A RIGOROUS YET SIMPLE APPROACH TO SysML MODELING

PAIN-FREE MBSE

BY DOUG ROSENBERG & BRIAN MOBERLEY

MBSE4U

Pain-Free MBSE

A Rigorous Yet Simple Approach to SysML Modeling

Doug Rosenberg and Brian Moberley

This book is available at https://leanpub.com/pain-free-mbse

This version was published on 2026-03-22 ISBN 978-3-911081-13-9

/MBSE4U

MBSE4U is a lean publishing house dedicated to model-based systems engineering (MBSE) books, delivering up-to-date content that reflects the dynamic changes in the MBSE community and markets.

Our mission is to provide knowledge, practical guidance, and inspiration for MBSE practitioners. We publish works on MBSE methodologies, languages such as SysML, and methods including SYSMOD, VAMOS, FAS, AIM, and MBSE craftsmanship.

Contents

About MBSE4U

MBSE4U is a lean and agile publishing platform dedicated to advancing Model-Based Systems Engineering (MBSE). We deliver concise, practitioner-focused content that evolves with the rapidly changing MBSE landscape—keeping professionals ahead of emerging trends, tools, and methodologies.

Our mission is to empower systems engineers and organizations by providing high-impact publications that span the full spectrum of MBSE—from foundational principles to advanced practices. Each release is designed to be practical, relevant, and immediately applicable to real-world engineering challenges.

At MBSE4U, we believe in the power of collaboration. Through partnerships with experts, contributors, and our growing community—including the MBSE Podcast—we amplify modern systems engineering knowledge and bring it directly to your fingertips.

Whether you're seeking fresh perspectives, proven methodologies, or hands-on guidance, MBSE4U equips you with the knowledge to lead with confidence in an increasingly complex systems world.

MBSE4U has published the following books so far:

Weilkiens, T., Molnár, V. (2025) *The SysML v2 Book - Practical Insights and Comprehensive Reference*

Rosenberg, D., Weilkiens, T., and Moberley, B. (2024) *AI Assisted MBSE with SysML.*

Helle, P. (2024) *MBSE with SysML and Eclipse Papyrus*.

Weilkiens, T., Vinarcik, M. and Fischer, C. (2023) *The Craft of MBSE*.

Selic, B., Gullekson, G. and Ward, P.T. (2023) *Real-Time Object-Oriented Modeling*.

Weilkiens, T. (2019) The New Engineering Game.

Weinert, B. (2018) Ein Framework zur Architekturbeschreibung von soziotechnischen maritimen Systemen.

Neureiter, C. (2017) A Domain-Specific, Model Driven Engineering Approach for Systems Engineering in the Smart Grid.

Weilkiens, T. (2016) Variant Modeling with SysML. MBSE4U.

The current catalog is available on the MBSE4U website at www.mbse4u.com.

MBSE4U is also the home of *The MBSE Podcast* (mbse-podcast.rocks).

About Us

Doug Rosenberg is the founder and CEO of Parallel Agile, Inc., and has been a thought leader in software and systems engineering and object-oriented design for more than three decades. Pain-Free MBSE (PFM) is his 10th book.

He founded and managed ICONIX Software Engineering from 1984 until 2014. His work in the 1990s included integrating the modeling approaches of Booch, Rumbaugh, and Jacobson several years before the creation of UML. This integrated approach is known as the ICONIX Process.

For the last decade, he has been on a mission to integrate systems engineering and software engineering since virtually every system being built involves software, yet many systems engineering approaches fail to consider software. PFM is a major step towards that goal.

Brian Moberley is a Senior Systems Engineer for Innovative Defense Technologies, LLC. His Systems Engineering experience spans DoD as well as commercial projects. Brian is the creator and host of the MBSE iNsights YouTube channel, where he creates instructional videos to teach the Systems Engineering community the ins and outs of the practice. As a veteran of the U.S. Air Force and commercial pilot, Brian's diverse background and experience bring a unique perspective to the Systems Engineering community.

Foreword by Rick Hefner

Throughout my career, it has been my distinct privilege to work with and train the world's leading systems engineers. I've often witnessed a persistent struggle: the gap between the immense promise of Model-Based Systems Engineering (MBSE) and the often-frustrating, over-engineered reality of SysML execution. I've witnessed the pain firsthand, as dedicated engineers grapple with methodologies that create models that are overly complex, unmaintainable, and ultimately worthless. Doug Rosenberg and Brian Moberley didn't accept this problem; they found a simpler way forward by cataloging the practices that inflict maximum pain with minimum value.

The two are uniquely qualified to write this book because each has spent their career bridging the gaps that traditional engineering practices overlook, such as the integration of software into the systems design process. As long-time thought leaders in software and systems engineering, they have consistently advocated for a streamlined, minimalist modeling approach focused on utility over dogma. In this book, the authors provide a methodology that is battle-tested and focused on robust, efficient, and resilient systems.

One of the greatest strengths of *Pain-Free MBSE* is its use of a single, continuously evolving example—the Lunar Lander—to ground the entire

methodology from start to finish. Introduced at the outset and expanded throughout the chapters, this example illustrates how well-formed requirements, disciplined functional decomposition, and precisely defined interfaces converge to produce robust behavioral and parametric models. The journey concludes with a focused yet compelling simulation that unifies structure, behavior, and physics, demonstrating the power and elegance of MBSE when the complexity is managed and the clarity remains.

The importance of *Pain-Free MBSE* lies in its practical roadmap to escape the pitfalls of complexity. The book's fundamental insight is the Value-to-Pain Ratio (VPR): a pragmatic metric that helps you ruthlessly shed the low-VPR practices—the "10% of the practices [that] cause 90% of the pain"—and focus exclusively on those that genuinely improve clarity, decision-making, and system quality. If you are ready to eliminate the unnecessary friction from your process and build better systems faster, turn the page and begin your journey to modeling with confidence.

Rick Hefner, PhD

Executive Director, Center for Technology & Management Education California Institute of Technology

Preface

Over the past two decades, I've had the opportunity to teach SysML and model-based systems engineering to dozens of companies across industries. No matter where I went, no matter what course materials I used, I kept noticing the same thing: people always got stuck in the same places. Most often, the stumbling blocks were overloaded activity diagrams, or internal block diagrams where teams were endlessly chasing item flow violations. And when it came to parametrics, very few people had much of a clue at all. Another common stumbling block was trying to functionally decompose use cases, which usually resulted in uneven levels of modeling detail and huge amounts of time spent.

When I began working with Brian, he came at the problem from a different angle. He wasn't teaching students; he was fixing real projects. In the company where we briefly worked together, Brian had become the go-to person when projects got stuck. As we compared notes, we realized we were both seeing the same issues: the very same sticking points I encountered in training rooms were the ones Brian encountered in live projects. That was when we began talking about "high pain modeling approaches." The irony was that the more painful the approach, the worse the results turned out to be. The idea behind this book is that modeling doesn't have to be painful. The best results come when the process is clear, natural, and free of unnecessary friction.

For the example system throughout this book, we chose the Apollo Lunar Module. That choice is partly professional — the LM is one of the most iconic engineering achievements in history — and partly personal. When Apollo 11 landed on the moon in July 1969, I was twelve years old, completely obsessed with the space race. I have a cousin — technically, my father's cousin — named Mel Levine, who worked at Grumman in Bethpage on the Lunar Module. He is twenty-some years older than me, and I didn't see him often, but when I did, I thought it was the most fascinating thing in the world. In 2003, he was recognized with a Pioneer Award for his

groundbreaking work in telemetry, a reminder of just how far-reaching his contributions are. To have someone in my own family working on Apollo gives me a sense of connection to the adventure that defined my youth. More than fifty years later, as Artemis and companies like Blue Origin and SpaceX prepare to return to the moon, that inspiration comes full circle — especially since I've had the privilege of teaching SysML to engineers working on the next generation of lunar landers.

This book is also illustrated with images from "Alice in Wonderland", a theme I first explored in a keynote talk called "Alice in Use Case Land" at UML World around 2000. Alice has always seemed like a fitting guide for wandering through the curious landscapes of modeling languages, where the rules can feel as puzzling as the Queen of Hearts' courtroom. Thanks to modern AI art tools, I can now create many of these illustrations myself — a process that has actually been a whole lot of fun. It allows me to pair technical explanation with imagery that is whimsical and expressive, reminding us that systems engineering, while serious in purpose, doesn't have to be solemn in presentation.

Finally, the circle closes with Brian's contribution: a parametric simulation of the Lunar Lander that forms the last three chapters of this book. It serves both as a teaching example for parametrics and as a functional lunar landing game that runs inside the modeling tool itself. From Mel's work at Grumman, to the Apollo missions of my youth, to today's renewed efforts to return to the moon, the Lunar Lander has been a constant thread. It is both a symbol of complexity well-mastered and a reminder that the journey to mastery doesn't have to be painful.

Artificial Intelligence

The images featuring Alice were generated using GPT. They are based on images from the public domain, such as those by Sir John Tenniel. The other images in this book were also created using AI tools and carefully reviewed by the author. AI was used as a creative instrument to generate the photographic visuals presented.

Unless otherwise stated, the texts were written by the authors.

Download Lunar Lander Module Model

The example model of the Lunar Lander Module can be downloaded here: https://www.parallelagile.com/pfm-training.html.

Acknowledgment

The authors would like to thank the following people for their contributions to this book.

We thank Joy Au of OptimiSE for her contribution to the section on the EARS method for writing requirements. We thank Sophie Regnier of Booz Allen Hamilton for reviewing all of the Alice artwork for potential copyright issues. We also thank Dassault Systèmes for providing Cameo licenses used in writing this book, and in particular, Osvaldas Jankauskas for his help in getting started with SysML v2.

Appendix B - Lunar Lander V2 Model

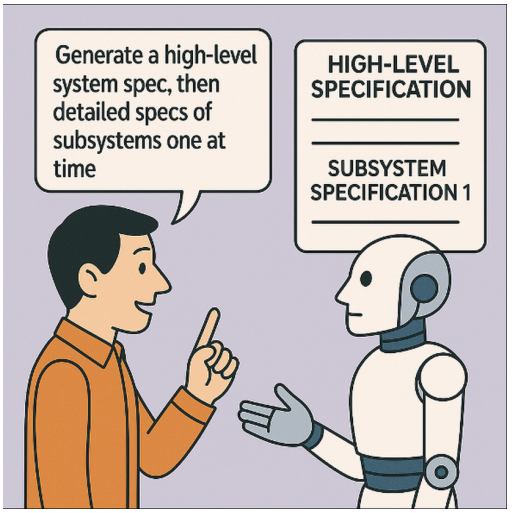
This appendix provides the Lunar Lander V2 model, a SysML v2 representation of the lunar lander example developed throughout the book. The model demonstrates how the techniques introduced in the main chapters can be expressed in SysML v2 using the textual syntax consistent with the "AI-Assisted MBSE" process (Rosenberg 2024). Let's start with the Marx Brothers in [1].

Is it just us, or does SysML v2 syntax remind you of the Marx Brothers?

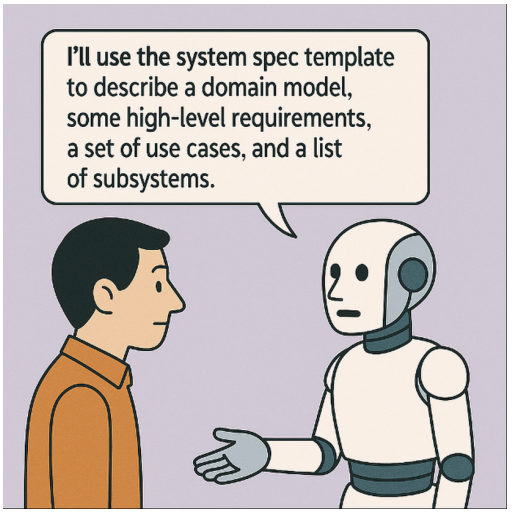
An outstanding benefit of SysML v2 is the ability for AI to code-generate v2 textual models.

Specifically, the v2 lander model was code-generated by AI using two code-generation templates: a System Spec template and a Logical Architecture Template.

[1] If you've never seen the famous "sanity clause" contract scene from A Night at the Opera, you can watch it here: https://youtu.be/dbUDSxJFsDA?si=_AAfLQy20LzDFpf9

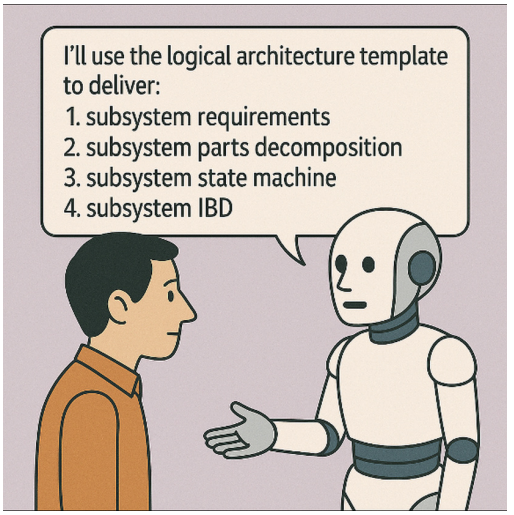


The system spec template provides a syntax reference for AI to generate v2 problem domain artifacts.



The appendix begins with the system-level view, including the domain model, glossary, top-level requirements, and mission state machine that together define the overall context and behavior of the lunar lander.

The Logical Architecture template generates domain-driven subsystem architecture textual notation.



It then decomposes the system into its primary subsystems—Ascent and Descent—each modeled with a consistent structure: part decomposition, requirements, state machine, and internal block diagram.

Taken together, these sections form a complete and traceable SysML v2 example that parallels the SysML v1 models presented earlier in the book, illustrating how the same engineering information can be represented more concisely and precisely in SysML v2. The SysML v2 textual notation is generated by AI, and the views in this Appendix are rendered in Cameo Systems Modeler. Templates were developed for AIM Training students.

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

1. Domain Model

Figure 7. Lunar Lander Top-Level Requirements (Textual Notation)

```
package 'Lunar Lander V2 Model' {
    package LunarLanderDomain_Grouped {
        ////////////////////////////////
        // Top-level conceptual groupings
        ////////////////////////////////
        part def LunarLander {
            part ascent         : Ascent;
            part descent        : Descent;
            part communications : Communications;
            part docking        : Docking;
            part controls       : Controls;
            part fuelSystem     : FuelSystem;
            part crewCabin      : CrewCabin;
        }

        // ---- Group: Ascent ----
        part def Ascent {
            part ascentEngine             : Engine;
            part reactionControlThrusters : ThrusterSet;
        }

        // ---- Group: Descent ----
        part def Descent {
            part descentEngine  : Engine;
            part landingGear    : LandingGear;
            part footpads       : FootpadSet;
            part contactProbes  : ContactProbeSet;
            part radarAltimeter : RadarAltimeter;
        }

        // ---- Group: Communications ----
        part def Communications {
            part antenna       : Antenna;
            part rfTransceiver : RFTransceiver;
        }

        // ---- Group: Docking ----
        part def Docking {
            part mechanism : DockingMechanism;
            part latches   : LatchSet;
            part sensors   : DockingSensorSet;
        }

        // ---- Group: Controls (guidance, sensing, crew interface) ----
```

```
        part def Controls {
            // guidance & computing
            part guidanceComputer        : GuidanceComputer;
            // sensing for guidance/attitude
            part inertialMeasurementUnit : IMU;
            part starTracker             : StarTracker;
            // crew interface
            part displaysAndControls     : DisplaysAndControls;
        }

        // ---- Group: Fuel System (propellants & lines) ----
        part def FuelSystem {
            part fuelTank        : Tank;
            part oxidizerTank    : Tank;
            part pressurantTank  : Tank;
            part valveSet        : ValveSet;
            part propellantLines : LineSet;
        }

        // ---- Group: CrewCabin (habitation + power/thermal) ----
        part def CrewCabin {
            // habitation shell & access
            part pressureVessel : PressureVessel;
            part hatch          : CrewHatch;
            part windows        : WindowSet;
            part crew           : Crew;

            // power & thermal local to cabin
            part battery           : Battery;
            part powerDistribution : PowerDistribution;
            part heater            : Heater;
            part radiator          : Radiator;
            part insulation        : Insulation;
        }

        ////////////////////////////////
        // External context (unchanged)
        ////////////////////////////////
        part def ExternalContext {
            part moon                : Moon;
            part lunarSurface        : LunarSurface;
            part lunarOrbit          : LunarOrbit;
            part landingSite         : LandingSite;
            part commandServiceModule : CommandServiceModule;
            part missionControl      : MissionControl;
            part missionTimeline     : MissionTimeline;
        }

        ////////////////////////////////
        // Domain noun type catalog  //
```

```
95          ////////////////////////////////
96
97          // people/vehicle shell
98          part def Crew { }
99          part def PressureVessel { }
100         part def CrewHatch { }
101         part def WindowSet { }
102
103         // avionics & sensing & UI
104         part def GuidanceComputer { }
105         part def IMU { }
106         part def StarTracker { }
107         part def DisplaysAndControls { }
108
109         // communications
110         part def Antenna { }
111         part def RFTransceiver { }
112
113         // power & thermal
114         part def Battery { }
115         part def PowerDistribution { }
116         part def Heater { }
117         part def Radiator { }
118         part def Insulation { }
119
120         // propulsion / docking / landing / lines
121         part def Engine { }
122         part def ThrusterSet { }
123         part def Tank { }
124         part def ValveSet { }
125         part def LineSet { }
126         part def LandingGear { }
127         part def FootpadSet { }
128         part def ContactProbeSet { }
129         part def DockingMechanism { }
130         part def LatchSet { }
131         part def DockingSensorSet { }
132         part def RadarAltimeter { }
133
134         // external/context nouns
135         part def Moon { }
136         part def LunarSurface { }
137         part def LunarOrbit { }
138         part def LandingSite { }
139         part def CommandServiceModule { }
140         part def MissionControl { }
141         part def MissionTimeline { }
142     }
143 }
```

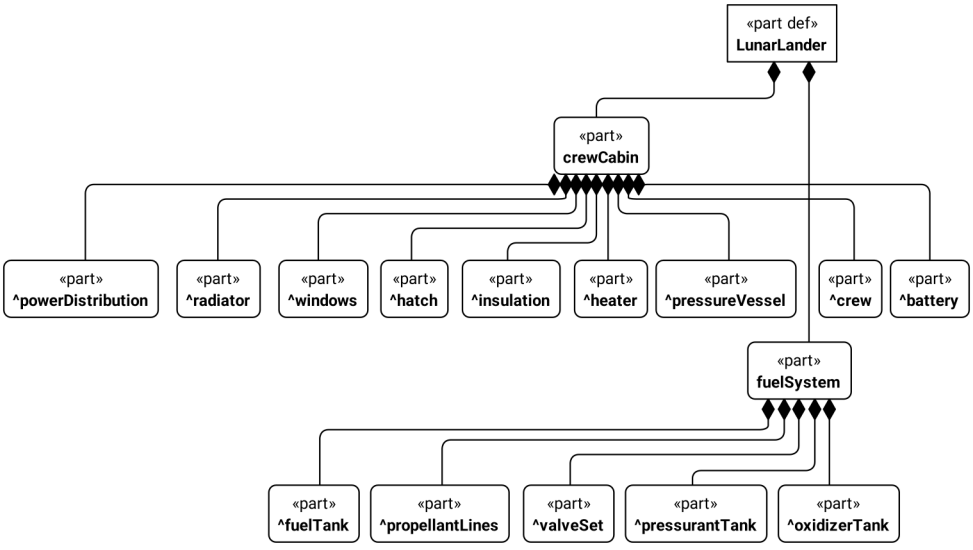


Figure 8. Lunar Lander Domain Model (General View)

2. Glossary

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.

- **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.
- **Footpads:** Absorb landing loads and stabilize the lander on the lunar surface.

- **FuelTank:** Stores fuel used by propulsion systems.
- **GuidanceComputer:** Executes navigation, guidance, and control computations.
- **Hatch:** Provides ingress and egress between the lander and the lunar surface or CSM.
- **Heater:** Maintains thermal balance within allowable temperature limits.
- **InertialMeasurementUnit:** Measures linear acceleration and angular rate for attitude/position estimation.
- **Insulation:** Protects cabin and systems from extreme lunar temperature variation.
- **LandingGear:** Deployable structure that supports the vehicle after touchdown.
- **LandingSite:** Designated region on the Moon's surface selected for landing.
- **Latches:** Mechanically secure a hard-dock connection with the Command Module.
- **LunarOrbit:** Orbital path around the Moon where docking and undocking occur.
- **LunarSurface:** The terrain of the Moon where the lander touches down.
- **Mechanism:** Structural alignment and capture hardware used during docking operations.
- **MissionControl:** Earth-based operations center that monitors and communicates with the lander.
- **MissionTimeline:** Sequenced mission events and phases defining operational timing.
- **Moon:** Celestial body providing the mission environment.
- **OxidizerTank:** Stores oxidizer used by propulsion systems.
- **PowerDistribution:** Routes and manages electrical power to subsystems.
- **PressureVessel:** Sealed, habitable volume providing a pressurized cabin for the crew.
- **PressurantTank:** Contains gas used to maintain propellant tank pressure.
- **PropellantLines:** Transfer propellants and pressurant between tanks and engines.

- **RadarAltimeter:** Measures altitude and rate of descent relative to the lunar surface.
- **Radiator:** Rejects waste heat to space to regulate internal temperatures.
- **ReactionControlThrusters:** Provide attitude and small translation control during ascent and docking.
- **RfTransceiver:** Manages radio-frequency transmission and reception of voice, data, and telemetry.
- **Sensors:** Docking sensors that detect and verify successful engagement.
- **StarTracker:** Provides precise celestial attitude reference for guidance alignment.
- **BalveSet:** Controls flow paths within propellant and pressurization circuits.
- **Windows:** Provide external visibility for landing and docking operations.

3. Top-Level Requirements

Figure 9. Lunar Lander Top-Level Requirements (Textual Notation)

```
package 'Lunar Lander V2 Model' {
    package LunarLanderRequirements {
        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_013_FaultManagement {
            doc /* The Lunar Lander shall provide fault detection, isolation, and
                 * recovery with health and status reporting to crew and ground. */
        }
        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. */
        }
    }
}
```

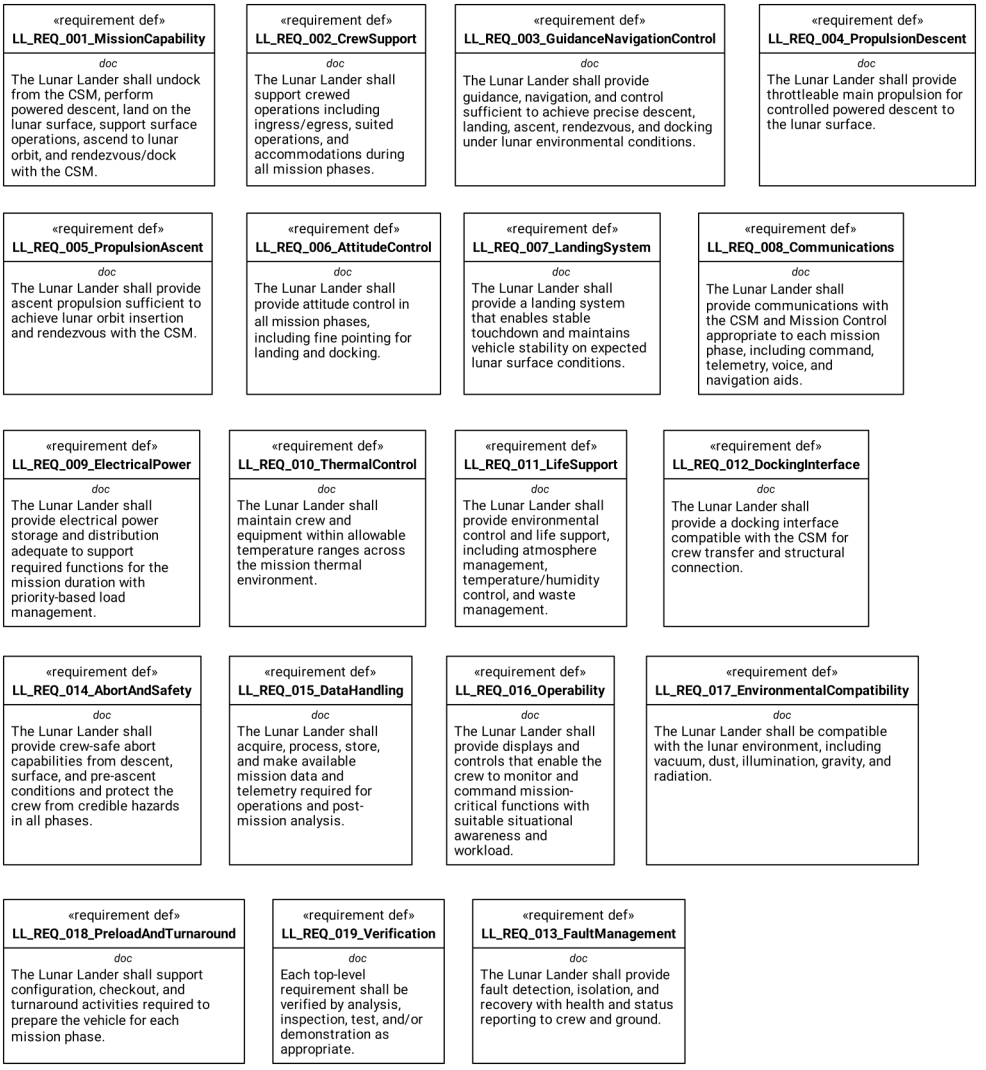


Figure 10. Lunar Lander Top-Level Requirements (General View)

4. Top-Level State Machine

Figure 11. Lunar Lander Top-Level State Machine (Textual Notation)

```
package 'Lunar Lander V2 Model' {
    package LunarLanderTopLevelStateMachine {
        item def GoUndock;
        item def SeparationConfirmed;
        item def GoDescent;
        item def Landed;
        item def GoAscent;
        item def OrbitAchieved;
        item def InProximity;
        item def DockComplete;
        part def LunarLander {
            exhibit state LanderMission;
        }
        state def LanderMission {
            state DockedToCSM;
            state Undocking;
            state SeparatedCheckout;
            state Descent;
            state SurfaceOps;
            state Ascent;
            state RendezvousAndDocking;
            state DockedToCSM_Final;
            transition first entryAction then DockedToCSM;
            transition t1 first DockedToCSM accept GoUndock then Undocking;
            transition t2 first Undocking accept SeparationConfirmed then
            ↪ SeparatedCheckout;
            transition t3 first SeparatedCheckout accept GoDescent then Descent;
            transition t4 first Descent accept Landed then SurfaceOps;
            transition t5 first SurfaceOps accept GoAscent then Ascent;
            transition t6 first Ascent accept OrbitAchieved then RendezvousAndDocking;
            transition t7 first RendezvousAndDocking accept DockComplete then
            ↪ DockedToCSM_Final;
        }
    }
}
```

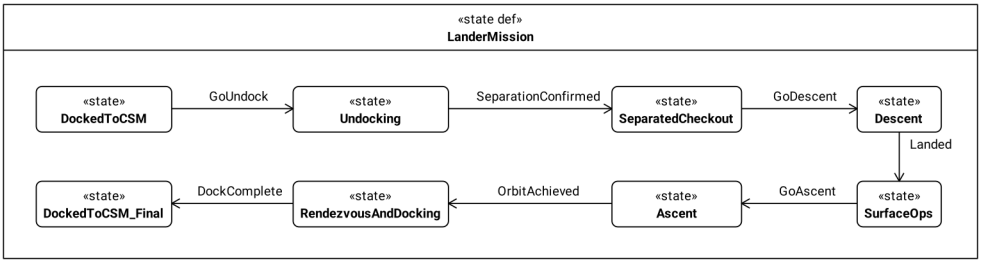


Figure 12. Lunar Lander Top-Level State Machine (General View)

5. Top-Level Subsystem Decomposition

Figure 13. Lunar Lander Top-Level Subsystem Decomposition (Textual Notation)

```
package 'Lunar Lander V2 Model' {
    package LunarLanderSubsystemsDecomposition {
        action def EstimateState;
        action def ComputeGuidanceTargets;
        action def GenerateAttitudeCommands;
        action def GenerateThrottleCommands;
        action def ManageModeTransitions;
        action def ExecuteFlightSoftware;
        action def AcquireSensorData;
        action def SynchronizeTime;
        action def PerformFaultDetectionIsolationRecovery;
        action def LogTelemetry;
        action def RouteCommands;
        action def IgniteDescentEngine;
        action def ShutdownDescentEngine;
        action def ThrottleDescentEngine;
        action def ControlThrustVector;
        action def ArmAscentEngine;
        action def IgniteAscentEngine;
        action def ShutdownAscentEngine;
        action def ExecuteAscentBurn;
        action def PerformRendezvousTrimBurn;
        action def AttitudeHold;
        action def SlewAttitude;
        action def TranslateForProximityOps;
        action def NullRates;
        action def MaintainTankPressure;
        action def ConditionPropellants;
        action def OpenCloseValves;
        action def ReportPropellantQuantity;
        action def StoreEnergy;
        action def DistributePower;
        action def RegulateBusVoltages;
        action def ShedNonCriticalLoads;
        action def MonitorPowerHealth;
        action def HeatCriticalComponents;
        action def RejectWasteHeat;
        action def InsulateStructures;
        action def MaintainThermalSetpoints;
        action def SupplyOxygen;
        action def RemoveCarbonDioxide;
        action def ControlCabinPressure;
        action def ControlCabinHumidity;
        action def ManageWaterAndWaste;
```

```
        action def MonitorLifeSupportHealth;
        action def EnableIngressEgress;
        action def ProvideViewing;
        action def RestrainCrewDuringDynamics;
        action def TransmitTelemetry;
        action def ReceiveCommands;
        action def ManageCommsModes;
        action def SupportRendezvousNavAids;
        action def MeasureIMU;
        action def TrackStars;
        action def MeasureAltitudeAndRangeRate;
        action def IndicateContactLight;
        action def AlignForDocking;
        action def CaptureSoftDock;
        action def LatchHardDock;
        action def VerifyDockLatchStatus;
        action def ManageDockingUmbilicals;
        action def DeployLandingGear;
        action def LockLandingGear;
        action def AbsorbTouchdownLoads;
        action def StabilizeOnSurface;
        action def SenseTouchdown;
        action def MaintainMissionClock;
        action def SequenceCriticalEvents;
        action def ArmTimeCriticalFunctions;
        part def AscentStage {
            part ascentEngineCtrl : AscentEngineControl;
            part ascentRendezvousCtrl : RendezvousControl;
            part ascentRCS : RCSModule;
            part ascentGNCComputer : GNCComputer;
            part ascentNavSensors : NavSensorSuite;
        }
        part def DescentStage {
            part descentEngineCtrl : DescentEngineControl;
            part landingSensorSuite : LandingSensorSuite;
            part landingGearAssembly : LandingGearAssembly;
            part descentGNCComputer : GNCComputer;
        }
        part def CrewCabin {
            part eclssAir : ECLSS_AirManagement;
            part eclssWaterWaste : ECLSS_WaterWaste;
            part crewInterface : CrewInterface;
            part cabinThermal : ThermalLoop;
            part cabinPowerPanel : PowerDistributionPanel;
            part crewProtection : CrewProtection;
        }
        part def CommunicationsAndTelemetry {
            part rfTransceiverUnit : RFTransceiverUnit;
            part commsAntennaSystem : AntennaSystem;
            part telemetryRecorder : TelemetryRecorder;
```

```
        }
        part def Docking {
            part dockingMechanism : DockingMechanismUnit;
            part dockingLatchCtrl : DockingLatchController;
            part dockingUmbilicals : DockingUmbilicalManager;
        }
        part def PropellantAndPressurization {
            part tankPressurization : TankPressurization;
            part propellantConditioner : PropellantConditioner;
            part valveManifold : ValveManifold;
            part propellantQuantity : PropellantQuantityEst;
        }
        part def ElectricalPower {
            part batteryStack : BatteryStack;
            part powerConverter : PowerConverter;
            part powerBusController : PowerBusController;
        }
        part def ThermalControl {
            part heaterController : HeaterController;
            part radiatorAssembly : RadiatorAssembly;
            part insulationLayer : InsulationLayer;
            part thermalSetpointMgr : ThermalSetpointManager;
        }
        part def AvionicsAndDataHandling {
            part flightComputer : FlightComputer;
            part timeSync : TimeSynchronizer;
            part fdirController : FDIRController;
            part cmdRouter : CommandRouter;
        }
        part def MissionOperations {
            part missionClock : MissionClock;
            part eventSequencer : EventSequencer;
            part armInterlocks : ArmInterlocks;
        }
        part def Sensors {
            part imuUnit : IMUUnit;
            part starTrackerUnit : StarTrackerUnit;
            part landingRadar : LandingRadar;
            part contactProbes : ContactProbeArray;
        }
        part def AscentEngineControl {
            action armAscentEngine : ArmAscentEngine;
            action igniteAscentEngine : IgniteAscentEngine;
            action shutdownAscentEngine : ShutdownAscentEngine;
            action executeAscentBurn : ExecuteAscentBurn;
        }
        part def RendezvousControl {
            action performRendezvousTrimBurn : PerformRendezvousTrimBurn;
            action translateForProximityOps : TranslateForProximityOps;
        }
```

```
        part def RCSModule {
            action attitudeHold : AttitudeHold;
            action slewAttitude : SlewAttitude;
            action nullRates : NullRates;
        }
        part def GNCComputer {
            action estimateState : EstimateState;
            action computeGuidanceTargets : ComputeGuidanceTargets;
            action generateAttitudeCommands : GenerateAttitudeCommands;
            action generateThrottleCommands : GenerateThrottleCommands;
            action manageModeTransitions : ManageModeTransitions;
        }
        part def NavSensorSuite {
            action measureIMU : MeasureIMU;
            action trackStars : TrackStars;
        }
        part def DescentEngineControl {
            action igniteDescentEngine : IgniteDescentEngine;
            action shutdownDescentEngine : ShutdownDescentEngine;
            action throttleDescentEngine : ThrottleDescentEngine;
            action controlThrustVector : ControlThrustVector;
        }
        part def LandingSensorSuite {
            action measureAltitudeAndRangeRate : MeasureAltitudeAndRangeRate;
            action indicateContactLight : IndicateContactLight;
        }
        part def LandingGearAssembly {
            action deployLandingGear : DeployLandingGear;
            action lockLandingGear : LockLandingGear;
            action absorbTouchdownLoads : AbsorbTouchdownLoads;
            action stabilizeOnSurface : StabilizeOnSurface;
            action senseTouchdown : SenseTouchdown;
        }
        part def ECLSS_AirManagement {
            action supplyOxygen : SupplyOxygen;
            action removeCarbonDioxide : RemoveCarbonDioxide;
            action controlCabinPressure : ControlCabinPressure;
            action controlCabinHumidity : ControlCabinHumidity;
            action monitorLifeSupportHealth : MonitorLifeSupportHealth;
        }
        part def ECLSS_WaterWaste {
            action manageWaterAndWaste : ManageWaterAndWaste;
        }
        part def CrewInterface {
            action enableIngressEgress : EnableIngressEgress;
            action provideViewing : ProvideViewing;
            action restrainCrewDuringDynamics : RestrainCrewDuringDynamics;
        }
        part def ThermalLoop {
            action heatCriticalComponents : HeatCriticalComponents;
```

```
            action rejectWasteHeat : RejectWasteHeat;
            action maintainThermalSetpoints : MaintainThermalSetpoints;
        }
        part def PowerDistributionPanel {
            action distributePower : DistributePower;
        }
        part def CrewProtection {
            action insulateStructures : InsulationLayer;
        }
        part def RFTransceiverUnit {
            action transmitTelemetry : TransmitTelemetry;
            action receiveCommands : ReceiveCommands;
            action manageCommsModes : ManageCommsModes;
        }
        part def AntennaSystem {
            action supportRendezvousNavAids : SupportRendezvousNavAids;
        }
        part def TelemetryRecorder {
            action logTelemetry : LogTelemetry;
        }
        part def DockingMechanismUnit {
            action alignForDocking : AlignForDocking;
            action captureSoftDock : CaptureSoftDock;
        }
        part def DockingLatchController {
            action latchHardDock : LatchHardDock;
            action verifyDockLatchStatus : VerifyDockLatchStatus;
        }
        part def DockingUmbilicalManager {
            action manageDockingUmbilicals : ManageDockingUmbilicals;
        }
        part def TankPressurization {
            action maintainTankPressure : MaintainTankPressure;
        }
        part def PropellantConditioner {
            action conditionPropellants : ConditionPropellants;
        }
        part def ValveManifold {
            action openCloseValves : OpenCloseValves;
        }
        part def PropellantQuantityEst {
            action reportPropellantQuantity : ReportPropellantQuantity;
        }
        part def BatteryStack {
            action storeEnergy : StoreEnergy;
            action monitorPowerHealth : MonitorPowerHealth;
        }
        part def PowerConverter {
            action regulateBusVoltages : RegulateBusVoltages;
        }
```

```
245         part def PowerBusController {
246             action shedNonCriticalLoads : ShedNonCriticalLoads;
247             action distributePower : DistributePower;
248         }
249         part def HeaterController {
250             action heatCriticalComponents : HeatCriticalComponents;
251         }
252         part def RadiatorAssembly {
253             action rejectWasteHeat : RejectWasteHeat;
254         }
255         part def InsulationLayer {
256             action insulateStructures : InsulateStructures;
257         }
258         part def ThermalSetpointManager {
259             action maintainThermalSetpoints : MaintainThermalSetpoints;
260         }
261         part def FlightComputer {
262             action executeFlightSoftware : ExecuteFlightSoftware;
263             action performFDIR : PerformFaultDetectionIsolationRecovery;
264             action routeCommands : RouteCommands;
265         }
266         part def TimeSynchronizer {
267             action synchronizeTime : SynchronizeTime;
268         }
269         part def FDIRController {
270             action performFDIR : PerformFaultDetectionIsolationRecovery;
271         }
272         part def CommandRouter { action routeCommands : RouteCommands; }
273         part def MissionClock {
274             action maintainMissionClock : MaintainMissionClock;
275         }
276         part def EventSequencer {
277             action sequenceCriticalEvents : SequenceCriticalEvents;
278         }
279         part def ArmInterlocks {
280             action armTimeCriticalFunctions : ArmTimeCriticalFunctions;
281         }
282         part def IMUUnit { action measureIMU : MeasureIMU; }
283         part def StarTrackerUnit { action trackStars : TrackStars; }
284         part def LandingRadar {
285             action measureAltitudeAndRangeRate : MeasureAltitudeAndRangeRate;
286         }
287         part def ContactProbeArray {
288             action indicateContactLight : IndicateContactLight;
289         }
290     }
291 }
```

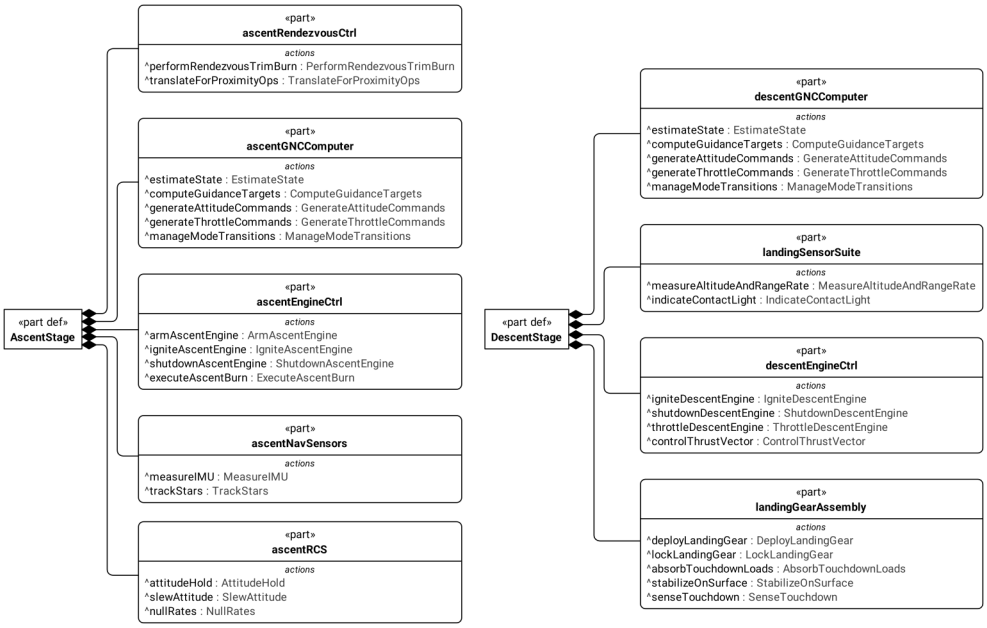


Figure 14. Lunar Lander Top-Level Subsystem Decomposition Part 1 (General View)

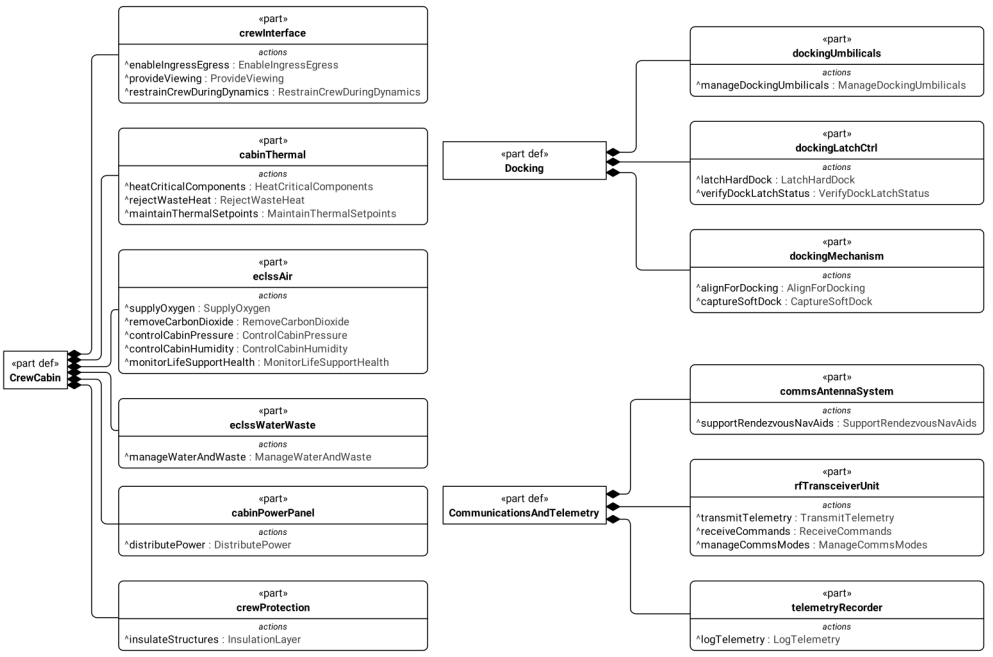


Figure 15. Lunar Lander Top-Level Subsystem Decomposition Part 2 (General View)

6. Ascent Stage Subsystem

6.1 Part Decomposition

Figure 16. Ascent Stage Subsystem Parts Decomposition (Textual Notation)

```
package 'Lunar Lander V2 Model' {
    package LunarLanderSubsystemsDecomposition {
        package Ascent_Stage {
            package PartDecomposition {
                private import ScalarValues::**;
                private import SI::**;
                action def ComputeGuidance;
                action def EstimateState;
                action def GenerateThrustCommand;
                action def GenerateAttitudeCommand;
                action def CommandIgnition;
                action def CommandShutdown;
                action def ModulateThrottle;
                action def FireRCS;
                action def MonitorHealth;
                action def ReportTelemetry;
                part def AscentEngineControl {
                    attribute engineState : String;
                    attribute throttlePct : Real;
                    attribute chamberPress_kPa : Real;
                    action ignite : CommandIgnition;
                    action shutdown : CommandShutdown;
                    action throttle : ModulateThrottle;
                    action report : ReportTelemetry;
                    action monitor : MonitorHealth;
                }
                part def RCSModule {
                    attribute mode : String;
                    attribute tankPressure_kPa : Real;
                    attribute propRemain_kg : Real;
                    action fire : FireRCS;
                    action report : ReportTelemetry;
                    action monitor : MonitorHealth;
                }
                part def NavSensorSuite {
                    attribute imuBias_dps : Real;
                    attribute starLock : Boolean;
                    action report : ReportTelemetry;
                    part imu : IMU;
                    part stars : StarTracker;
                }
                part def IMU {
```

```
43                      attribute gyroRate_dps : Real;
44                      attribute accel_mps2 : Real;
45                      action report : ReportTelemetry;
46                  }
47                  part def StarTracker {
48                      attribute starCount : Integer;
49                      attribute updateRate_Hz : Real;
50                      action report : ReportTelemetry;
51                  }
52                  part def GuidanceComputer {
53                      attribute mode : String;
54                      attribute targetOrbit : String;
55                      attribute guidanceHz : Integer;
56                      action estimate : EstimateState;
57                      action compute : ComputeGuidance;
58                      action genThrust : GenerateThrustCommand;
59                      action genAtt : GenerateAttitudeCommand;
60                      action report : ReportTelemetry;
61                      action monitor : MonitorHealth;
62                  }
63                  part def CommsInterface {
64                      attribute tlmRate_Hz : Integer;
65                      attribute linkStatus : String;
66                      action report : ReportTelemetry;
67                      action monitor : MonitorHealth;
68                  }
69                  part def PowerInterface {
70                      attribute busVoltage_V : Real;
71                      attribute busHealth : String;
72                      action monitor : MonitorHealth;
73                  }
74                  part def FuelInterface {
75                      attribute fuelTemp_K : Real;
76                      attribute oxidTemp_K : Real;
77                      attribute press_kPa : Real;
78                      action monitor : MonitorHealth;
79                  }
80                  part def ThermalInterface {
81                      attribute loopTemp_K : Real;
82                      attribute heatLoad_W : Real;
83                      action monitor : MonitorHealth;
84                  }
85                  part def CrewInterface {
86                      attribute lastCmd : String;
87                      attribute cmdSeqNum : Integer;
88                      action report : ReportTelemetry;
89                  }
90                  part def AscentStructure {
91                      attribute mass_kg : Real;
92                      attribute margin_pct : Real;
```

```
                }
                part def Ascent {
                    part engineCtrl : AscentEngineControl;
                    part rcs : RCSModule;
                    part navSensors : NavSensorSuite;
                    part guidance : GuidanceComputer;
                    part commsIf : CommsInterface;
                    part powerIf : PowerInterface;
                    part fuelIf : FuelInterface;
                    part thermalIf : ThermalInterface;
                    part crewIf : CrewInterface;
                    part structure : AscentStructure;
                }
            }
        }
    }
}
```

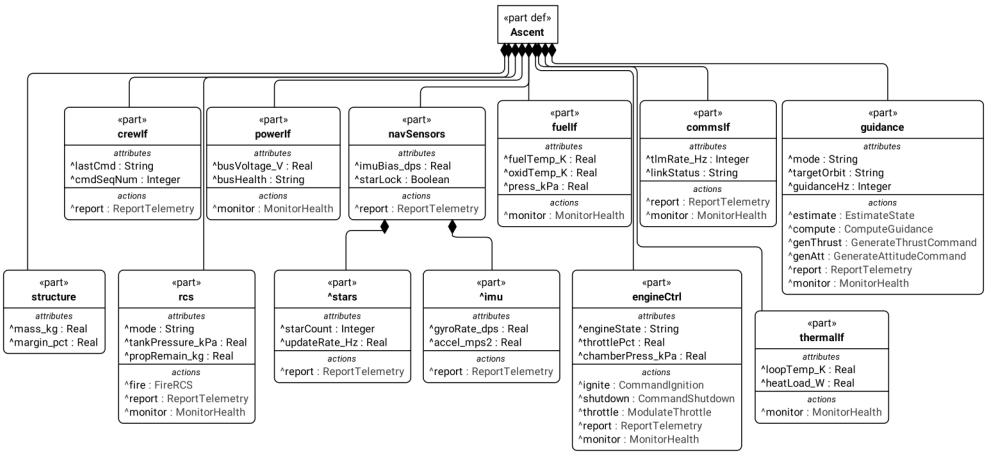


Figure 17. Ascent Stage Subsystem Parts Decomposition (General View)

6.2 Requirements

Figure 18. Ascent Stage Subsystem Requirements (Textual Notation)

```
package 'Lunar Lander V2 Model' {
    package LunarLanderSubsystemsDecomposition {
        package Ascent_Stage {
            package Requirements {
                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. */
                }
            }
```

```
45 }}}
```

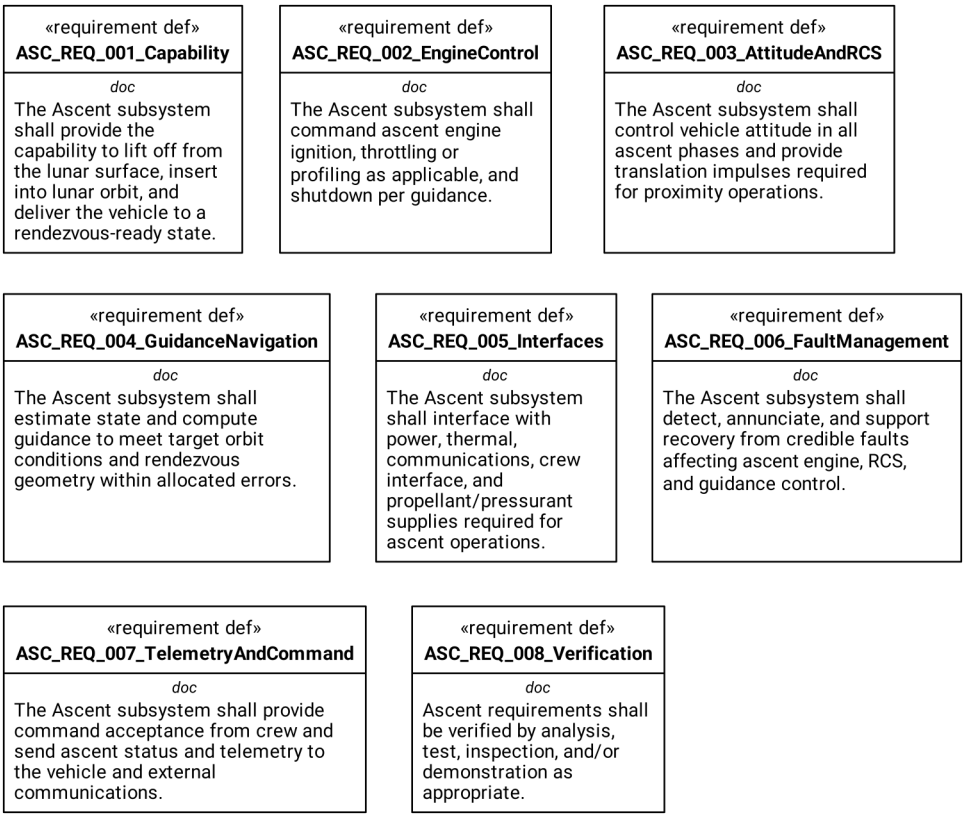


Figure 19. Ascent Stage Subsystem Requirements (General View)

6.3 State Machine

Figure 20. Ascent Stage Subsystem State Machine (Textual Notation)

```
package 'Lunar Lander V2 Model' {
    package LunarLanderSubsystemsDecomposition {
        package Ascent_Stage {
            package StateMachine {
                item def GoAscent;
                item def LiftOffConfirmed;
                item def MECO;
                item def OrbitAchieved;
                item def RendezvousReady;
                part def AscentSM_Owner {
                    exhibit state AscentSM;
                }
                state def AscentSM {
                    state PreAscentPrep;
                    state Ignition;
                    state Liftoff;
                    state OrbitInsertion;
                    state PhasingAndRendezvousPrep;
                    state Complete;
                    transition t0 first entryAction then PreAscentPrep;
                    transition t1 first PreAscentPrep accept GoAscent then Ignition;
                    transition t2 first Ignition accept LiftOffConfirmed then Liftoff;
                    transition t3 first Liftoff accept MECO then OrbitInsertion;
                    transition t4 first OrbitInsertion accept OrbitAchieved then
                    ↪ PhasingAndRendezvousPrep;
                    transition t5 first PhasingAndRendezvousPrep accept RendezvousReady
                    ↪ then Complete;
                }
            }
        }
    }
}
```

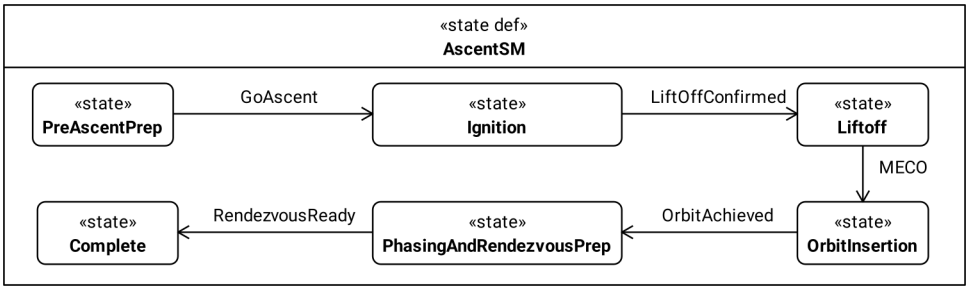


Figure 21. Ascent Stage Subsystem State Machine (General View)

6.4 Connections

Figure 22. Ascent Stage Subsystem Connections (Textual Notation)

```
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;
                    port tlmIn : iTelemetry;
                    port tlmOut : iTelemetry;
                }
                part def AscentEngineControl {
                    port pwr : iPower;
                    port prop : iPropellant;
                    port press : iPressurant;
                    port thrustCmd : iThrustCmd;
                    port engTlm : iTelemetry;
                }
                part def RCSModule {
                    port pwr : iPower;
                    port attCmd : iAttCmd;
                    port rcsTlm : iTelemetry;
                }
```

```
                part def NavSensorSuite {
                    port pwr : iPower;
                    port nav : iNavData;
                }
                part def GuidanceComputer {
                    port pwr : iPower;
                    port navIn : iNavData;
                    port crewIn : iCrewCmd;
                    port thrust : iThrustCmd;
                    port att : iAttCmd;
                    port tlm : iTelemetry;
                }
                part def Ascent {
                    part power : PowerIF;
                    part fuel : FuelIF;
                    part thermal : ThermalIF;
                    part crew : CrewIF;
                    part comms : CommsIF;
                    part engineCtrl : AscentEngineControl;
                    part rcs : RCSModule;
                    part nav : NavSensorSuite;
                    part guidance : GuidanceComputer;
                    interface connect power.bus to engineCtrl.pwr;
                    interface connect power.bus to rcs.pwr;
                    interface connect power.bus to nav.pwr;
                    interface connect power.bus to guidance.pwr;
                    interface connect power.bus to comms.pwr;
                    interface connect fuel.prop to engineCtrl.prop;
                    interface connect fuel.press to engineCtrl.press;
                    interface connect guidance.thrust to engineCtrl.thrustCmd;
                    interface connect guidance.att to rcs.attCmd;
                    interface connect nav.nav to guidance.navIn;
                    interface connect crew.crewCmd to guidance.crewIn;
                    interface connect engineCtrl.engTlm to comms.tlmIn;
                    interface connect rcs.rcsTlm to comms.tlmIn;
                    interface connect guidance.tlm to comms.tlmIn;
                }
            }
        }
    }
}
```

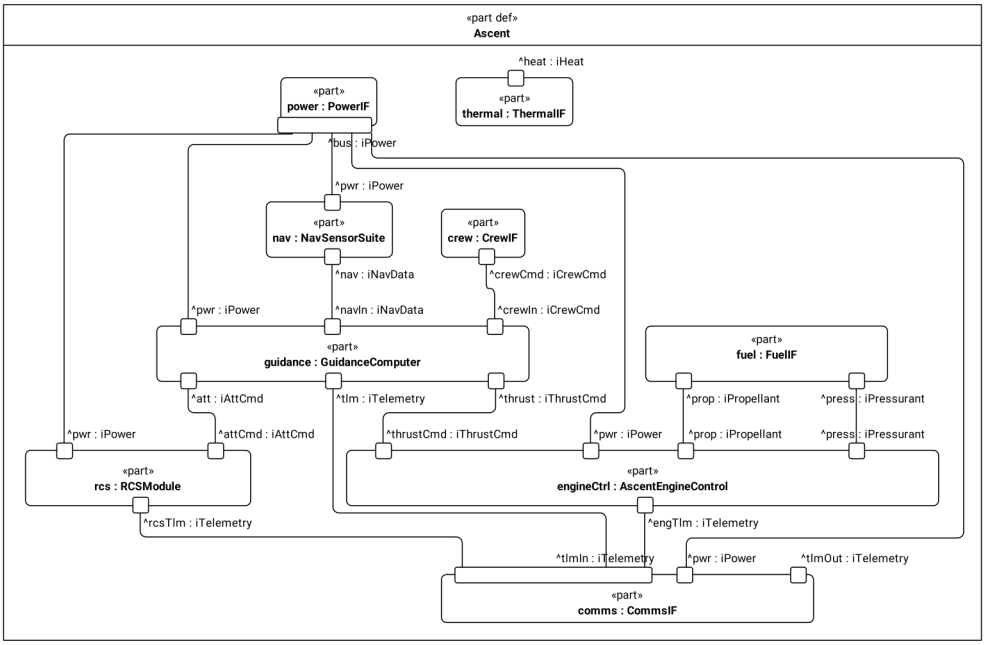


Figure 23. Ascent Stage Subsystem Connections (General View)

7. Descent Stage Subsystem

7.1 Part Decomposition

Figure 24. Descent Stage Subsystem Parts Decomposition (Textual Notation)

```
package 'Lunar Lander V2 Model' {
    package LunarLanderSubsystemsDecomposition {
        package Descent_Stage {
            package PartDecomposition {
                private import ScalarValues::**;
                private import SI::**;
                action def EstimateState;
                action def ComputeDescentGuidance;
                action def GenerateThrottleCommand;
                action def GenerateGimbalCommand;
                action def GenerateAttitudeCommand;
                action def CommandDescentIgnition;
                action def CommandDescentShutdown;
                action def ModulateDescentThrottle;
                action def ControlGimbal;
                action def FireRCS;
                action def DeployLandingGear;
                action def LockLandingGear;
                action def DampenLandingLoads;
                action def StabilizeOnSurface;
                action def SenseContact;
                action def ReportTelemetry;
                action def MonitorHealth;
                part def DescentEngineControl {
                    attribute engineState : String;
                    attribute throttlePct : Real;
                    attribute chamberPress_kPa : Real;
                    action ignite : CommandDescentIgnition;
                    action shutdown : CommandDescentShutdown;
                    action throttle : ModulateDescentThrottle;
                    action gimbal : ControlGimbal;
                    action report : ReportTelemetry;
                    action monitor : MonitorHealth;
                }
                part def RCSModule {
                    attribute mode : String;
                    attribute tankPressure_kPa : Real;
                    attribute propRemain_kg : Real;
                    action fire : FireRCS;
                    action report : ReportTelemetry;
                    action monitor : MonitorHealth;
                }
```

```
43                  part def LandingGearAssembly {
44                      attribute gearState : String;
45                      attribute strokeUsage_pct : Real;
46                      action deploy : DeployLandingGear;
47                      action lock : LockLandingGear;
48                      action dampen : DampenLandingLoads;
49                      action stabilize : StabilizeOnSurface;
50                      action report : ReportTelemetry;
51                      action monitor : MonitorHealth;
52                  }
53                  part def LandingSensorSuite {
54                      attribute radarAlt_m : Real;
55                      attribute rate_mps : Real;
56                      attribute contact : Boolean;
57                      action senseContact : SenseContact;
58                      action report : ReportTelemetry;
59                      part radar : RadarAltimeter;
60                      part probes : ContactProbes;
61                  }
62                  part def RadarAltimeter {
63                      attribute altitude_m : Real;
64                      attribute rate_mps : Real;
65                      action report : ReportTelemetry;
66                  }
67                  part def ContactProbes {
68                      attribute contact : Boolean;
69                      action report : ReportTelemetry;
70                  }
71                  part def GuidanceComputer {
72                      attribute mode : String;
73                      attribute updateHz : Integer;
74                      attribute targetSite : String;
75                      action estimate : EstimateState;
76                      action compute : ComputeDescentGuidance;
77                      action genThrottle : GenerateThrottleCommand;
78                      action genGimbal : GenerateGimbalCommand;
79                      action genAtt : GenerateAttitudeCommand;
80                      action report : ReportTelemetry;
81                      action monitor : MonitorHealth;
82                  }
83                  part def CommsInterface {
84                      attribute tlmRate_Hz : Integer;
85                      attribute linkStatus : String;
86                      action report : ReportTelemetry;
87                      action monitor : MonitorHealth;
88                  }
89                  part def PowerInterface {
90                      attribute busVoltage_V : Real;
91                      attribute busHealth : String;
92                      action monitor : MonitorHealth;
```

```
                }
                part def FuelInterface {
                    attribute fuelTemp_K : Real;
                    attribute oxidTemp_K : Real;
                    attribute press_kPa : Real;
                    action monitor : MonitorHealth;
                }
                part def ThermalInterface {
                    attribute loopTemp_K : Real;
                    attribute heatLoad_W : Real;
                    action monitor : MonitorHealth;
                }
                part def CrewInterface {
                    attribute lastCmd : String;
                    attribute cmdSeqNum : Integer;
                    action report : ReportTelemetry;
                }
                part def DescentStructure {
                    attribute mass_kg : Real;
                    attribute margin_pct : Real;
                }
                part def Descent {
                    part engineCtrl : DescentEngineControl;
                    part rcs : RCSModule;
                    part sensors : LandingSensorSuite;
                    part landingGear : LandingGearAssembly;
                    part guidance : GuidanceComputer;
                    part commsIf : CommsInterface;
                    part powerIf : PowerInterface;
                    part fuelIf : FuelInterface;
                    part thermalIf : ThermalInterface;
                    part crewIf : CrewInterface;
                    part structure : DescentStructure;
                }
            }
        }
    }
}
```

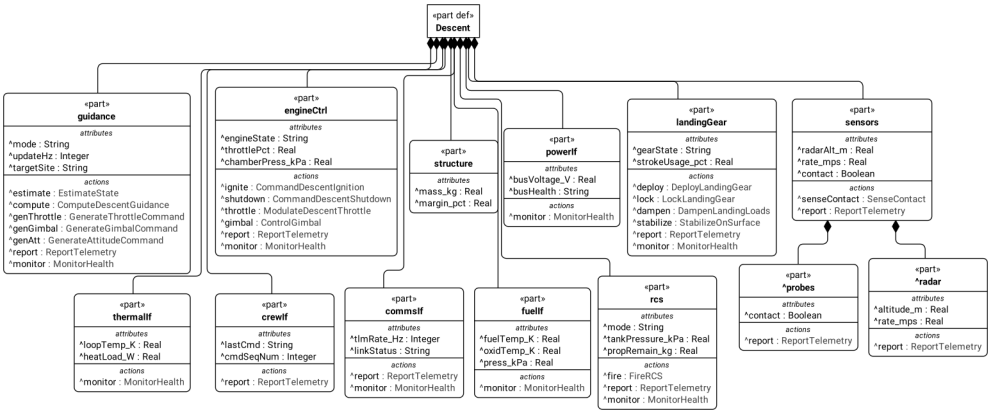


Figure 25. Descent Stage Subsystem Parts Decomposition (General View)

7.2 Requirements

Figure 26. Descent Stage Subsystem Requirements (Textual Notation)

```
package 'Lunar Lander V2 Model' {
    package LunarLanderSubsystemsDecomposition {
        package Descent_Stage {
            package Requirements {
                requirement def DSC_REQ_001_Capability {
                    doc /* The Descent subsystem shall provide controlled
                         * powered descent from lunar orbit to the surface
                         * and deliver the lander to a safe, stable touchdown. */
                }
                requirement def DSC_REQ_002_EngineControl {
                    doc /* The Descent subsystem shall command descent engine
                         * ignition, throttling, thrust vector control,
                         * and shutdown per guidance. */
                }
                requirement def DSC_REQ_003_AttitudeAndRCS {
                    doc /* The Descent subsystem shall control vehicle attitude
                         * throughout descent and provide translation impulses
                         * required during approach. */
                }
                requirement def DSC_REQ_004_GuidanceNavigation {
                    doc /* The Descent subsystem shall estimate state and compute
                         * guidance using onboard sensors to meet landing site
                         * conditions within allocated dispersions. */
                }
                requirement def DSC_REQ_005_LandingSensors {
                    doc /* The Descent subsystem shall provide altitude/rate
                         * sensing and surface contact indication for final
                         * approach and touchdown. */
                }
                requirement def DSC_REQ_006_LandingGear {
                    doc /* The Descent subsystem shall deploy and lock landing gear
                        * prior to touchdown and absorb landing loads within limits. */
                }
                requirement def DSC_REQ_007_Interfaces {
                    doc /* The Descent subsystem shall interface with power, thermal,
                         * communications, crew interface, and propellant/pressurant
                         * supplies required for descent operations. */
                }
                requirement def DSC_REQ_008_FaultManagement {
                    doc /* The Descent subsystem shall detect, annunciate, and support
                         * recovery from credible faults affecting descent engine,
                         ↪  sensors,
                         * landing gear, and guidance control. */
                }
                requirement def DSC_REQ_009_Verification {
```

```
                    doc /* Descent requirements shall be verified by analysis, test,
                         * inspection, and/or demonstration as appropriate. */
                }
            }
        }
    }
}
```

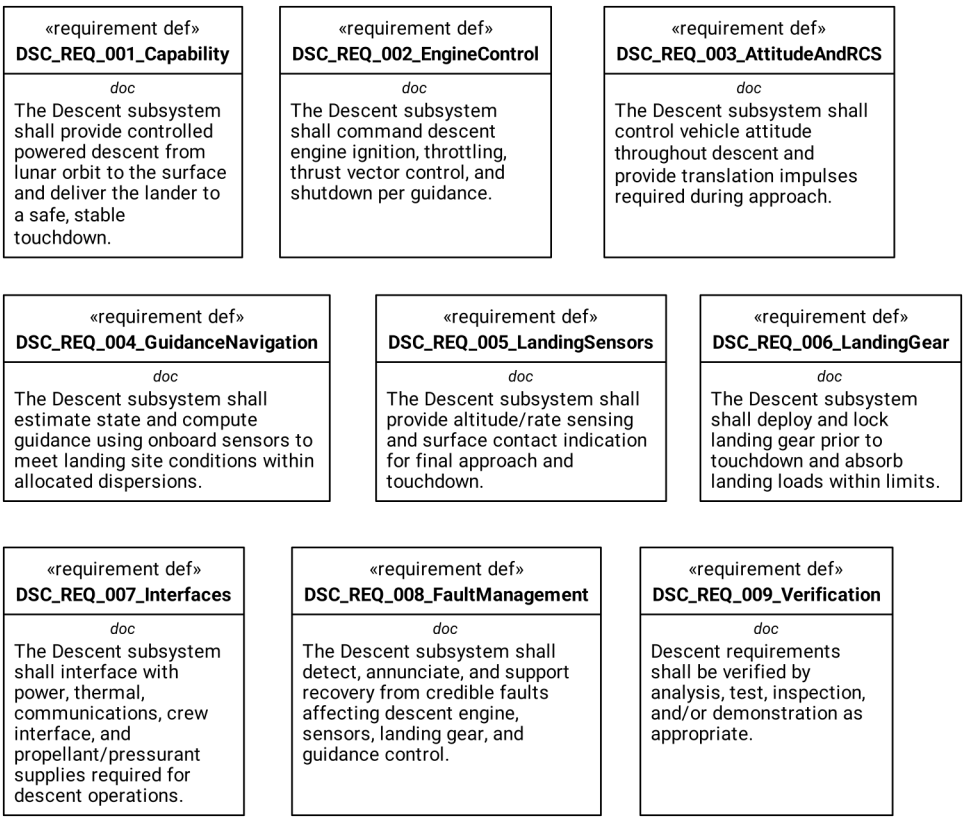


Figure 27. Descent Stage Subsystem Requirements (General View)

7.3 State Machine

Figure 28. Descent Stage Subsystem State Machine (Textual Notation)

```
package 'Lunar Lander V2 Model' {
    package LunarLanderSubsystemsDecomposition {
        package Descent_Stage {
            package StateMachine {
                item def GoDescent;
                item def PDI;
                item def ApproachGate;
                item def ContactLight;
                item def EngineShutdown;
                item def SafeComplete;
                part def DescentSM_Owner {
                    exhibit state DescentSM;
                }
                state def DescentSM {
                    state PreDescentPrep;
                    state PoweredDescent;
                    state Approach;
                    state Landing;
                    state TouchdownSafing;
                    state SurfaceIdle;
                    transition t0 first entryAction then PreDescentPrep;
                    transition t1 first PreDescentPrep accept GoDescent then
                    ↪  PoweredDescent;
                    transition t2 first PoweredDescent accept PDI then Approach;
                    transition t3 first Approach accept ApproachGate then Landing;
                  transition t4 first Landing accept ContactLight then TouchdownSafing;
                    transition t5 first TouchdownSafing accept EngineShutdown then
                    ↪  SurfaceIdle;
                  transition t6 first SurfaceIdle accept SafeComplete then SurfaceIdle;
                }
            }
        }
    }
}
```

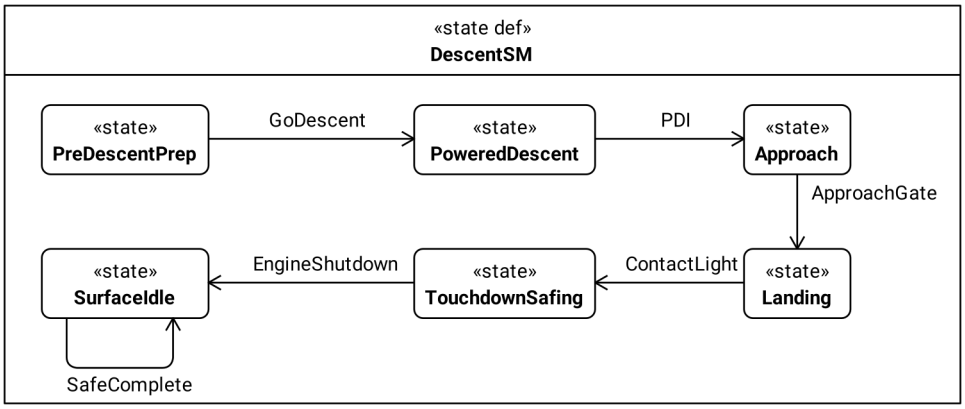


Figure 29. Descent Stage Subsystem State Machine (General View)

7.4 Connections

Figure 30. Descent Stage Subsystem Connections (Textual Notation)

```
package 'Lunar Lander V2 Model' {
    package LunarLanderSubsystemsDecomposition {
        package Descent_Stage {
            package Connections {
                private import ScalarValues::**;
                private import SI::**;
                interface def iPower;
                interface def iPropellant;
                interface def iPressurant;
                interface def iHeat;
                interface def iAltRate;
                interface def iContact;
                interface def iThrottleCmd;
                interface def iGimbalCmd;
                interface def iAttCmd;
                interface def iCrewCmd;
                interface def iDeployCmd;
                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;
                    port tlmIn : iTelemetry;
                    port tlmOut : iTelemetry;
                }
                part def DescentEngineControl {
                    port pwr : iPower;
                    port prop : iPropellant;
                    port press : iPressurant;
                    port throttle : iThrottleCmd;
                    port gimbal : iGimbalCmd;
                    port engTlm : iTelemetry;
                }
                part def RCSModule {
```

```
46                      port pwr : iPower;
47                      port attCmd : iAttCmd;
48                      port rcsTlm : iTelemetry;
49                  }
50                  part def LandingSensorSuite {
51                      port pwr : iPower;
52                      port altRate : iAltRate;
53                      port contact : iContact;
54                  }
55                  part def LandingGearAssembly {
56                      port deploy : iDeployCmd;
57                      port status : iTelemetry;
58                      port contact : iContact;
59                  }
60                  part def GuidanceComputer {
61                      port pwr : iPower;
62                      port altIn : iAltRate;
63                      port contactIn : iContact;
64                      port crewIn : iCrewCmd;
65                      port throttle : iThrottleCmd;
66                      port gimbal : iGimbalCmd;
67                      port att : iAttCmd;
68                      port gearCmd : iDeployCmd;
69                      port tlm : iTelemetry;
70                  }
71                  part def Descent {
72                      part power : PowerIF;
73                      part fuel : FuelIF;
74                      part thermal : ThermalIF;
75                      part crew : CrewIF;
76                      part comms : CommsIF;
77                      part engineCtrl : DescentEngineControl;
78                      part rcs : RCSModule;
79                      part sensors : LandingSensorSuite;
80                      part landingGear : LandingGearAssembly;
81                      part guidance : GuidanceComputer;
82                      interface connect power.bus to engineCtrl.pwr;
83                      interface connect power.bus to rcs.pwr;
84                      interface connect power.bus to sensors.pwr;
85                      interface connect power.bus to guidance.pwr;
86                      interface connect power.bus to comms.pwr;
87                      interface connect fuel.prop to engineCtrl.prop;
88                      interface connect fuel.press to engineCtrl.press;
89                      interface connect guidance.throttle to engineCtrl.throttle;
90                      interface connect guidance.gimbal to engineCtrl.gimbal;
91                      interface connect guidance.att to rcs.attCmd;
92                      interface connect guidance.gearCmd to landingGear.deploy;
93                      interface connect sensors.altRate to guidance.altIn;
94                      interface connect sensors.contact to guidance.contactIn;
95                      interface connect sensors.contact to landingGear.contact;
```

```
96                      interface connect crew.crewCmd to guidance.crewIn;
97                      interface connect engineCtrl.engTlm to comms.tlmIn;
98                      interface connect rcs.rcsTlm to comms.tlmIn;
99                      interface connect landingGear.status to comms.tlmIn;
100                     interface connect guidance.tlm to comms.tlmIn;
101                 }
102             }
103         }
104     }
105 }
```

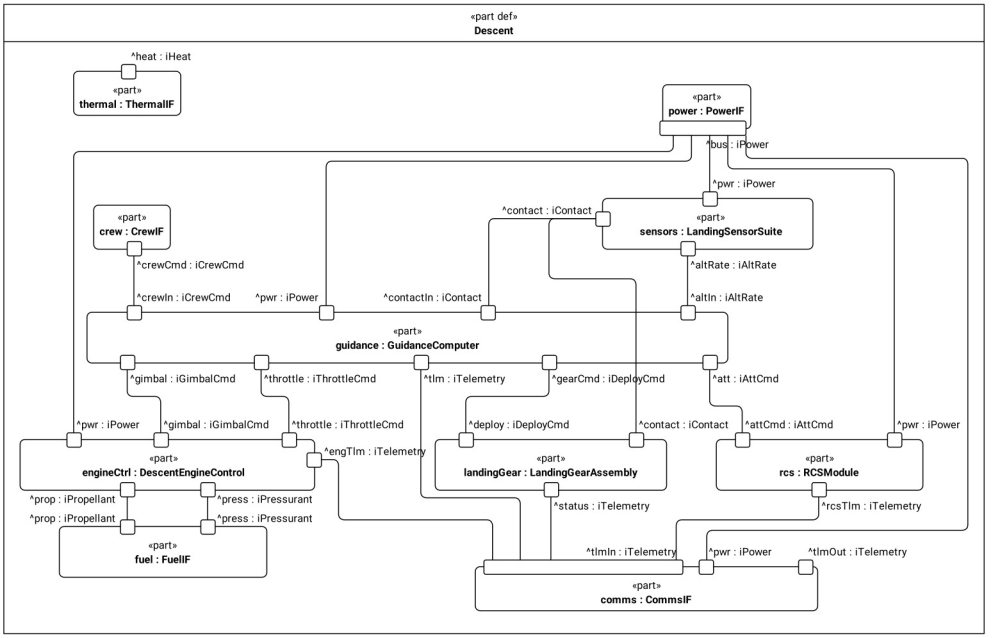
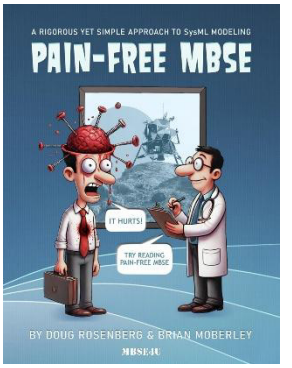


Figure 31. Descent Stage Subsystem Connections (General View)

MBSE4U

Pain-Free MBSE
A Rigorous Yet Simple Approach to SysML Modeling

by Doug Rosenberg, Brian Moberley



Published by MBSE4U, www.mbse4u.com

Buy the Book

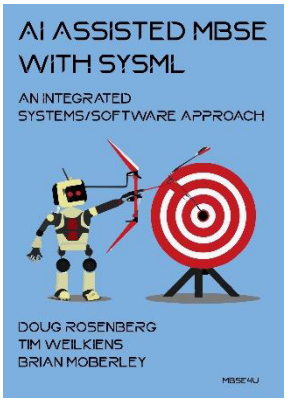
- eBook (PDF/EPUB): https://leanpub.com/pain-free-mbse
- Print: https://www.lulu.com/shop/doug-rosenberg-and-brian-moberley/pain-free-mbse/paperback/product-p6nk69v.html

The book will also be available on Amazon and other bookstores.

Related books

AI-Assisted MBSE with SysML

by Doug Rosenberg, Tim Weilkiens, and Brian Moberley



- eBook (PDF/EPUB):
- Print: https://www.lulu.com/shop/doug-rosenberg-and-tim-weilkiens-and-brian-moberley/ai-assisted-mbse-with-sysml/paperback/product-7kye6qj.html
- Also available on Amazon and other bookstores.