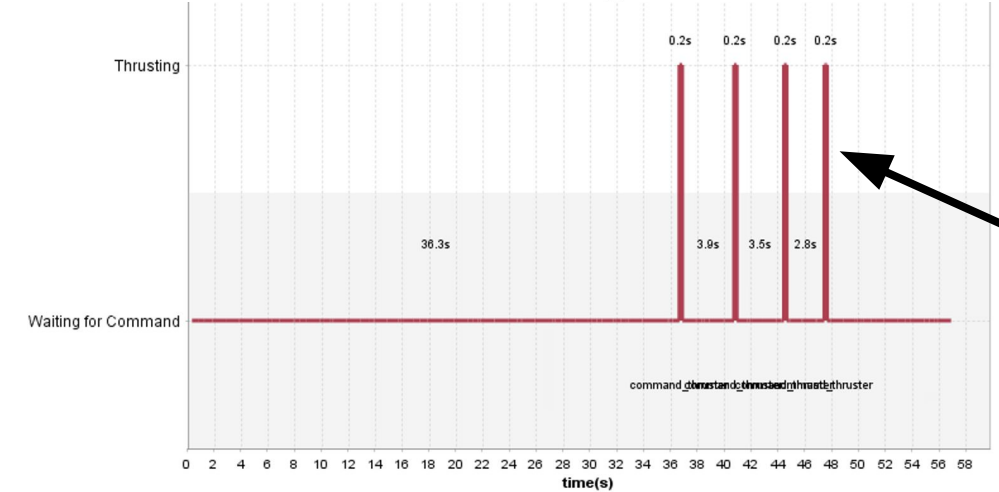


# Simulation Results - State Machines



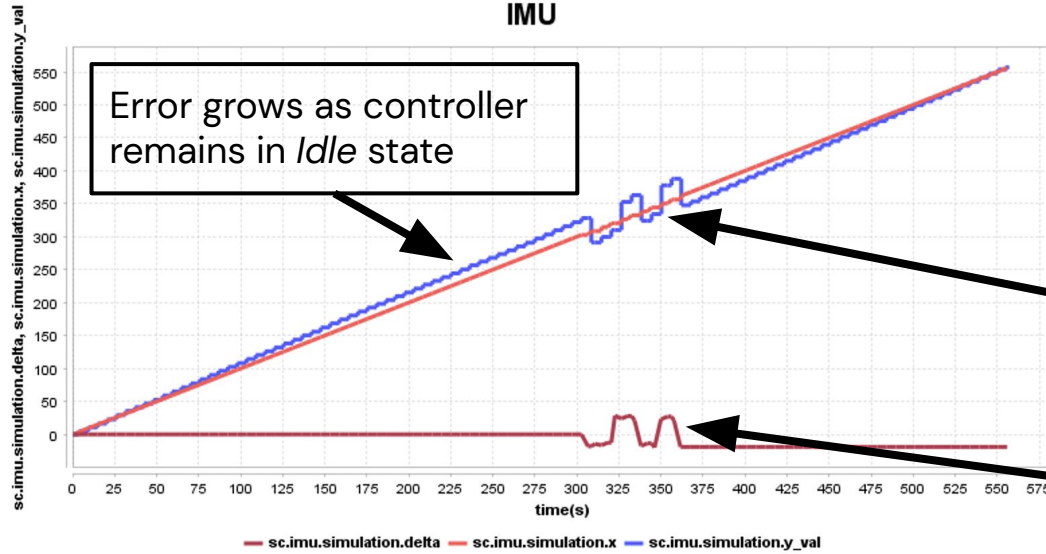
Control is applied upon receipt of the *start\_controlling* signal

Controller is always consuming new measurements as they arrive from the the estimator



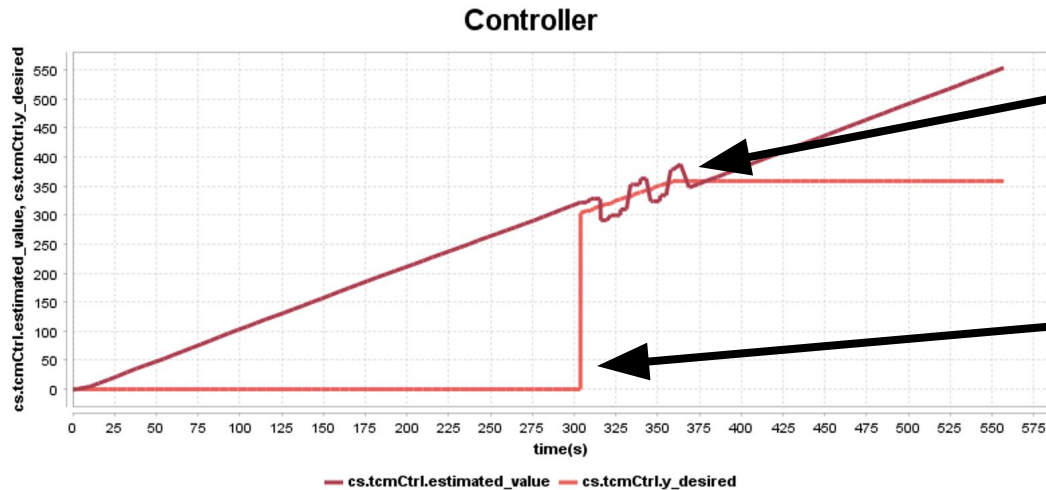
Thruster is active when controller is in the *Controlling* state and provides *command\_thruster* signal

# Simulation Results - Value properties



Measured value responds to changes induced by the thruster [low fidelity sim produces oscillations around desired value]

Correction delta is applied by thruster when system is in the *Controlling* and *Thrusting* states



Estimated value responds to changes induced by the thruster when in *Thrusting* state.

Initially, controller is in *Idle* state so no desired value is calculated. Desired value is calculated when controller is in *Controlling* state. Desired value remains unchanged after controller returns to *Idle*.

# Topic: Enabling Technology

We are almost there

fmi



ONDA

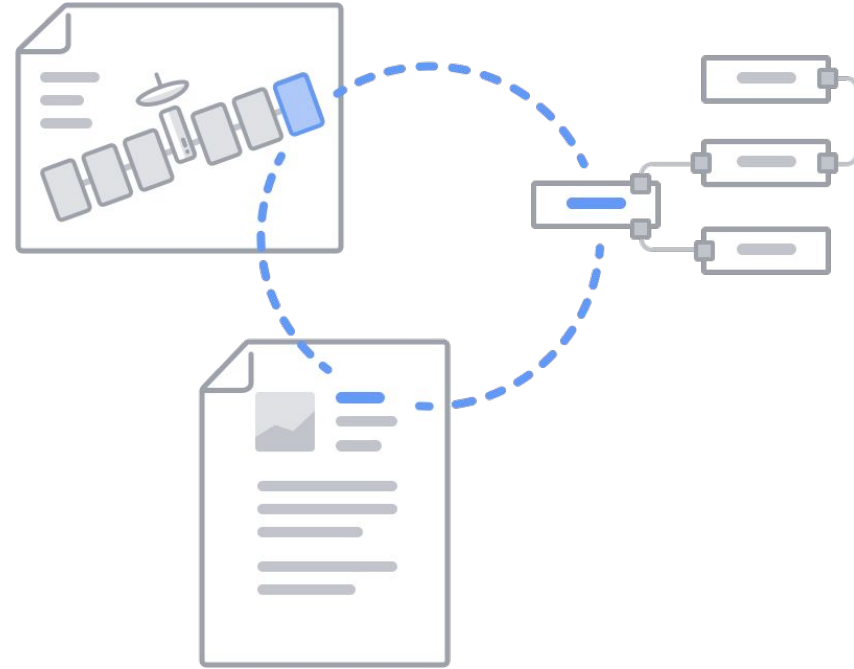


OpenMBEE

# 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,  
...
- > 500 members



Linked Data Documents with OpenMBEE

# Ecosystem 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

J



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...

The image displays a collage of Jupyter Notebook interfaces. The top-left window shows a document titled "In Depth: Linear Regression" with introductory text about regression models. Below it, a code editor shows a simple linear regression model: `x = 10 + 2 * y`. The bottom-left window shows a Julia notebook with code for plotting data and calculating eigenvalues. The bottom-middle window shows a Python notebook with code for plotting the Lorenz attractor and solving differential equations. The bottom-right window shows an R notebook with code for plotting iris data. The top-right window shows a visualization of Seattle weather data from 2010-2015, including a scatter plot of temperature and a bar chart of precipitation. The middle-right window shows a launcher interface with icons for Python 3, C++11, C++14, C++17, Julia 1.0, phylogenetics (Python 3.7), and R.

# SysML v2 Key Elements

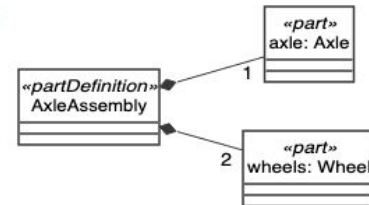


- New metamodel that is not constrained by UML
  - Grounded in formal semantics
- Robust visualizations based on flexible view & viewpoint specification and execution
  - Equivalent textual and graphical
- Standardized API to access the model

```
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]:



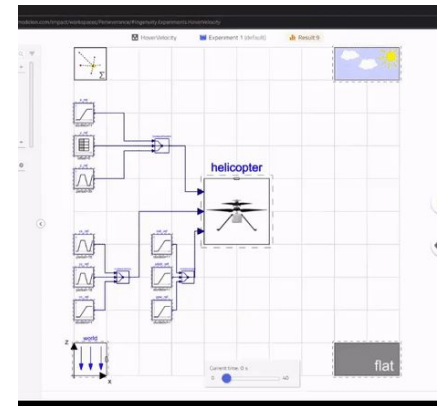
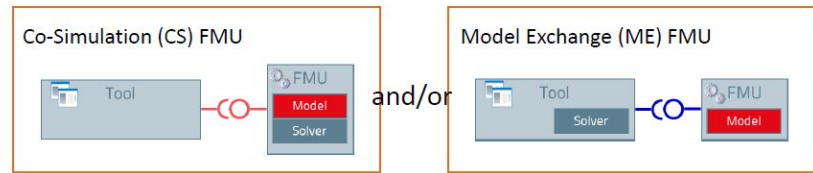
# Functional Mock Up Interface

- 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



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).

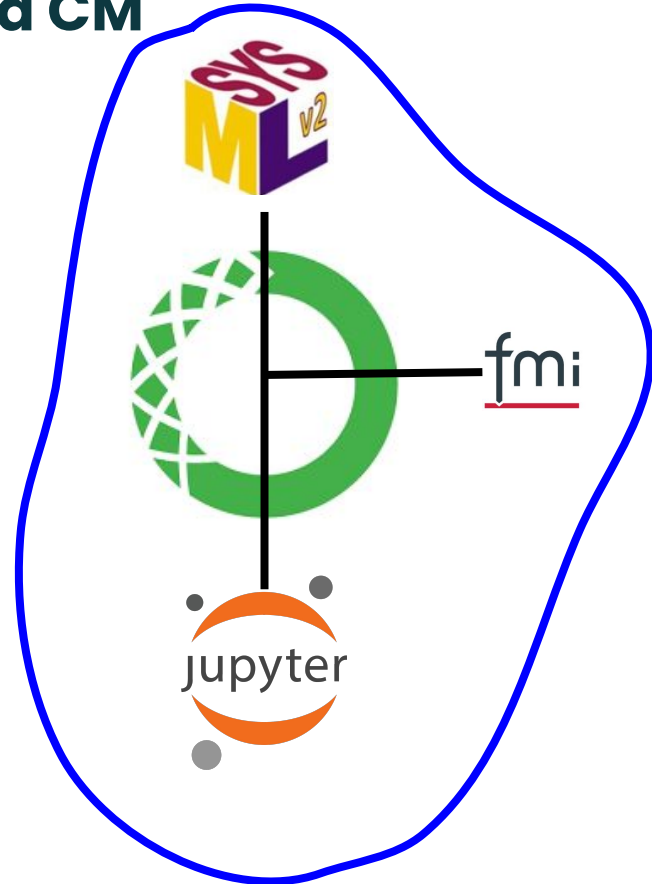




# Conda enables packaging of and CM

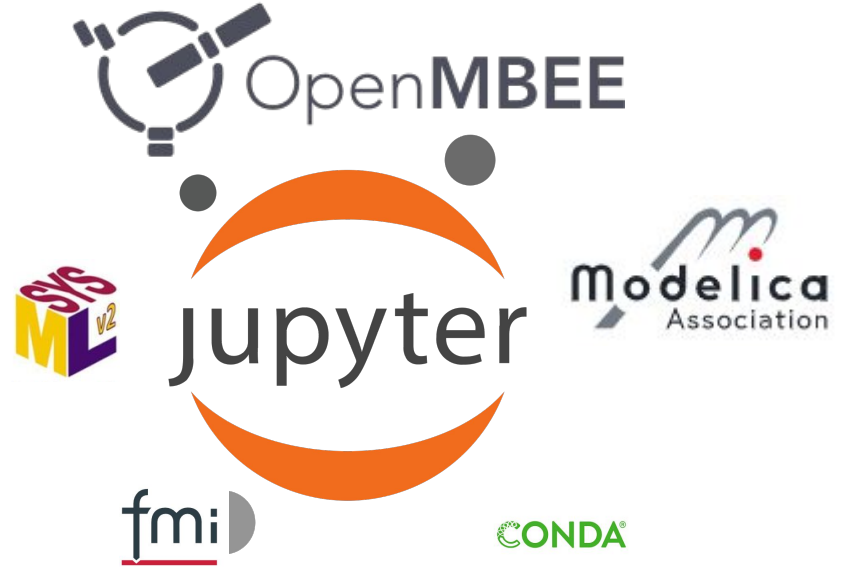
---

- 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

---



# Conclusions

---

- SE paradigm shift to formal languages and automation, i.e. MBSE
- Systematic qualification and audit trail
- Close gap between engineering documents and models
- Break up the engineering silos with digital twin pipelines
- Standards based and commoditized

# Questions?

---



**Backup**