



CATIA USER SYMPOSIUM AMERICAS 2026

MAY 18-21 | RENO | NEVADA | USA



AI-Assisted MBSE with SysML v2 in Cameo 2026x

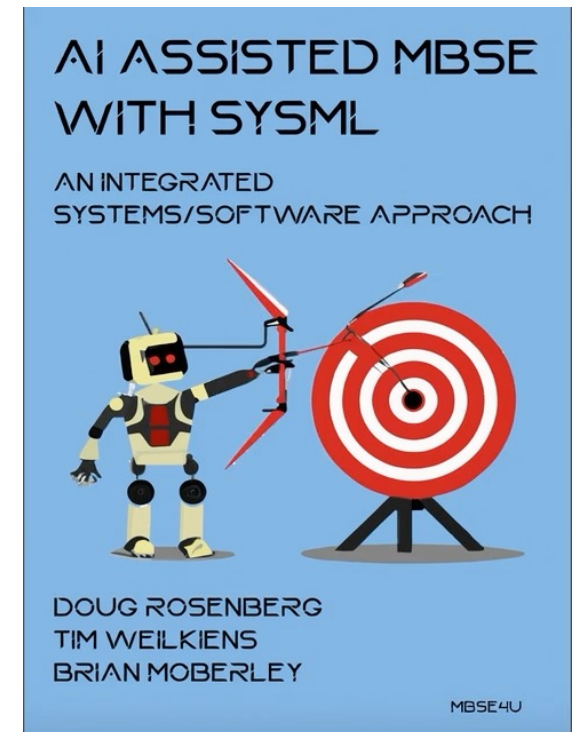
A Template-Driven Path to Radical Productivity Improvement

Doug Rosenberg – Parallel Agile, Inc. and Caltech CTME

AIM has moved from concept to reality

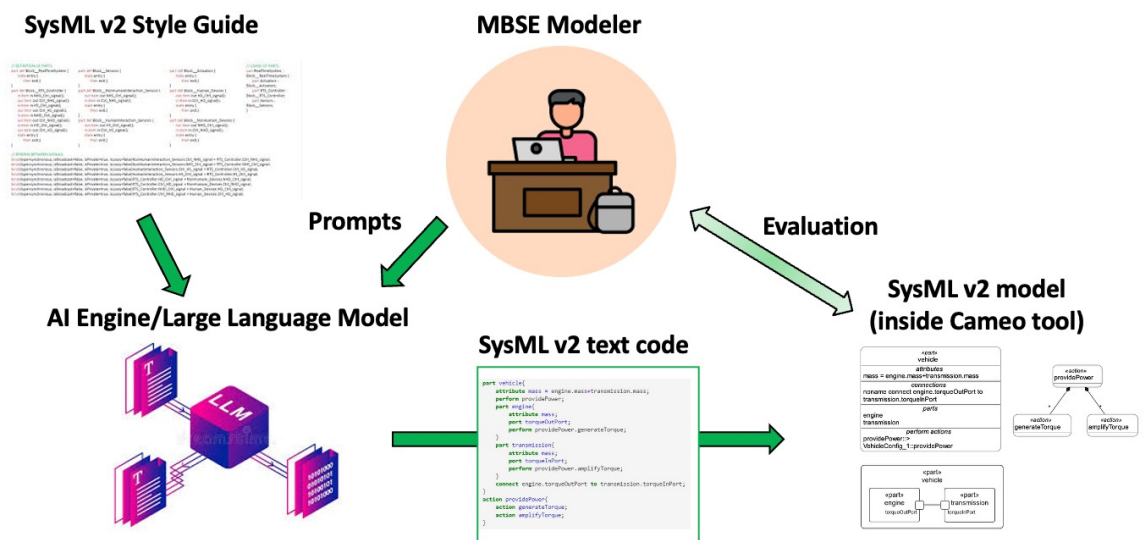
AI Assisted MBSE (AIM) is now a reality with the introduction of SysML v2 support in Cameo 2026x

AIM 2nd Edition is now being written, as the current edition was published before SysML v2 was available



Use AI to generate SysML v2 textual notation

- We explored using AI as a Subject Matter Expert (#AISME) in the AI Assisted MBSE book using an Electron Microscope
- Now, we can leverage that capability by “code generating” SysML v2 directly from AI
- Cameo 2026x can import this v2 textual notation and create SysML diagrams from it



What is the SysML v2 Style Guide and why do we need it?

- **Cameo-compliant SysML v2 code samples** for use as syntax reference with AI Code Generation
- Needed because code samples for SysML v2 are not available in quantities compared to languages like JavaScript
- Fortunately, AI is good at “follow this pattern”
- There are 9 SysML diagram types (v1), so we need a set of at least 9 code samples to match v1 capability using v2
- These samples guide AI in generating syntactically valid v2 code

```
part def DescentEngine {  
    action ignite : Ignite;  
    action adjustThrust : AdjustThrust;  
    action shutdown : Shutdown;  
}  
part def DescentFuelTanks {  
    action supplyFuel : SupplyFuel;  
    action monitorFuelLevel : MonitorFuelLevel;  
}  
part def GuidanceComputer {  
    action descendToLunarSurface : DescendToLunarSurface;  
    action ascendFromLunarSurface : AscendFromLunarSurface;  
    action dockWithCommandModule : DockWithCommandModule;  
    action undockFromCommandModule : UndockFromCommandModule;  
    action adjustCourse : AdjustCourse_Guidance;  
}  
part def TelemetrySubsystem {  
    action transmitData : TransmitData;  
    action receiveCommands : ReceiveCommands;  
}  
part def ManeuveringSubsystem {  
    action fireRCSThrusters : FireRCSThrusters;  
    action stabilizeSpacecraft : StabilizeSpacecraft;
```

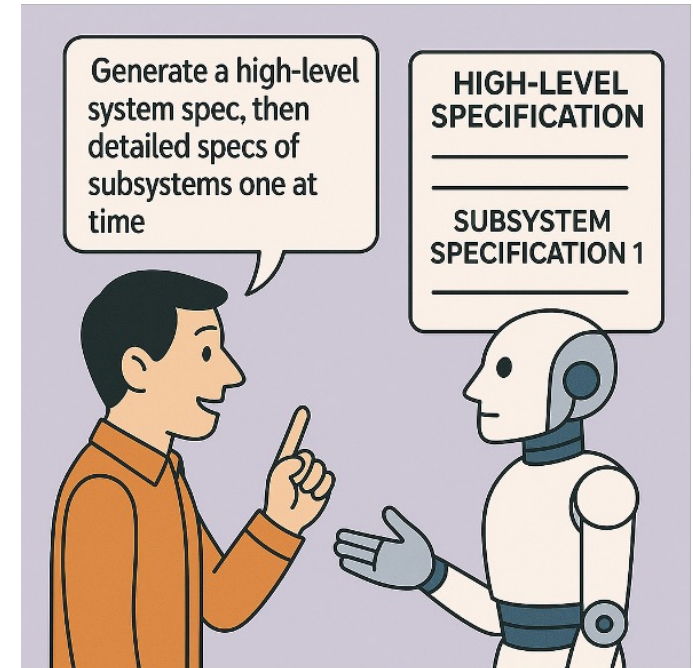
There was a significant learning curve

- We started with Cameo 2024x r3 beta
- **THANK YOU** to Osvaldas Jankauskas from Dassault for helping me get started with v2
- Debugging SysML v2 code with errors is not a lot of fun
- Note that SysML v2 compliant is not identical to Cameo-compliant
- We started teaching AIM at Caltech CTME and found success when providing students with code samples

```
package [$2] SEMRequirements {  
  
    package [$3] HighLevelRequirements {  
  
        requirement [$4] SEM_HL_01 {  
            text [$5] /* The SEM shall generate images  
        }  
  
        requirement [$6] SEM_HL_02 {  
            text [$7] /* The SEM shall produce images  
        }  
  
        requirement [$8] SEM_HL_03 {  
            text [$9] /* The SEM shall support variabl  
        }  
  
        requirement [$10] SEM_HL_04 {  
            doc [$11] /* The SEM shall control the ele  
        }  
    }  
}
```

As we gained experience, we got more ambitious

- Once our “right the first time” rate exceeded 90% we decided to try combining templates
- We tried a “system spec” template which combined Requirements, Use Cases, Domain Model, and Subsystems
- Then we tried a “logical architecture” template which included Parts, State Machine, Requirements and IBD, suitable for a subsystem at a time
- Template contents can be tailored as desired
- Currently we are investigating SysML as spec for agentic AI code generators



We used our System Spec template to generate v2 code for a lunar lander

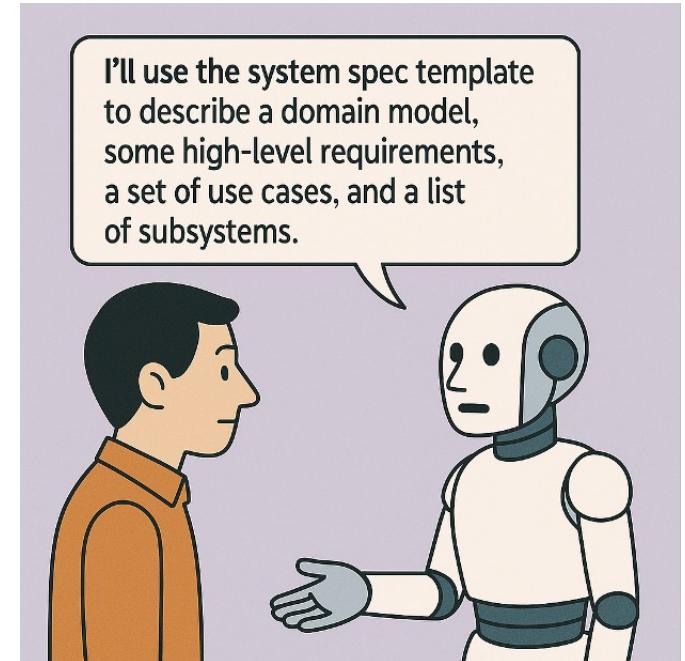
Lunar Lander V2 Model

1. Domain Model
2. Glossary
3. Top-Level Requirements
4. Top-Level State Machine
5. Top-Level Subsystem Decomposition
6. Ascent Stage Subsystem

- Part Decomposition
- Requirements
- State Machine
- Connections

7. Descent Stage Subsystem

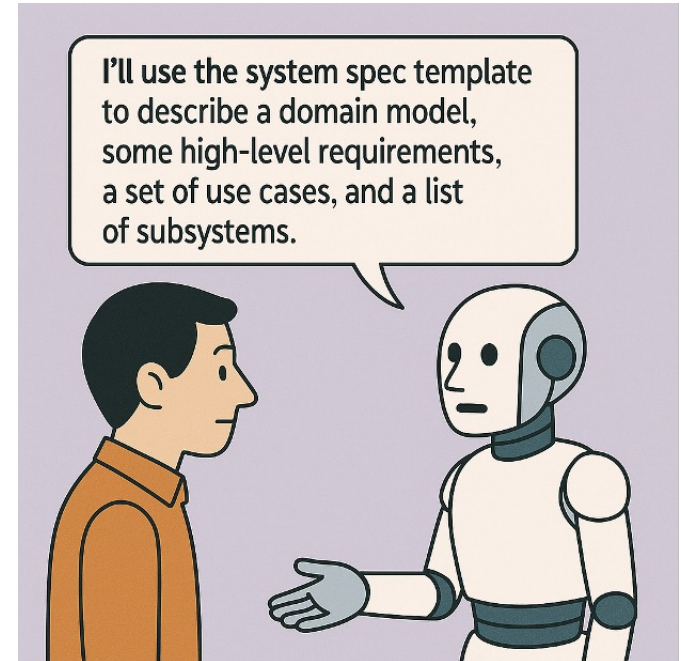
- Part Decomposition
- Requirements
- State Machine
- Connections



The complete example is published as Appendix B of Pain-Free MBSE

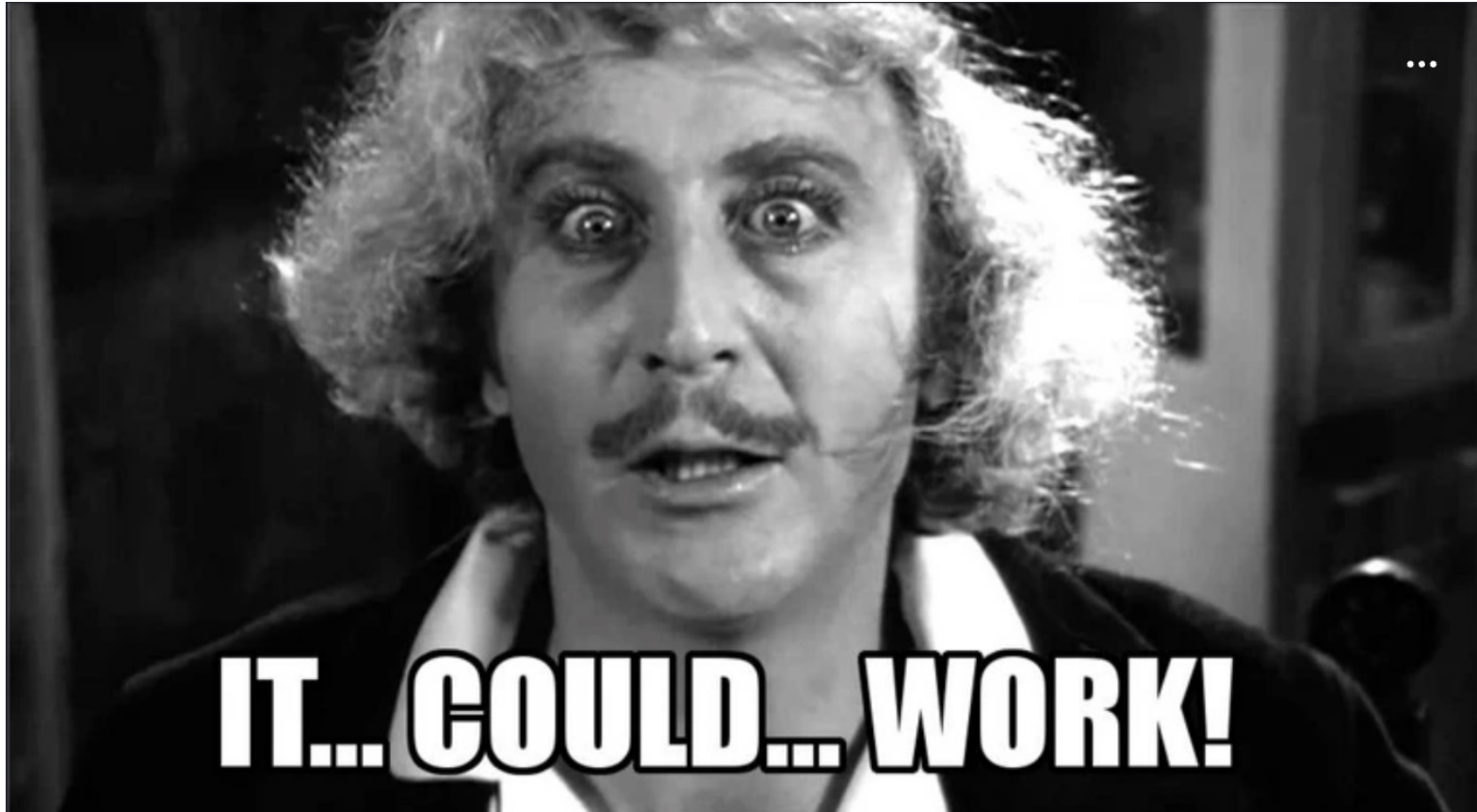
One prompt to generate the System Spec

Using the System Spec template for reference, generate a high-level spec for a Lunar Lander



The complete example is published as Appendix B of Pain-Free MBSE

One prompt for a high-level spec???



Lunar Lander Domain Model

```

1 package 'Lunar Lander V2 Model' {
2   package LunarLanderDomain_Grouped {
3     ///////////////////////////////////////////////////////////////////
4     // Top-level conceptual groupings
5     ///////////////////////////////////////////////////////////////////
6     part def LunarLander {
7       part ascent      : Ascent;
8       part descent     : Descent;
9       part communications : Communications;
10      part docking      : Docking;
11      part controls     : Controls;
12      part fuelSystem   : FuelSystem;
13      part crewCabin    : CrewCabin;
14    }
15
16    // ---- Group: Ascent ----
17    part def Ascent {
18      part ascentEngine : Engine;
19      part reactionControlThrusters : ThrusterSet;
20    }
21
22    // ---- Group: Descent ----
23    part def Descent {
24      part descentEngine : Engine;
25      part landingGear   : LandingGear;
26      part footpads      : FootpadSet;
27      part contactProbes : ContactProbeSet;
28      part radarAltimeter : RadarAltimeter;
29    }
30
31    // ---- Group: Communications ----
32    part def Communications {
33      part antenna       : Antenna;
34      part rfTransceiver : RFTransceiver;
35    }
36  }

```

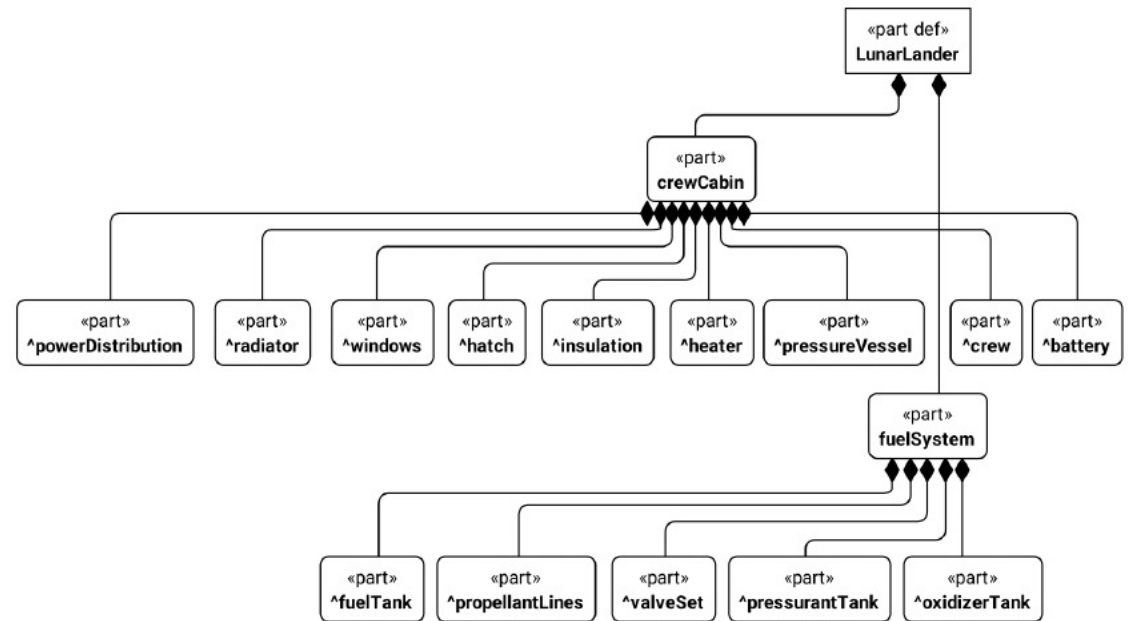


Figure 8. Lunar Lander Domain Model (General View)

AI produces a Glossary with definitions

Term	Description
Antenna	Transmits and receives radio signals between the lander, the CSM, and Earth.
AscentEngine	Provides main thrust for lunar ascent and orbital insertion.
Battery	Stores electrical energy for vehicle power.
CommandServiceModule	Companion spacecraft in lunar orbit used for docking and crew transfer.
ContactProbes	Detect surface contact just before landing.
Crew	Astronauts operating the lunar lander during mission phases.
DescentEngine	Provides throttleable thrust for powered descent and landing.
DisplaysAndControls	Present vehicle status and accept crew input for manual control.

Top Level System Requirements

```

1 package 'Lunar Lander V2 Model' {
2   package LunarLanderRequirements {
3     requirement def LL_REQ_001_MissionCapability {
4       doc /* The Lunar Lander shall undock from the CSM, perform powered
5          * descent, land on the lunar surface, support surface operations,
6          * ascend to lunar orbit, and rendezvous/dock with the CSM. */
7     }
8     requirement def LL_REQ_002_CrewSupport {
9       doc /* The Lunar Lander shall support crewed operations including
10          * ingress/egress, suited operations, and accommodations during
11          * all mission phases. */
12     }
13     requirement def LL_REQ_003_GuidanceNavigationControl {
14       doc /* The Lunar Lander shall provide guidance, navigation, and control
15          * sufficient to achieve precise descent, landing, ascent, rendezvous,
16          * and docking under lunar environmental conditions. */
17     }
18     requirement def LL_REQ_004_PropulsionDescent {
19       doc /* The Lunar Lander shall provide throttleable main propulsion for
20          * controlled powered descent to the lunar surface. */
21     }
22     requirement def LL_REQ_005_PropulsionAscent {
23       doc /* The Lunar Lander shall provide ascent propulsion sufficient to
24          * achieve lunar orbit insertion and rendezvous with the CSM. */
25     }
26     requirement def LL_REQ_006_AttitudeControl {
27       doc /* The Lunar Lander shall provide attitude control in all mission
28          * phases, including fine pointing for landing and docking. */
29     }
30     requirement def LL_REQ_007_LandingSystem {
31       doc /* The Lunar Lander shall provide a landing system that enables
32          * stable touchdown and maintains vehicle stability on expected
33          * lunar surface conditions. */
34     }
35   }
36 }

```

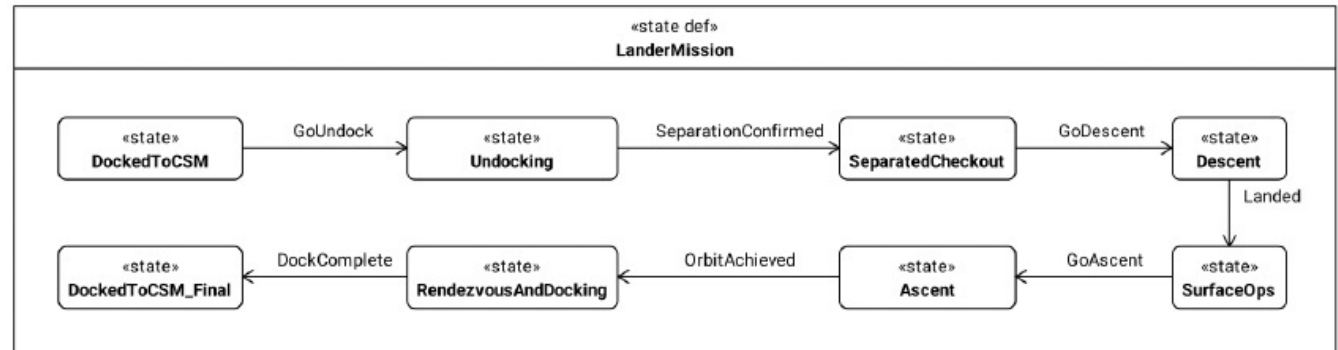
«requirement def» LL_REQ_001_MissionCapability doc The Lunar Lander shall undock from the CSM, perform powered descent, land on the lunar surface, support surface operations, ascend to lunar orbit, and rendezvous/dock with the CSM.	«requirement def» LL_REQ_002_CrewSupport doc The Lunar Lander shall support crewed operations including ingress/egress, suited operations, and accommodations during all mission phases.	«requirement def» LL_REQ_003_GuidanceNavigationControl doc The Lunar Lander shall provide guidance, navigation, and control sufficient to achieve precise descent, landing, ascent, rendezvous, and docking under lunar environmental conditions.	«requirement def» LL_REQ_004_PropulsionDescent doc The Lunar Lander shall provide throttleable main propulsion for controlled powered descent to the lunar surface.
«requirement def» LL_REQ_005_PropulsionAscent doc The Lunar Lander shall provide ascent propulsion sufficient to achieve lunar orbit insertion and rendezvous with the CSM.	«requirement def» LL_REQ_006_AttitudeControl doc The Lunar Lander shall provide attitude control in all mission phases, including fine pointing for landing and docking.	«requirement def» LL_REQ_007_LandingSystem doc The Lunar Lander shall provide a landing system that enables stable touchdown and maintains vehicle stability on expected lunar surface conditions.	«requirement def» LL_REQ_008_Communications doc The Lunar Lander shall provide communications with the CSM and Mission Control appropriate to each mission phase, including command, telemetry, voice, and navigation aids.
«requirement def» LL_REQ_009_ElectricalPower doc The Lunar Lander shall provide electrical power storage and distribution adequate to support required functions for the mission duration with priority-based load management.	«requirement def» LL_REQ_010_ThermalControl doc The Lunar Lander shall maintain crew and equipment within allowable temperature ranges across the mission thermal environment.	«requirement def» LL_REQ_011_LifeSupport doc The Lunar Lander shall provide environmental control and life support, including atmosphere management, temperature/humidity control, and waste management.	«requirement def» LL_REQ_012_DockingInterface doc The Lunar Lander shall provide a docking interface compatible with the CSM for crew transfer and structural connection.
«requirement def» LL_REQ_014_AbortAndSafety doc The Lunar Lander shall provide crew-safe abort capabilities from descent, surface, and pre-ascent conditions and protect the crew from credible hazards in all phases.	«requirement def» LL_REQ_015_DataHandling doc The Lunar Lander shall acquire, process, store, and make available mission data and telemetry required for operations and post-mission analysis.	«requirement def» LL_REQ_016_Operability doc The Lunar Lander shall provide displays and controls that enable the crew to monitor and command mission-critical functions with suitable situational awareness and workload.	«requirement def» LL_REQ_017_EnvironmentalCompatibility doc The Lunar Lander shall be compatible with the lunar environment, including vacuum, dust, illumination, gravity, and radiation.
«requirement def» LL_REQ_018_PreloadAndTurnaround doc The Lunar Lander shall support configuration, checkout, and turnaround activities required to prepare the vehicle for each mission phase.	«requirement def» LL_REQ_019_Verification doc Each top-level requirement shall be verified by analysis, inspection, test, and/or demonstration as appropriate.	«requirement def» LL_REQ_013_FaultManagement doc The Lunar Lander shall provide fault detection, isolation, and recovery with health and status reporting to crew and ground.	

Top Level State Machine

```

1 package 'Lunar Lander V2 Model' {
2   package LunarLanderTopLevelStateMachine {
3     item def GoUndock;
4     item def SeparationConfirmed;
5     item def GoDescent;
6     item def Landed;
7     item def GoAscent;
8     item def OrbitAchieved;
9     item def InProximity;
10    item def DockComplete;
11    part def LunarLander {
12      exhibit state LanderMission;
13    }
14    state def LanderMission {
15      state DockedToCSM;
16      state Undocking;
17      state SeparatedCheckout;
18      state Descent;
19      state SurfaceOps;
20      state Ascent;
21      state RendezvousAndDocking;
22      state DockedToCSM_Final;
23      transition first entryAction then DockedToCSM;
24      transition t1 first DockedToCSM accept GoUndock then Undocking;
25      transition t2 first Undocking accept SeparationConfirmed then
26        ↳ SeparatedCheckout;
27      transition t3 first SeparatedCheckout accept GoDescent then Descent;
28      transition t4 first Descent accept Landed then SurfaceOps;
29      transition t5 first SurfaceOps accept GoAscent then Ascent;
30      transition t6 first Ascent accept OrbitAchieved then RendezvousAndDocking;
31      transition t7 first RendezvousAndDocking accept DockComplete then
32        ↳ DockedToCSM_Final;
33    }
34  }
35 }

```

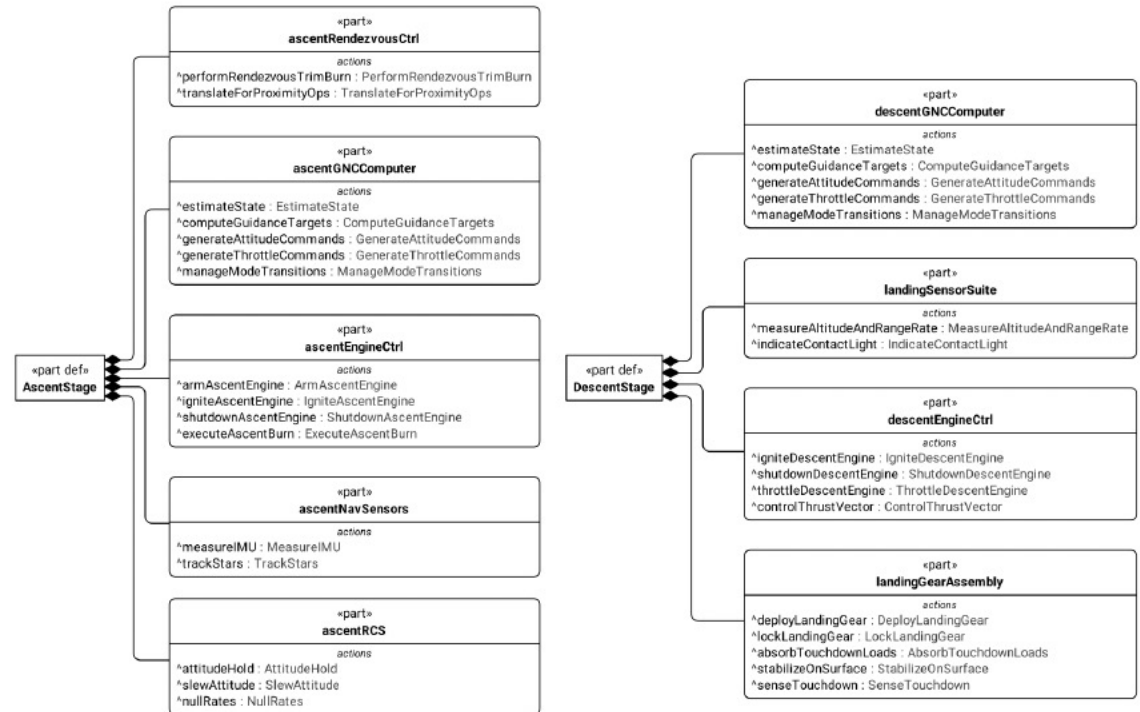


Top Level Subsystems

```

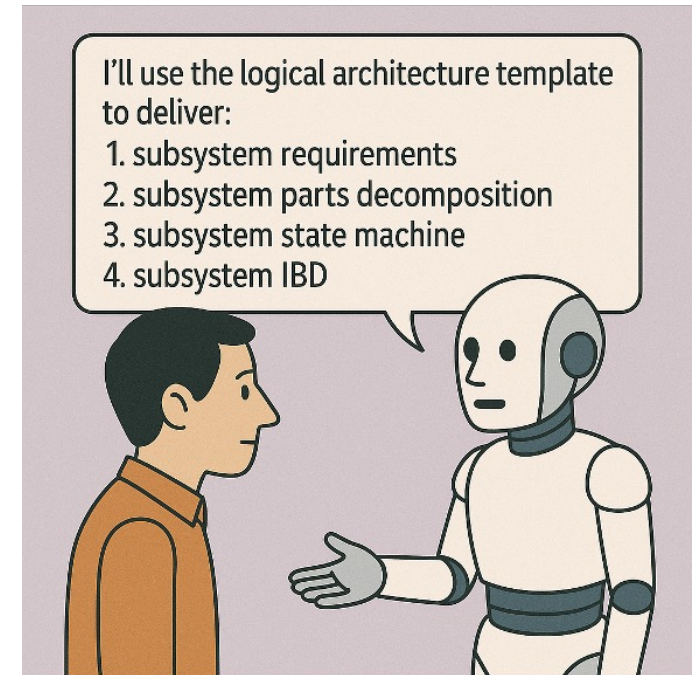
70 part def AscentStage {
71   part ascentEngineCtrl : AscentEngineControl;
72   part ascentRendezvousCtrl : RendezvousControl;
73   part ascentRCS : RCSModule;
74   part ascentGNCCComputer : GNCCComputer;
75   part ascentNavSensors : NavSensorSuite;
76 }
77 part def DescentStage {
78   part descentEngineCtrl : DescentEngineControl;
79   part landingSensorSuite : LandingSensorSuite;
80   part landingGearAssembly : LandingGearAssembly;
81   part descentGNCCComputer : GNCCComputer;
82 }
83 part def CrewCabin {
84   part eclssAir : ECLSS_AirManagement;
85   part eclssWaterWaste : ECLSS_WaterWaste;
86   part crewInterface : CrewInterface;
87   part cabinThermal : ThermalLoop;
88   part cabinPowerPanel : PowerDistributionPanel;
89   part crewProtection : CrewProtection;
90 }
91 part def CommunicationsAndTelemetry {
92   part rfTransceiverUnit : RFTransceiverUnit;
93   part commsAntennaSystem : AntennaSystem;
94   part telemetryRecorder : TelemetryRecorder;

```

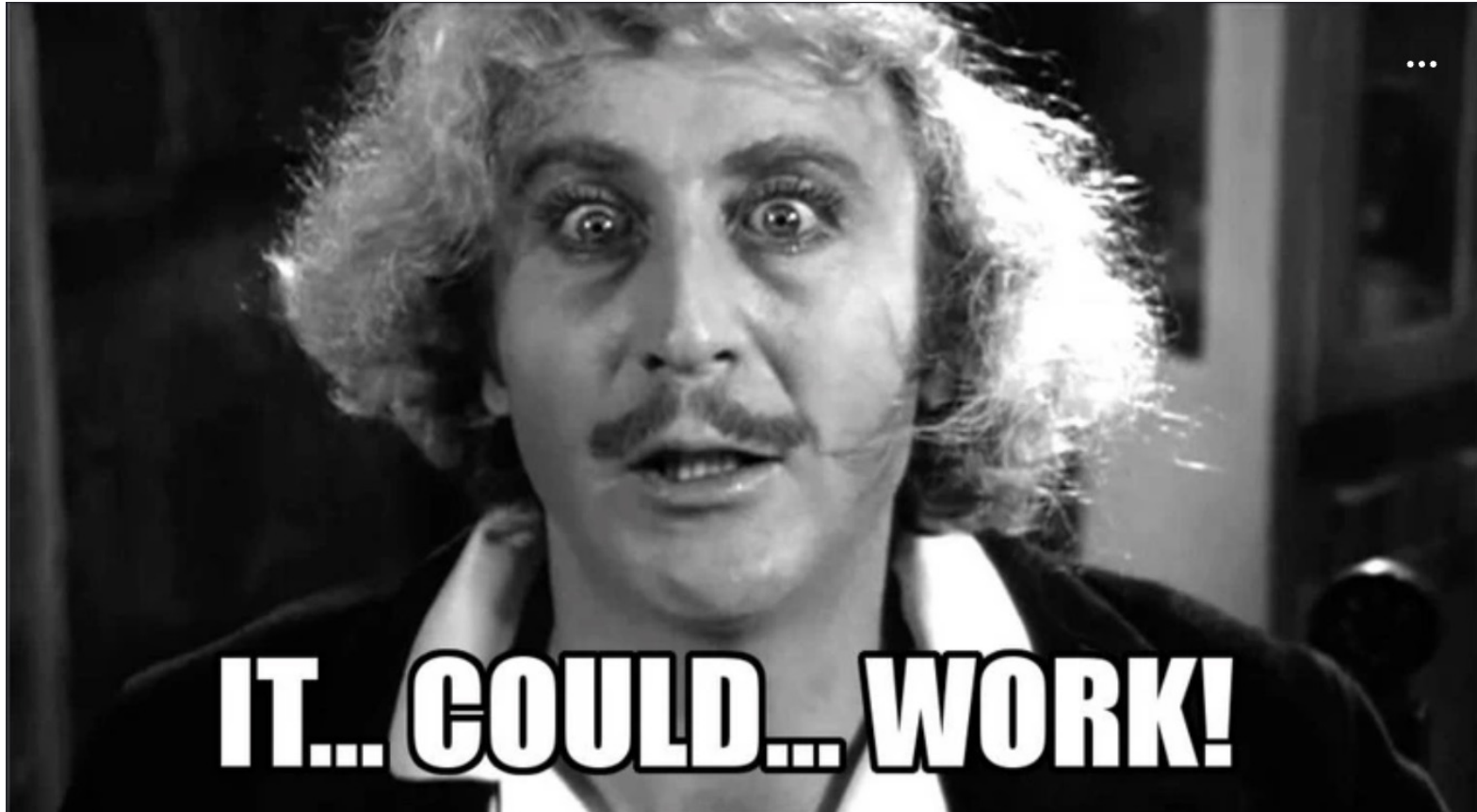


Generate Logical Architecture specs a Subsystem at a time

*Using the Logical
Architecture template
for reference, generate
a Subsystem spec for the
Ascent Subsystem of a
Lunar Lander*



One prompt per subsystem???

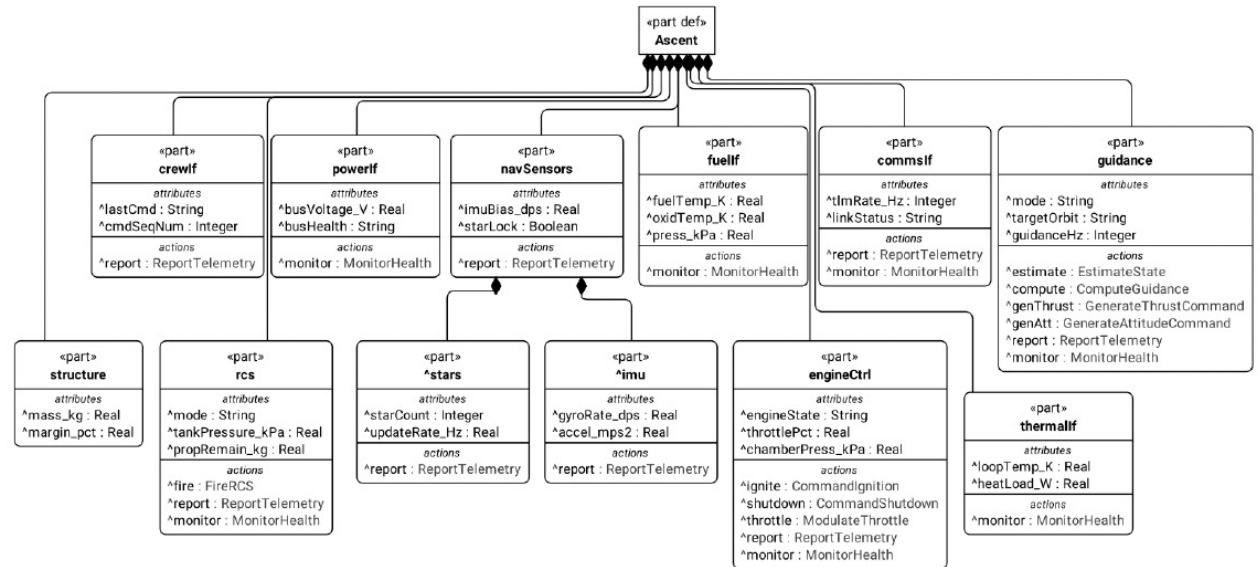


Ascent Stage Part Decomposition

```

1 package 'Lunar Lander V2 Model' {
2   package LunarLanderSubsystemsDecomposition {
3     package Ascent_Stage {
4       package PartDecomposition {
5         private import ScalarValues::**;
6         private import SI::**;
7         action def ComputeGuidance;
8         action def EstimateState;
9         action def GenerateThrustCommand;
10        action def GenerateAttitudeCommand;
11        action def CommandIgnition;
12        action def CommandShutdown;
13        action def ModulateThrottle;
14        action def FireRCS;
15        action def MonitorHealth;
16        action def ReportTelemetry;
17        part def AscentEngineControl {
18          attribute engineState : String;
19          attribute throttlePct : Real;
20          attribute chamberPress_kPa : Real;
21          action ignite : CommandIgnition;
22          action shutdown : CommandShutdown;
23          action throttle : ModulateThrottle;
24          action report : ReportTelemetry;
25          action monitor : MonitorHealth;
26        }
27        part def RCSModule {
28          attribute mode : String;
29          attribute tankPressure_kPa : Real;
30          attribute propRemain_kg : Real;
31          action fire : FireRCS;
32          action report : ReportTelemetry;
33          action monitor : MonitorHealth;
34        }
35        part def NavSensorSuite {
36          attribute imuBias_dps : Real;
37          attribute starLock : Boolean;
38          action report : ReportTelemetry;
39          part imu : IMU;
40          part stars : StarTracker;

```



Ascent Stage Subsystem Requirements

```

1 package 'Lunar Lander V2 Model' {
2   package LunarLanderSubsystemsDecomposition {
3     package Ascent_Stage {
4       package Requirements {
5         requirement def ASC_REQ_001_Capability {
6           doc /* The Ascent subsystem shall provide the capability to
7              * lift off from the lunar surface, insert into lunar orbit,
8              * and deliver the vehicle to a rendezvous-ready state. */
9         }
10        requirement def ASC_REQ_002_EngineControl {
11          doc /* The Ascent subsystem shall command ascent engine
12             * ignition, throttling or profiling as applicable,
13             * and shutdown per guidance. */
14        }
15        requirement def ASC_REQ_003_AttitudeAndRCS {
16          doc /* The Ascent subsystem shall control vehicle attitude in
17             * all ascent phases and provide translation impulses
18             * required for proximity operations. */
19        }
20        requirement def ASC_REQ_004_GuidanceNavigation {
21          doc /* The Ascent subsystem shall estimate state and compute
22             * guidance to meet target orbit conditions and rendezvous
23             * geometry within allocated errors. */
24        }
25        requirement def ASC_REQ_005_Interfaces {
26          doc /* The Ascent subsystem shall interface with power, thermal,
27             * communications, crew interface, and propellant/pressurant
28             * supplies required for ascent operations. */
29        }
30        requirement def ASC_REQ_006_FaultManagement {

```

«requirement def» ASC_REQ_001_Capability doc The Ascent subsystem shall provide the capability to lift off from the lunar surface, insert into lunar orbit, and deliver the vehicle to a rendezvous-ready state.	«requirement def» ASC_REQ_002_EngineControl doc The Ascent subsystem shall command ascent engine ignition, throttling or profiling as applicable, and shutdown per guidance.	«requirement def» ASC_REQ_003_AttitudeAndRCS doc The Ascent subsystem shall control vehicle attitude in all ascent phases and provide translation impulses required for proximity operations.
--	--	---

«requirement def» ASC_REQ_004_GuidanceNavigation doc The Ascent subsystem shall estimate state and compute guidance to meet target orbit conditions and rendezvous geometry within allocated errors.	«requirement def» ASC_REQ_005_Interfaces doc The Ascent subsystem shall interface with power, thermal, communications, crew interface, and propellant/pressurant supplies required for ascent operations.	«requirement def» ASC_REQ_006_FaultManagement doc The Ascent subsystem shall detect, annunciate, and support recovery from credible faults affecting ascent engine, RCS, and guidance control.
--	---	--

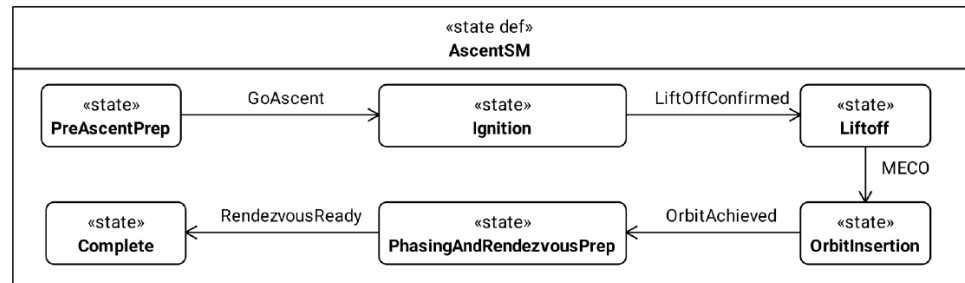
«requirement def» ASC_REQ_007_TelemetryAndCommand doc The Ascent subsystem shall provide command acceptance from crew and send ascent status and telemetry to the vehicle and external communications.	«requirement def» ASC_REQ_008_Verification doc Ascent requirements shall be verified by analysis, test, inspection, and/or demonstration as appropriate.
--	--

Ascent Subsystem State Machine

```

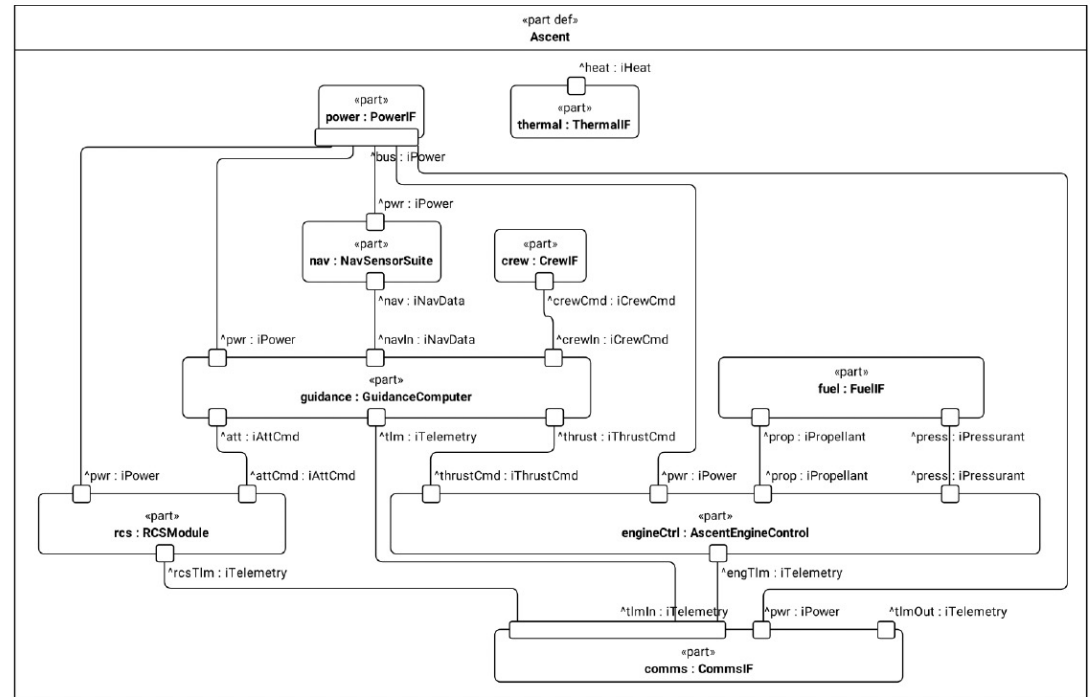
1 package 'Lunar Lander V2 Model' {
2   package LunarLanderSubsystemsDecomposition {
3     package Ascent_Stage {
4       package StateMachine {
5         item def GoAscent;
6         item def LiftOffConfirmed;
7         item def MECO;
8         item def OrbitAchieved;
9         item def RendezvousReady;
10        part def AscentSM_Owner {
11          exhibit state AscentSM;
12        }
13      }
14      state def AscentSM {
15        state PreAscentPrep;
16        state Ignition;
17        state Liftoff;
18        state OrbitInsertion;
19        state PhasingAndRendezvousPrep;
20        state Complete;
21        transition t0 first entryAction then PreAscentPrep;
22        transition t1 first PreAscentPrep accept GoAscent then Ignition;
23        transition t2 first Ignition accept LiftOffConfirmed then Liftoff;
24        transition t3 first Liftoff accept MECO then OrbitInsertion;
25        transition t4 first OrbitInsertion accept OrbitAchieved then
26          ↪ PhasingAndRendezvousPrep;
27        transition t5 first PhasingAndRendezvousPrep accept RendezvousReady
28          ↪ then Complete;
29      }
30    }
31  }
32 }

```



Ascent Subsystem Interfaces (IBD)

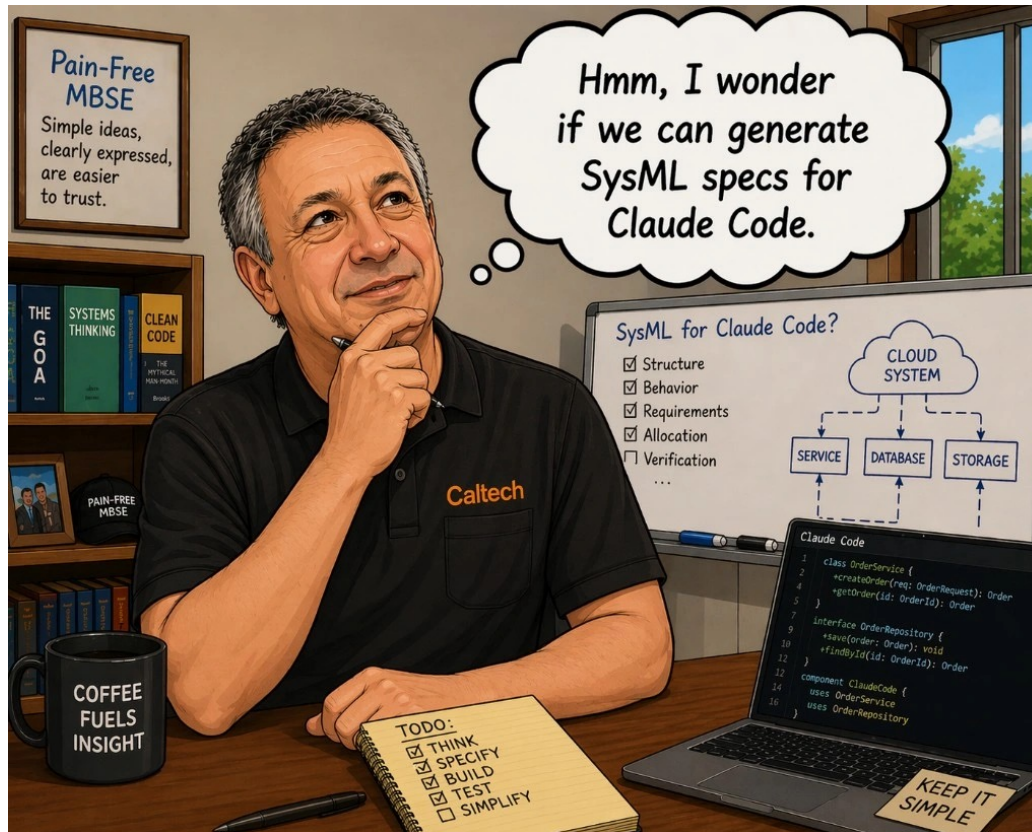
```
package 'Lunar Lander V2 Model' {
  package LunarLanderSubsystemsDecomposition {
    package Ascent_Stage {
      package Connections {
        private import ScalarValues::**;
        private import SI::**;
        interface def iPower;
        interface def iPropellant;
        interface def iPressurant;
        interface def iHeat;
        interface def iNavData;
        interface def iThrustCmd;
        interface def iAttCmd;
        interface def iCrewCmd;
        interface def iTelemetry;
        part def PowerIF {
          port bus : iPower;
        }
        part def FuelIF {
          port prop : iPropellant;
          port press : iPressurant;
        }
        part def ThermalIF {
          port heat : iHeat;
        }
        part def CrewIF {
          port crewCmd : iCrewCmd;
        }
        part def CommsIF {
          port pwr : iPower;
        }
      }
    }
  }
}
```



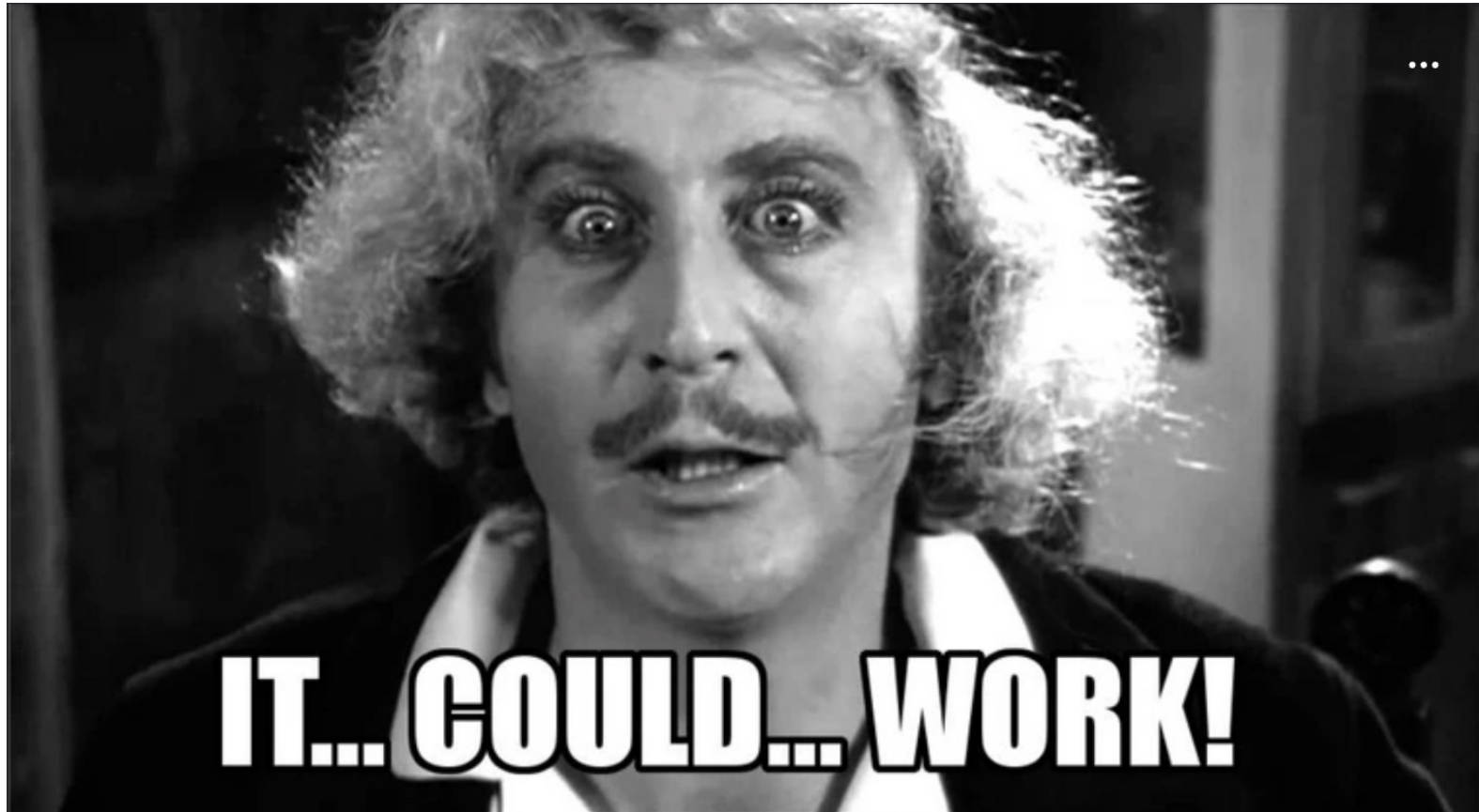
Summing up

- Code generation templates provide a syntax reference that allows AI to **reliably** generate Cameo compliant SysML v2 code
 - This fully enables “AI as a Subject Matter Expert” that we described in AI Assisted MBSE
- We started with one template per SysML 1 diagram type
- As our reliability rates improved, we started combining templates
 - System Spec Template
 - Logical Architecture template (one subsystem at a time)
 - Build your own templates as needed
- We can now generate specs (parts, states, interfaces, requirements) a subsystem at a time
 - As always with AI, humans must review the results
- This represents an enormous productivity boost for systems engineers

But wait...there's more...because a few weeks ago...



Can Claude build an app from a SysML v2 spec???



We'll never know unless we try...

1. System Overview

****OVR_001**** – StockPicker is a browser-based decision-support system for analyzing stock price data and presenting advisory buy, hold, or exit signals.

The system is intended to help a user inspect:

- Price behavior
- Trend behavior
- Trend-change behavior
- Simulated portfolio performance using simple, explainable rules

Build 1 Scope

Build 1 focuses on:

- Long-only strategies
- Simple Moving Average (SMA) smoothing
- Visualization of price and analysis curves
- Simulated portfolio support
- Explicit UI behavior
- Explicit testing structure

Pull stock data from online sources



Trend analysis



Simulated portfolio to track investment strategy

Ticker	Date	Shares	Buy Price	Current Price	Value	Gain/Loss \$	Gain/Loss %	Signal	Actions
GOOGL	2026-04-22	52	\$332.29	\$332.29	\$17279.08	\$0.00	0.00%	Hold	<button>Sell</button>
AMZN	2026-04-22	100	\$249.91	\$249.91	\$24991.00	\$0.00	0.00%	Buy	<button>Sell</button>
NVDA	2026-04-22	100	\$199.88	\$199.88	\$19988.00	\$0.00	0.00%	Buy	<button>Sell</button>
NVDA	2026-04-22	100	\$199.88	\$199.88	\$19988.00	\$0.00	0.00%	Buy	<button>Sell</button>
Total					\$82246.08	\$0.00	0.00%		

Sell NVDA

You own **100** shares at **\$199.88**

Shares to sell:

Cancel Sell

Hang on a tic...what's in that spec?

- Requirements
- Use Cases
- Domain Model (Glossary)
- State Machines
- UI Specs
- Data Models
- Testing Strategy

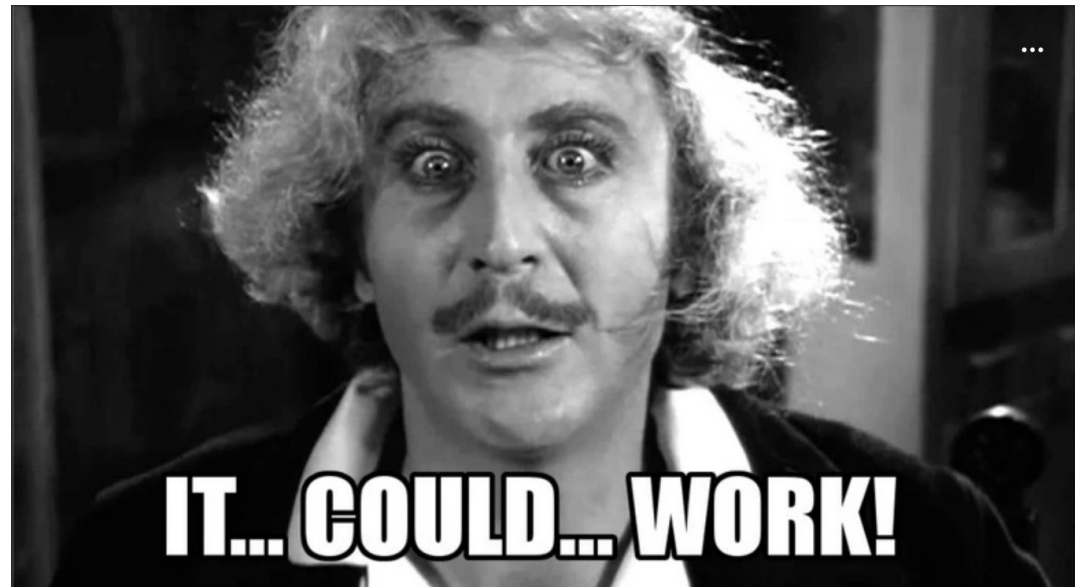
Table of Contents

1. [System Overview](#1-system-overview)
2. [Glossary](#2-glossary)
3. [Functional Requirements](#3-functional-requirements)
 - 3.1 Core Data and Analysis
 - 3.2 User Interface
 - 3.3 Portfolio Simulation
 - 3.4 Algorithm Transparency
 - 3.5 Portfolio Display
 - 3.6 Deferred to Build 2
4. [Build Constraints](#4-build-constraints)
5. [Use Cases](#5-use-cases)
6. [System State Behavior](#6-system-state-behavior)
 - 6.1 Daily Cycle State Machine
 - 6.2 UI State Machine
 - 6.3 Data Lifecycle State Machine
7. [Algorithm Specification](#7-algorithm-specification)
8. [Data Model](#8-data-model)
9. [UI/UX Specification](#9-uiux-specification)
10. [Testing Specification](#10-testing-specification)
 - 10.1 Assertion Tests
 - 10.2 Thread Tests
 - 10.3 Coverage Matrix

One prompt to generate the spec and three prompts to build the app?

3 prompts to Claude:

- 1) Build your markdown spec from this SysML v2 spec
- 2) Make a plan
- 3) Build the app



Learn more...

Dr. Rick Hefner, Caltech CTME:
rhefner@caltech.edu

Doug Rosenberg:
doug@parallelagile.com

Caltech | Center for Technology & Management Education

Individuals Organizations

AI-Assisted Model-Based Systems Engineering

AI-Assisted MBSE Training: Integrating AI into Enterprise Systems Modeling

AI-Assisted Model-Based Systems Engineering (AIM) is a compelling enterprise training program at the intersection of AI and digital

Learners	Intermediate
Time	Client definable
Duration	3–5 Days; Definable
Program Type	Customizable Programs
Certificate Type	Certificate
Format	ANY FORMAT/LOCATION
CEUS	Available
PDUS	Available
Program Number	AIM-Custom
Fees	Group Rate

<https://ctme.caltech.edu/ai-assisted-model-based-systems-engineering-aim-certificate-custom.html>