

Systems engineering, MBSE & V&V

As a system grows more complex, it becomes impossible to optimize it by improving each subsystem independently. A larger battery changes mass and thermal design; a more powerful antenna changes power and pointing; a new procedure changes software, training and operations. Systems engineering organizes those dependencies from stakeholder needs to evidence that the realized product actually satisfies the mission.

Before you start — Prerequisites: modules 01 to 09 recommended. Every important symbol is defined again at first use.

Mastery objectives

- explain concepts with units and assumptions
- redo a simple calculation by hand before using a tool
- identify at least one failure mode or model limitation
- connect the discipline to a complete Mars architecture

1. Need, objective and requirement are not the same thing

A need explains why a capability matters. An objective describes what the mission seeks to achieve. A requirement imposes a verifiable property on the system. 'Survive on Mars' is not a good requirement; 'maintain habitat pressure within a defined range for a defined time after a single failure' can become one. A clear requirement has a subject, action, condition and measurable criterion.

Requirement testability check. Convert the mission need into one requirement that contains a measurable threshold, an operating condition and a verification method. If any of those three elements is missing, the requirement is not yet testable.

Approfondissement — A requirement becomes verifiable only when its verb can produce evidence

"The system shall be robust" is not a usable requirement because no test can answer it without interpretation. A strong requirement states a function, condition and criterion. Example: "After loss of one converter, the ECLSS subsystem shall maintain at least 8 kW of available power for 30 min." Here 8 kW is the minimum power and 30 min the duration. Verification can then be planned by test, analysis, inspection or demonstration.

A verification matrix connects every requirement to method, owner, test level and result. Without traceability, a team can run hundreds of tests and still fail to prove that all survival requirements were covered. This is especially dangerous for a Mars habitat because subsystems evolve for years and a critical requirement can disappear during a supplier or architecture change.

2. ConOps: describe how the system will actually be used

A Concept of Operations describes actors, phases, environments, information flows, states and nominal or degraded scenarios. It prevents teams from designing machinery without understanding the human work around it. An excellent technical requirement can be useless if the operational scenario is wrong. ConOps therefore connects mission, operations and architecture.

CONOPS review rule. Write the CONOPS as a time-ordered sequence of actors, commands, resources and off-nominal branches. A useful review asks what the crew or software does when the planned next step is unavailable.

3. Functional decomposition and architecture

Start from required functions, then decide which physical or software elements perform them. This prevents premature attachment to a favourite solution. The function 'reject heat' may be implemented through different combinations of radiators, loops and operating modes. Architecture then allocates functions, performance and interfaces to subsystems.

Functional decomposition rule. Start functional decomposition with verbs rather than hardware names: provide power, reject heat, acquire data, isolate a leak. Only after the functions are stable should they be allocated to physical subsystems.

4. Interfaces: where good subsystems can become a bad system

An interface is more than a connector. It includes mechanical, electrical, data, thermal, software, procedural and responsibility boundaries. Units, tolerances, allowed states and fault behaviour must be defined. Integration problems often come from implicit assumptions made differently by two teams.

Interface-control rule. For every interface, record what crosses the boundary: energy, material, data, force or responsibility. Then assign an owner and an acceptance test so that no interface remains somebody else's assumption.

Approfondissement — Interface management: systems often fail at boundaries between competent teams

An interface defines what crosses a boundary: energy, data, fluid, mechanical load, heat, volume, schedule or operational responsibility. An electrical interface may specify voltage, maximum current, connector, pinout, grounding and fault behavior. A data interface adds protocol, throughput, format, synchronization and error handling. A mechanical interface includes dimensions, stiffness, fastener torque and allowable loads.

An interface document works only if it has ownership and configuration control. If team A changes a connector while team B continues manufacturing the old mate, both subsystems can be perfectly compliant with their own files and the integrated system still fails. MBSE, Model-Based Systems Engineering, can represent dependencies, but a digital model does not replace governance: who may change the interface, who approves the change and how does every affected team learn about it?

5. Budgets and margins as decision tools

Mass, power, energy, data rate, volume, crew time and performance are tracked as budgets. Margin is not a decorative percentage: it absorbs uncertainty and maturation. When margin shrinks, the decision should be visible. Budgets create a common language between disciplines that might otherwise optimize different criteria.

Budget and margin rule. Treat each budget as a decision ledger rather than a spreadsheet total. Keep allocation, best estimate, uncertainty and reserve separate so reviewers can see whether margin is real or merely hidden optimism.

6. MBSE: use models to connect engineering products

Model-Based Systems Engineering (MBSE) is not about producing attractive diagrams. A useful model connects requirements, functions, components, interfaces and V&V evidence so that a change leaves a trace. SysML can formalize these relations. The model does not replace judgement; it primarily reduces hidden inconsistency between independent documents.

MBSE traceability rule. In MBSE, link each requirement to the function that satisfies it, the component that implements it and the evidence that verifies it. A model is valuable when a change can reveal downstream consequences before hardware is modified.

7. Verification and validation ask different questions

Verification asks whether the product meets its requirements through analysis, inspection, demonstration or test. Validation asks whether the product is actually suitable for mission use and stakeholder expectations. A system can be perfectly verified against bad requirements and still be useless. Both loops must remain visible from the beginning.

Verification-versus-validation rule. Use different evidence for verification and validation: verification asks whether the specified requirement was met, whereas validation asks whether that requirement solves the real mission need in the intended context.

Approfondissement — Verification, validation and qualification answer three different questions

Verification asks, “did we build it to the requirements?” Validation asks, “do the requirements describe what the mission actually needs?” It is possible to verify the wrong need perfectly. Qualification adds evidence that a design survives its environment with required margin. A valve may meet nominal flow on a bench and later fail after dust and temperature cycling: the function was verified in a weak context, not qualified for Mars.

A V&V strategy, meaning Verification and Validation, therefore climbs from component to system: unit tests, assemblies, integrated hardware, simulation, environmental testing and operational demonstrations. Some functions can only be validated with representative humans. An alarm may be technically audible and still impossible to distinguish among fifty other alarms. Systems engineering has to connect hardware, software, operations and human factors instead of validating each discipline in isolation.

8. Verification matrix and traceability

A verification matrix records how compliance will be demonstrated. For every important requirement, identify a method, level, test configuration, owner and expected evidence. This prevents discovering late that a requirement cannot be proven. Traceability also answers the reverse question: if a test fails, which requirements, functions and decisions are affected?

Verification-matrix review. Read a verification matrix row by row and look for orphan requirements, duplicate tests and evidence that cannot be traced to an acceptance criterion. A blank evidence path is a programme risk, not a documentation nuisance.

9. Configuration and change

Space systems evolve. Configuration management establishes what constitutes the reference at a given time: requirements, software, drawings, parameters and procedures. A modification must be assessed for cross-system effects. Without a baseline and change control, a local fix can reintroduce a failure elsewhere — exactly what anti-regression work seeks to prevent.

Configuration-control rule. Before approving a configuration change, trace its effects through interfaces, mass, power, thermal control, software, procedures and verification evidence. Baseline the approved state so later teams know exactly what configuration was tested.

Worked example step by step

Traceability example: need → requirement R-042 → function → component → verification method → test result. A broken chain is technical debt.

The work method is always the same: state what every symbol represents, convert all units into a coherent system, perform the operation, then translate the result into a sentence. Finally perform an order-of-magnitude check. If the answer changes by a factor of one thousand because millimetres were treated as metres, the conversion must be visible in the calculation.

Progressive exercise

1. Choose a simple case and list every input with units.
2. Compute the nominal result without margin.
3. Vary the most uncertain parameter by $\pm 20\%$ and compare.
4. Inject one credible failure and explain which indicator detects it.
5. Decide whether the system continues, degrades or stops.

Detailed solution — reasoning and decision

Reasoned solution

A good solution is not only the final number. It shows conversions, why the equation applies, sensitivity and the resulting decision. If different plausible assumptions lead to the same operational decision, the design is relatively robust to that uncertainty. If a small variation reverses the decision, the parameter becomes a priority for measurement or margin.

Validation mini-project

Build a two-to-four-page engineering note applying this course to one Mars subsystem. Include need, assumptions, functional sketch, hand calculation, second calculation or simulation, uncertainties, injected failure, decision criteria and three primary references. The goal is a chain of evidence that another reader can reproduce.

Common errors to detect

- mixing units or frames without explicit conversion;
- presenting calculated values as measured data;
- ignoring a model's validity range;
- confusing numerical precision with physical accuracy;
- sizing only the nominal case with no margin or degraded mode.

Primary sources and pathways

- [NASA — Systems Engineering Handbook](#)
- [NASA — Fundamentals of Systems Engineering](#)

Systems-engineering foundations and mission reasoning

This English systems-engineering module now follows the certified French teaching standard: needs become testable requirements, interfaces are explicit, evidence is traceable, and exercises require the learner to decide rather than copy an answer.

Four concepts to master first

requirement

Definition. A requirement states a property or performance that must be demonstrably satisfied, including the conditions and acceptance criterion.

Mission example. A water system requirement can specify the minimum potable-water flow in a named operating mode.

Pitfall. Avoid words such as reliable, adequate or fast unless a measurable threshold makes them testable.

Mental check. Ask whether two independent reviewers would know exactly what evidence proves compliance.

Evidence

Use an approved requirement record plus its measured test or analysis result showing the stated criterion under the declared operating mode.

Decision use

If the criterion is not met or cannot be verified, the requirement must be revised, the design changed or acceptance withheld.

interface

Definition. An interface is a boundary where two elements exchange matter, energy, data, forces, heat or operational responsibility.

Mission example. An airlock interfaces mechanically, electrically, pneumatically, digitally and procedurally with the habitat and suit.

Pitfall. An interface list that names hardware only can miss software timing, human actions or contamination paths.

Mental check. Trace what crosses the boundary, in which direction, under which mode and who owns the constraint.

Evidence

Use an interface-control record, connector or data specification, and an integration test showing that exchanged flows stay within agreed limits.

Decision use

A mismatch requires an interface change, compatibility fix or operational restriction before the connected elements are accepted together.

verification

Definition. Verification asks whether the built system satisfies its stated requirements through test, analysis, inspection or demonstration.

Mission example. A pressure vessel may be verified by structural analysis plus a qualification pressure test.

Pitfall. Passing one attractive demonstration does not prove every applicable requirement.

Mental check. Map every requirement to an accepted verification method and a recorded result.

Evidence

Use the signed test report, analysis package, inspection record or demonstration result tied directly to the requirement identifier.

Decision use

Failed or incomplete verification keeps the requirement open and blocks acceptance until valid evidence closes the discrepancy.

traceability

Definition. Traceability links needs, requirements, architecture, analyses, interfaces, tests, anomalies and accepted evidence.

Mission example. Changing one pressure sensor should reveal which requirements, software functions and tests must be reconsidered.

Pitfall. A beautiful model without bidirectional traceability can hide orphan requirements and unverified design choices.

Mental check. Pick one requirement and follow the chain both toward the original need and toward the final verification record.

Evidence

Use the requirement database links that connect mission need, requirement, architecture element, change record and verification evidence.

Decision use

Broken traceability triggers a review of affected evidence because the team can no longer prove which design or test supports the current baseline.

Calculation laboratory

Treat each systems-engineering relationship as an audit trail. Name the requirement or interface being checked, keep units visible where quantities appear, and finish by identifying the evidence needed for acceptance.

Canonical quantitative mini-lessons

This block is the module quantitative reference; exercise values are pedagogical and must not be confused with mission data.

Verification coverage

$$C_verify = N_accepted / N_applicable$$

- 1 — Concrete question
What does “ $C_verify = N_accepted / N_applicable$ ” compute in “Verification coverage”?
- 2 — Intuition without symbols
Coverage shows what fraction of verification obligations already has accepted evidence.
- 3 — Quantities
 C_verify : verification coverage; $N_accepted$: requirements closed by accepted evidence; $N_applicable$: applicable requirements
- 4 — Formula
 $C_verify = N_accepted / N_applicable$
- 5 — Read aloud
Read “ $C_verify = N_accepted / N_applicable$ ” by naming every operation, subscript and grouping explicitly.
- 6 — Symbols and meaning
 C_verify : verification coverage; $N_accepted$: requirements closed by accepted evidence; $N_applicable$: applicable requirements
- 7 — Pronunciation
Say subscripts, powers, Greek letters and parentheses explicitly so the relation can be reconstructed without seeing the page.
- 8 — Units
counts dimensionless; C_verify may be shown as a percentage
- 9 — Convention
Keep the same frame, time basis, system boundary and sign convention throughout the substitution.
- 10 — Why this operation
Division compares one quantity with a homogeneous reference and produces a proportion indicator.
- 11 — Assumptions
Both quantities must belong to the same scope and time period.
- 12 — Unit check
counts dimensionless; C_verify may be shown as a percentage Verify that dimensional reduction reaches the unit of the requested output.
- 13 — Numerical case
With $N_accepted=171$ and $N_applicable=180$, $C_verify=0.95=95\%$.
- 14 — Why the calculation works
Division compares one quantity with a homogeneous reference and produces a proportion indicator.
- 15 — Independent check
Multiplying the result by the denominator must recover the numerator.
- 16 — Mental estimate
Before calculating, compare numerator and denominator mentally to predict whether the result should be below, near or above one.
- 17 — Interpretation
High coverage says nothing about the quality or criticality of the remaining evidence.
- 18 — What the result does not prove
The calculation proves neither that the inputs are correct nor that the model remains valid outside the declared scenario; those questions require separate evidence.
- 19 — Sensitivity
The result rises with the numerator and falls with the denominator, all else equal.
- 20 — Guided and autonomous exercises
Guided exercise. $N_accepted=284$, $N_applicable=320$.

Detailed guided correction — open after trying

$N_accepted=284$, $N_applicable=320$. $C_verify=0.8875=88.75\%$.

Autonomous exercise. $N_{\text{accepted}}=95$, $N_{\text{applicable}}=100$.

Autonomous correction — open after trying

$N_{\text{accepted}}=95$, $N_{\text{applicable}}=100$. $C_{\text{verify}}=95\%$.

21 — Mission decision

Close critical requirements and blocking evidence first rather than merely maximising percentage.

Absolute mass margin

$M_{\text{mass}} = M_{\text{allocated}} - M_{\text{estimated}}$

1 — Concrete question

What does " $M_{\text{mass}} = M_{\text{allocated}} - M_{\text{estimated}}$ " compute in "Absolute mass margin"?

2 — Intuition without symbols

Mass margin measures how much allocation remains after the current estimate.

3 — Quantities

M_{mass} : mass margin; $M_{\text{allocated}}$: maximum allocation; $M_{\text{estimated}}$: current estimated mass

4 — Formula

$M_{\text{mass}} = M_{\text{allocated}} - M_{\text{estimated}}$

5 — Read aloud

Read " $M_{\text{mass}} = M_{\text{allocated}} - M_{\text{estimated}}$ " by naming every operation, subscript and grouping explicitly.

6 — Symbols and meaning

M_{mass} : mass margin; $M_{\text{allocated}}$: maximum allocation; $M_{\text{estimated}}$: current estimated mass

7 — Pronunciation

Say subscripts, powers, Greek letters and parentheses explicitly so the relation can be reconstructed without seeing the page.

8 — Units

all masses in the same unit

9 — Convention

Keep the same frame, time basis, system boundary and sign convention throughout the substitution.

10 — Why this operation

Subtraction isolates a margin or change between two quantities expressed on the same basis.

11 — Assumptions

Both terms must use the same frame, unit and system boundary.

12 — Unit check

all masses in the same unit Verify that dimensional reduction reaches the unit of the requested output.

13 — Numerical case

With $M_{\text{allocated}}=1,000$ kg and $M_{\text{estimated}}=860$ kg, $M_{\text{mass}}=140$ kg.

14 — Why the calculation works

Subtraction isolates a margin or change between two quantities expressed on the same basis.

15 — Independent check

Adding the difference back to the subtracted term must recover the initial term.

16 — Mental estimate

Comparing the two terms is enough to predict the sign and scale of the difference.

17 — Interpretation

Positive margin may still be insufficient when the estimate is immature or reserves must be protected.

18 — What the result does not prove

The calculation proves neither that the inputs are correct nor that the model remains valid outside the declared scenario; those questions require separate evidence.

19 — Sensitivity

Increasing the first term raises the result; increasing the second lowers it with unit slope.

20 — Guided and autonomous exercises

Guided exercise. $M_{\text{allocated}}=500$ kg, $M_{\text{estimated}}=455$ kg.

Detailed guided correction — open after trying

$M_{\text{allocated}}=500$ kg, $M_{\text{estimated}}=455$ kg. $M_{\text{mass}}=45$ kg.

Autonomous exercise. $M_{\text{allocated}}=2,000$ kg, $M_{\text{estimated}}=1,920$ kg.

Autonomous correction — open after trying

M_allocated=2,000 kg, M_estimated=1,920 kg. M_mass=80 kg.

21 — Mission decision

Associate every margin with maturity, risks and the programme reserve policy.

Margin as a percentage of allocation

$$M_{\text{pct}} = 100 \times M_{\text{mass}} / M_{\text{allocated}}$$

1 — Concrete question

What does “ $M_{\text{pct}} = 100 \times M_{\text{mass}} / M_{\text{allocated}}$ ” compute in “Margin as a percentage of allocation”?

2 — Intuition without symbols

Expressing margin relative to allocation enables comparison across subsystems of different sizes.

3 — Quantities

M_{pct} : margin percentage; M_{mass} : absolute margin; $M_{\text{allocated}}$: maximum allocation

4 — Formula

$$M_{\text{pct}} = 100 \times M_{\text{mass}} / M_{\text{allocated}}$$

5 — Read aloud

Read “ $M_{\text{pct}} = 100 \times M_{\text{mass}} / M_{\text{allocated}}$ ” by naming every operation, subscript and grouping explicitly.

6 — Symbols and meaning

M_{pct} : margin percentage; M_{mass} : absolute margin; $M_{\text{allocated}}$: maximum allocation

7 — Pronunciation

Say subscripts, powers, Greek letters and parentheses explicitly so the relation can be reconstructed without seeing the page.

8 — Units

M_{pct} in percent; masses in the same unit

9 — Convention

Keep the same frame, time basis, system boundary and sign convention throughout the substitution.

10 — Why this operation

Division compares one quantity with a homogeneous reference and produces a proportion indicator.

11 — Assumptions

Both quantities must belong to the same scope and time period.

12 — Unit check

M_{pct} in percent; masses in the same unit. Verify that dimensional reduction reaches the unit of the requested output.

13 — Numerical case

With $M_{\text{mass}}=140$ kg and $M_{\text{allocated}}=1,000$ kg, $M_{\text{pct}}=14\%$.

14 — Why the calculation works

Division compares one quantity with a homogeneous reference and produces a proportion indicator.

15 — Independent check

Multiplying the result by the denominator must recover the numerator.

16 — Mental estimate

Before calculating, compare numerator and denominator mentally to predict whether the result should be below, near or above one.

17 — Interpretation

The percentage depends on the chosen denominator, which must always be named explicitly.

18 — What the result does not prove

The calculation proves neither that the inputs are correct nor that the model remains valid outside the declared scenario; those questions require separate evidence.

19 — Sensitivity

The result rises with the numerator and falls with the denominator, all else equal.

20 — Guided and autonomous exercises

Guided exercise. $M_{\text{mass}}=45$ kg, $M_{\text{allocated}}=500$ kg.

Detailed guided correction — open after trying

$M_{\text{mass}}=45$ kg, $M_{\text{allocated}}=500$ kg. $M_{\text{pct}}=9\%$.

Autonomous exercise. $M_{\text{mass}}=80$ kg, $M_{\text{allocated}}=2,000$ kg.

Autonomous correction — open after trying

$M_{\text{mass}}=80$ kg, $M_{\text{allocated}}=2,000$ kg. $M_{\text{pct}}=4\%$.

Compare margins only when definition and denominator are identical.

Traceability coverage

$$C_{\text{trace}} = N_{\text{traced}} / N_{\text{total}}$$

1 — Concrete question

What does “ $C_{\text{trace}} = N_{\text{traced}} / N_{\text{total}}$ ” compute in “Traceability coverage”?

2 — Intuition without symbols

Traceability is complete only when every requirement can trace upward to need and downward to evidence.

3 — Quantities

C_{trace} : traceability coverage; N_{traced} : requirements linked to a source and evidence; N_{total} : requirements in scope

4 — Formula

$$C_{\text{trace}} = N_{\text{traced}} / N_{\text{total}}$$

5 — Read aloud

Read “ $C_{\text{trace}} = N_{\text{traced}} / N_{\text{total}}$ ” by naming every operation, subscript and grouping explicitly.

6 — Symbols and meaning

C_{trace} : traceability coverage; N_{traced} : requirements linked to a source and evidence; N_{total} : requirements in scope

7 — Pronunciation

Say subscripts, powers, Greek letters and parentheses explicitly so the relation can be reconstructed without seeing the page.

8 — Units

counts dimensionless; C_{trace} may be shown as a percentage

9 — Convention

Keep the same frame, time basis, system boundary and sign convention throughout the substitution.

10 — Why this operation

Division compares one quantity with a homogeneous reference and produces a proportion indicator.

11 — Assumptions

Both quantities must belong to the same scope and time period.

12 — Unit check

counts dimensionless; C_{trace} may be shown as a percentage Verify that dimensional reduction reaches the unit of the requested output.

13 — Numerical case

With $N_{\text{traced}}=240$ and $N_{\text{total}}=250$, $C_{\text{trace}}=0.96=96\%$.

14 — Why the calculation works

Division compares one quantity with a homogeneous reference and produces a proportion indicator.

15 — Independent check

Multiplying the result by the denominator must recover the numerator.

16 — Mental estimate

Before calculating, compare numerator and denominator mentally to predict whether the result should be below, near or above one.

17 — Interpretation

A present but wrong or obsolete link is not quality traceability.

18 — What the result does not prove

The calculation proves neither that the inputs are correct nor that the model remains valid outside the declared scenario; those questions require separate evidence.

19 — Sensitivity

The result rises with the numerator and falls with the denominator, all else equal.

20 — Guided and autonomous exercises

Guided exercise. $N_{\text{traced}}=178$ and $N_{\text{total}}=200$.

Detailed guided correction — open after trying

$N_{\text{traced}}=178$ and $N_{\text{total}}=200$. $C_{\text{trace}}=89\%$.

Autonomous exercise. $N_{\text{traced}}=99$ and $N_{\text{total}}=100$.

Autonomous correction — open after trying

$N_{\text{traced}}=99$ and $N_{\text{total}}=100$. $C_{\text{trace}}=99\%$.

Interface closure

$$C_interface = N_closed / N_total$$

1 — Concrete question

What does “C_interface = N_closed / N_total” compute in “Interface closure”?

2 — Intuition without symbols

Open interfaces often concentrate risks that are invisible in each subsystem’s standalone performance.

3 — Quantities

C_interface: closure rate; N_closed: technically closed interfaces; N_total: interfaces in scope

4 — Formula

$$C_interface = N_closed / N_total$$

5 — Read aloud

Read “C_interface = N_closed / N_total” by naming every operation, subscript and grouping explicitly.

6 — Symbols and meaning

C_interface: closure rate; N_closed: technically closed interfaces; N_total: interfaces in scope

7 — Pronunciation

Say subscripts, powers, Greek letters and parentheses explicitly so the relation can be reconstructed without seeing the page.

8 — Units

counts dimensionless; C_interface may be shown as a percentage

9 — Convention

Keep the same frame, time basis, system boundary and sign convention throughout the substitution.

10 — Why this operation

Division compares one quantity with a homogeneous reference and produces a proportion indicator.

11 — Assumptions

Both quantities must belong to the same scope and time period.

12 — Unit check

counts dimensionless; C_interface may be shown as a percentage Verify that dimensional reduction reaches the unit of the requested output.

13 — Numerical case

With N_closed=45 and N_total=50, C_interface=0.90=90%.

14 — Why the calculation works

Division compares one quantity with a homogeneous reference and produces a proportion indicator.

15 — Independent check

Multiplying the result by the denominator must recover the numerator.

16 — Mental estimate

Before calculating, compare numerator and denominator mentally to predict whether the result should be below, near or above one.

17 — Interpretation

The rate does not distinguish a minor interface from a vital one; a criticality register remains essential.

18 — What the result does not prove

The calculation proves neither that the inputs are correct nor that the model remains valid outside the declared scenario; those questions require separate evidence.

19 — Sensitivity

The result rises with the numerator and falls with the denominator, all else equal.

20 — Guided and autonomous exercises

Guided exercise. N_closed=72 and N_total=80.

Detailed guided correction — open after trying

N_closed=72 and N_total=80. C_interface=90%.

Autonomous exercise. N_closed=38 and N_total=40.

Autonomous correction — open after trying

N_closed=38 and N_total=40. C_interface=95%.

21 — Mission decision

Mission reasoning

From mission need to testable requirement

A Mars settlement needs statements that survive design changes. “Keep the crew supplied with water” is a need. A requirement becomes testable only after the operating mode, quantity, duration and acceptance rule are named. Requirement quality matters because vague language propagates uncertainty into architecture, procurement and validation. In a review, engineers should be able to point to one sentence and answer four questions: what must happen, under which condition, how much is enough, and what evidence will prove it.

Architecture and interfaces

MBSE is useful when the model exposes relationships rather than merely drawing boxes. A power subsystem, habitat, rover fleet and communications network share interfaces that can carry hidden constraints. For example, a rover charger may be electrically compatible but operationally unusable if charge time collides with EVA return windows. Interface control therefore combines physical connectors, data protocols, timing, thermal loads, maintenance access and crew procedures. A change is safe only after affected interfaces are traced and re-verified.

Verification versus validation

Verification asks whether the system was built according to its requirements. Validation asks whether the resulting system actually meets the real mission need in its intended environment. A perfectly verified procedure can still fail validation if it takes too long during an emergency. Mars programmes therefore need both levels: requirement-by-requirement evidence and scenario-based demonstrations in representative conditions. The distinction prevents teams from equating paperwork closure with mission usefulness.

Configuration control and evidence

Every accepted result belongs to a configuration. A test performed on software version 3.2 and valve design A cannot automatically verify software 4.0 and valve design B. Configuration management records what was tested, with which calibration, under which assumptions and against which requirement revision. When hardware changes, impact analysis determines whether old evidence remains valid. This discipline is not bureaucracy for its own sake; it prevents a future crew from relying on proof that no longer describes the flown system.

Systems-engineering practice — attempt each problem before revealing its answer

Exercise A — Requirement rewrite

Rewrite “the oxygen system must be reliable” as a testable requirement for one specified mission mode. Include an observable performance threshold and a verification method.

Detailed correction — Exercise A

A defensible answer could state that, in nominal habitat mode with the declared crew size, the oxygen-delivery function shall remain above a stated flow or partial-pressure threshold for the specified duration, with compliance verified by an instrumented integrated test. The exact threshold must come from the system requirement set, not be invented. The key is condition + measurable criterion + evidence method.

Before accepting this requirement, check that the operating mode, measured quantity and verification method are all explicit; a statement that cannot be tested is not yet a usable requirement.

Exercise B — Interface audit

For a Mars airlock, identify five different interface categories and name one failure that could cross each boundary.

Detailed correction — Exercise B

Mechanical: hatch alignment or seal damage. Electrical: loss of heater or actuator power. Fluid: pressure or contamination transfer. Digital: stale state information or command mismatch. Human/procedural: an incorrect sequence or ambiguous indication. A strong interface audit also assigns ownership and identifies how each boundary is verified.

Check that each airlock interface names both what crosses the boundary and the failure that could propagate across it; a list of components alone does not constitute an interface audit.

Exercise C — Verification plan

A structural pressure requirement can be checked by analysis, test, inspection or demonstration. Propose a justified verification combination rather than choosing only one method.

Detailed correction — Exercise C

Use analysis to predict stresses across load cases and a qualification or proof test to expose manufacturing and modelling errors. Inspection can confirm dimensions, weld condition and configuration before the test. The final method mix depends on consequence, uncertainty and destructiveness; the reasoning must show why combined evidence is sufficient.

Confirm that the chosen verification method can actually expose the structural failure mode and that the acceptance criterion is quantitative enough to produce an unambiguous pass or fail.

Exercise D — Traceability impact

A CO2 sensor model is replaced late in the programme. List the records that should be revisited before accepting the substitution.

Detailed correction — Exercise D

Revisit the originating need, requirements involving range/accuracy/latency, interface-control records, power and software assumptions, calibration data, hazard analysis, verification procedures, previous test evidence and operational procedures. Traceability is valuable precisely because it makes that impact set discoverable.

Before approving the sensor replacement, trace the change through requirement, interface, software, calibration, test evidence and configuration records so no dependent evidence remains silently obsolete.

Exercise E — Coverage decision

A review shows 284 accepted requirements out of 320 applicable ones. Calculate verification coverage, identify what the number does not prove, and state the next management question.

Detailed correction — Exercise E

$284/320 = 0.8875 = 88.75\%$. The percentage does not prove that requirements are correct, independent, complete or validated in realistic operations. The next question is which 36 requirements remain open and whether any of them protect crew-critical functions or gate the current configuration.

Recompute $284/320$ as a percentage and then identify the 36 requirements still lacking accepted evidence; coverage is a management indicator, not proof that the system is safe.

Interactive beginner glossary

Use the interactive terms as a working systems-engineering vocabulary. Each definition should help you decide what must be specified, controlled, verified or traced in the scenario.

- **system** — A system is a set of interacting elements organized to deliver a defined function within a declared boundary and environment. Its behavior depends on both the elements and their interfaces.
- **subsystem** — A subsystem is a specialized part of a larger system, such as power, thermal control, propulsion or life support, with its own functions and interfaces.
- **requirement** — A requirement states a measurable property, function or performance that the system must satisfy. A useful requirement is unambiguous and can be verified by evidence.
- **mission need** — A mission need is the operational problem or capability that justifies the system. It precedes detailed requirements and explains why the mission needs a particular outcome.
- **acceptance criterion** — An acceptance criterion is the explicit pass/fail condition used to decide whether a requirement has been met, including the quantity, limit, operating condition and allowed tolerance.
- **interface** — An interface is a boundary where two elements exchange matter, energy, data, force, heat or operational responsibility. Interface failures can propagate between otherwise healthy subsystems.
- **interface control** — Interface control is the disciplined definition and management of what crosses an interface, in what form, with what limits and who owns each side of the boundary.
- **verification** — Verification asks whether the product was built in accordance with its stated requirements. Evidence can come from test, analysis, inspection or demonstration.
- **validation** — Validation asks whether the resulting product is suitable for the intended mission and user need. A system can verify correctly yet fail validation if the original requirements were wrong.
- **traceability** — Traceability links mission needs to requirements, architecture, analyses, tests, anomalies and final evidence so that every decision can be followed in both directions.
- **MBSE** — Model-Based Systems Engineering (MBSE) connects requirements, functions, architecture, interfaces and evidence in a coherent model rather than leaving them scattered across unrelated documents.
- **architecture** — System architecture is the organized arrangement of functions, physical elements, interfaces and responsibilities that explains how the whole system will satisfy mission needs.

- **configuration** — Configuration is the exact controlled state of a system at a given time: hardware version, software version, settings, documents and approved modifications.
- **configuration control** — Configuration control is the process that authorizes, records and tracks changes so tests and decisions always refer to a known hardware, software and documentation state.
- **baseline** — A baseline is an approved reference configuration used as the starting point for further work. Changes after baselining require controlled review so evidence remains traceable.
- **evidence** — Evidence is objective information supporting a technical claim, such as test data, analysis results, inspection records, demonstrations or qualified operational history.
- **test** — A test obtains evidence by operating hardware or software under defined conditions and measuring the result against acceptance criteria.
- **analysis** — Analysis obtains evidence by calculation, modelling or structured reasoning rather than by directly operating the complete system. Its assumptions and model limits must be stated.
- **inspection** — Inspection verifies a property by direct examination, measurement or record review, for example checking dimensions, workmanship, labels, connectors or documentation.
- **demonstration** — A demonstration verifies a function by showing that it can be performed under defined conditions, usually without the detailed quantitative instrumentation of a formal test.
- **hazard** — A hazard is a condition or source with the potential to cause injury, loss of mission, damage or unacceptable environmental effects. Risk depends on both consequence and likelihood.
- **assumption** — An assumption is a condition accepted as true for an analysis or decision when it has not yet been fully demonstrated. Important assumptions must be visible and later confirmed or bounded.
- **constraint** — A constraint is a limit that restricts the design space, such as mass, power, schedule, temperature, launch volume, regulation or an interface that cannot be changed.
- **change impact** — Change impact is the set of consequences produced by modifying a requirement, component, interface or configuration, including effects on analyses, tests, documentation and other subsystems.
- **verification matrix** — A verification matrix maps each requirement to its verification method, acceptance criteria, planned evidence, responsible owner and verification status so no requirement is left unproven.

Operational depth: from calculation to mission decision

Requirement quality under pressure

In an emergency programme review, a weak requirement usually reveals itself by disagreement. One reviewer imagines nominal operation, another imagines emergency operation, and a supplier interprets the same adjective differently. The cure is not more prose but more structure. A useful requirement names the system state, the stimulus or condition, the observable response, the threshold and the verification approach. That structure lets a later crew understand why the requirement existed, while also letting a test engineer reproduce the evidence without asking the original author what the sentence was supposed to mean.

MBSE as a relationship map

Model-based systems engineering earns its value when relationships remain navigable. A block diagram alone is not MBSE. The model should let an engineer move from a crew need to a requirement, from that requirement to a function, from the function to hardware or software, then onward to interfaces, hazards and verification evidence. The practical test is change impact: if a valve, sensor, software task or operating mode changes, the model should reveal which assumptions and verification artefacts are now suspect. A model that cannot answer that question is documentation, not decision support.

Verification evidence hierarchy

Evidence has different strength depending on the claim. Inspection can prove a connector is installed and labelled, but it cannot by itself prove thermal survival. Analysis can cover many load cases, but its credibility depends on model validation and conservative assumptions. Testing exposes real hardware behaviour but may cover only a limited sample or environment. Demonstration can show an operational sequence but may not reveal hidden margins. Mature verification plans combine methods so that one method compensates for the blind spots of another, especially for crew-critical functions.

Validation with realistic humans

A technically compliant system can still fail its mission if operators cannot use it under realistic conditions. Validation therefore includes representative crew, lighting, gloves, delays, dust, workload and degraded modes. The question changes from “did the unit meet its specification?” to “can the mission need actually be met with this integrated system?” That distinction is particularly important on Mars because maintenance, communications and emergency response happen with fewer external resources. Validation scenarios should therefore include recovery after mistakes, not just a flawless nominal sequence.

Decision review discipline

Before a design review closes an item, the team should ask what remains uncertain and what evidence would change the decision. A verification percentage can create false comfort if the open requirements are concentrated in life support, pressure integrity or abort capability. Management therefore combines quantitative coverage with criticality. The same 90% coverage can represent a nearly complete benign subsystem or an unacceptable crew-safety gap. Good governance exposes the identity and consequence of the remaining open items instead of celebrating the percentage in isolation.

Final mission assurance note

A final systems-engineering habit is to distinguish closure from confidence. A requirement can be marked verified because the specified test passed, yet reviewers may still have low confidence if the test environment omitted a dominant Mars condition. Conversely, a difficult requirement may rely on several mutually reinforcing analyses and demonstrations. Review boards should therefore record not only status but evidence quality, unresolved assumptions and configuration relevance. This avoids a dashboard culture in which green cells hide weak proof. The same discipline should extend to waivers: every waiver needs rationale, consequence, duration and an owner for later review.

Mission integration note

A requirements database should also preserve rationale. Rationale explains why a threshold exists, what hazard or mission need it protects, and what trade study led to the selected value. That information becomes crucial years later when a new team asks whether the requirement can be changed. Without rationale, a perfectly traceable requirement can become a historical mystery. With rationale, reviewers can judge whether the original constraint still applies to the new configuration or whether new evidence justifies a controlled revision.

Operational review checklist

Before accepting a systems-engineering result, declare the system boundary, identify the requirement or interface concerned, separate evidence from assumptions, check units where quantities are used, and state what verification or configuration decision follows.

Plan for evidence failure as well as nominal success: ask what happens when a sensor record is missing, an assumption changes, an interface is revised or a verification result contradicts the baseline. The response should remain traceable.

- [www.nasa.gov](#) — institutional documentation
- [standards.nasa.gov](#) — institutional documentation
- [www.nasa.gov](#) — institutional documentation
- Space Academy — full pathway

FR ↔ EN parity matrix

EN	FR
Verification coverage	Couverture de vérification
Absolute mass margin	Marge absolue de masse
Margin as a percentage of allocation	Marge relative à l'allocation
Traceability coverage	Couverture de traçabilité
Interface closure	Fermeture des interfaces