Chapter Outline

Exercises and Classroom Discussion

Chapter 7 Health Management and Safety Assessment

7.1 Health Management Fundamentals

7.2 Unified Fault Modeling

7.3 Safety Reinforcement and Fault Data

7.4 Fault Hazards and Safety Assessment

7.5 Diagnosis and Prognostics

7.6 Safety Decisions and Improvement

7.7 Fault Injection Interfaces

7.8 Experiments and Validation

7.9 Exercises and Classroom Discussion

3 Units in This Section

7.9.1 Concepts and Metrics

Textbook: Sec. 7.4, Exercises 1–5

7.9.2 Fault Modeling and Interfaces

Textbook: Sec. 7.4, Exercises 6–8

7.9.3 Safety Plan Design

Textbook: Sec. 7.4, Exercises 9–10

7.9.1 Concepts and Metrics: Health Management

Exercises 1–2: What Is Missing from a Fault Alarm?

Exercise 1: Compare health management and safety assessment. How are they connected?

Exercise 2: Explain prognostics and health management (PHM) and the core functions of the Open System Architecture for Condition-Based Maintenance (OSA-CBM).

Scenario: The system reports "motor anomaly," without its severity, validity period, or mission impact.

Discuss: Can this alarm alone justify a return? Which functions must provide further evidence?

Review: Section 7.1; textbook Sec. 7.1.2.1 (health management) and Sec. 7.1.2.3 (differences and connections).

7.9.1 Concepts and Metrics: Health Management

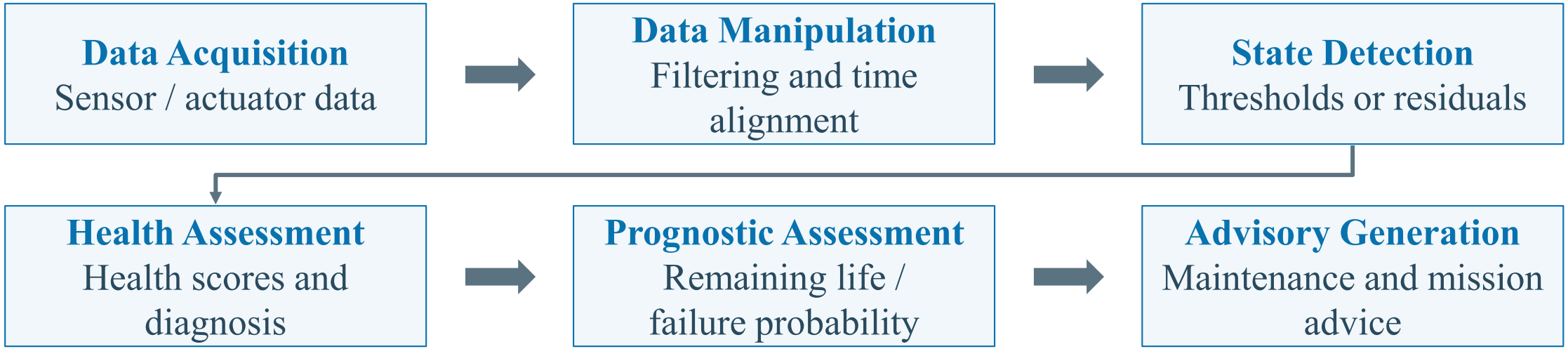
Exercise 1: Compare Health and Safety Using Table 7.1

Dimension	Health Management	Safety Assessment
Focus	Internal operating state	Risks posed to the surroundings
Data sources	Real-time onboard sensor data	Historical operation and external observations
Time scale	Real-time or short-term monitoring	Design and operational phases
Methods	State monitoring, fault diagnosis, and life prediction	Fault tree analysis; failure modes and effects analysis; accident-data analysis
Objective	Improve self-awareness and recovery capability	Reduce accident likelihood and the extent of harm

Health estimates and predictions inform safety assessment, mission decisions, and responses.

7.9.1 Concepts and Metrics: Health Management

Exercise 2: Six Functions Turn Data into Decision Support



Layer 7—Human–machine interaction: display states, assessments, predictions, and advice; receive threshold and maintenance feedback.

The motor-anomaly alarm comes from State Detection. Three later functions develop a mission recommendation.

7.9.1 Concepts and Metrics: Safety Methods and Metrics

Exercises 3 and 5: Choose the Method Before Calculating

Exercise 3: What questions do common safety methods answer? Which suit an inspection UAV?

Exercise 5: Explain mean time between failures (MTBF), Development Assurance Level (DAL), and as low as reasonably practicable (ALARP) in aviation safety.

Claim: "MTBF is 1000 h, so failure cannot occur before hour 999." What is wrong?

Claim: "Diagnosis accuracy is 99%, so a particular DAL is met." Why does this not follow?

Review: Section 7.4—Safety Analysis Methods, Scope of Aviation Classifications, Safety Adaptation for Unmanned Systems, and Mean Time Between Failures; textbook Secs. 7.1.2.2 and 7.1.2.4.

7.9.1 Concepts and Metrics: Safety Methods and Metrics

Exercise 3: Analyze Functions, Components, and Combinations

Method	Starting Point	Inspection Example
Functional hazard assessment (FHA)	Lost or incorrect function	Consequences of losing altitude control
Failure modes and effects analysis (FMEA)	A component's failure mode	Effects of motor degradation on attitude and mission
Fault tree analysis (FTA)	A defined top event	Fault combinations causing an uncontrolled descent

Other methods: FMES summarizes FMEA; SSA integrates system evidence; CCA/CMA examine redundancy independence; ASA/PASA assess aircraft-level safety.

7.9 Exercises and Classroom Discussion

7.9.1 Concepts and Metrics: Safety Methods and Metrics

Exercise 5: MTBF, DAL, and ALARP Address Different Questions

Term	Reference Points	Limitations
MTBF	Mean time between failures = total operating time / number of failures	A statistical average, not a life guarantee
DAL	Levels A–E relate to failure consequences and set development and verification rigor	Not a diagnosis-accuracy grade
ALARP	Reduce risk as low as reasonably practicable and retain supporting evidence	Not a claim of zero risk

MTBF measures reliability; DAL sets assurance rigor; ALARP guides risk reduction.

7.9.1 Concepts and Metrics: Safety Methods and Metrics

Exercise 5: Failure Consequences and Assurance Levels A–E

DAL A	DAL B	DAL C	DAL D	DAL E
Catastrophic $< 10^{-9}$	Hazardous $< 10^{-7}$	Major $< 10^{-5}$	Minor $< 10^{-3}$	No safety effect No probability requirement

Consequences → severity → probability objective per flight hour and assurance level.

DAL sets development and verification rigor; diagnosis accuracy measures an algorithm.

Transport-airplane context: DO-178C software / DO-254 hardware; FAA, EASA, CAAC.

Meaning and scope: Section 7.4, Scope of Aviation Classifications.

7.9.1 Concepts and Metrics: Mission-Based Metrics

Exercise 4: Evaluate Success, Health, and Safety Margin

Exercise 4: Define mission completion probability, mission health, and mission safety margin. Explain their relationship.

Given: Battery energy is sufficient, one motor is near its control limit, and wind is stronger above the destination.

Specify: For each metric, state its object, necessary conditions, and one observable input.

Discuss: Does barely completing one mission justify repeated dispatches?

Review: Section 7.4—Mission Metrics and Safety Margin Example; textbook Sec. 7.1.2.4.

7.9.1 Concepts and Metrics: Mission-Based Metrics

Exercise 4: Report Probability, Score, and Physical Margin Separately

Metric	Operational Definition	Conditions to Report
Mission completion probability	Probability of successful completion in a specified scenario	Success criteria, mission distribution, and statistical interval
Mission health	A measure of how system state matches mission demands	Variables, normalization, weights, and validation evidence
Mission safety margin	Headroom to a chosen capability limit or safety constraint	Constraint, units, model, and uncertainty

Completion can be likely despite low thrust margin. Check pulse-width modulation (PWM) saturation time, wind, battery energy, and degradation before dispatch.

Chapter Outline

Exercises and Classroom Discussion

Chapter 7 Health Management and Safety Assessment

7.1 Health Management Fundamentals

7.2 Unified Fault Modeling

7.3 Safety Reinforcement and Fault Data

7.4 Fault Hazards and Safety Assessment

7.5 Diagnosis and Prognostics

7.6 Safety Decisions and Improvement

7.7 Fault Injection Interfaces

7.8 Experiments and Validation

7.9 Exercises and Classroom Discussion

3 Units in This Section

7.9.1 Concepts and Metrics

Textbook: Sec. 7.4, Exercises 1–5

7.9.2 Fault Modeling and Interfaces

Textbook: Sec. 7.4, Exercises 6–8

7.9.3 Safety Plan Design

Textbook: Sec. 7.4, Exercises 9–10

7.9.2 Fault Modeling and Interfaces: IMU Fault Models

Exercise 6: Make IMU Faults Injectable and Diagnosable

Exercise 6: Design an inertial measurement unit (IMU) fault model and identify features suitable for diagnosis.

Model: Relate ground truth, fault terms, and measurements. State units, coordinate frames, and timing.

Diagnose: Select and justify at least one feature for each of bias, noise, freezing, and delay.

Validate: Compare healthy and faulty segments; include non-fault controls such as maneuvers and vibration.

Review: Section 7.2; textbook Secs. 7.1.3.3–7.1.3.4 (fault modeling methods and approaches).

7.9.2 Fault Modeling and Interfaces: IMU Fault Models

Exercise 6: Distinguish Bias, Noise, Freezing, and Delay

$$\tilde{a}(t) = a(t-\tau) + b(t) + k_n n(t)$$

$\tilde{a}, a, n$: measurement, truth, noise; τ, b, k_n: delay, bias, noise gain.
Acceleration: $\mathrm{m/s^2}$. Freezing holds the output constant (Sec. 7.2, Component Fault Models).

Fault	Features to Compare	Confounders to Exclude
Bias / noise	Mean or residual / variance after mean removal	Attitude, vibration, and operating-condition changes
Freezing	Constant output; near-zero rate of change and variance	Actual rest and quantization limits
Delay	Dynamic-signal cross-correlation or phase lag	Clock mismatch and queued messages

7.9.2 Fault Modeling and Interfaces: Automated Data Collection

Exercise 7: Turn Automated Injection into Traceable Data

Exercise 7: Outline automated fault injection and data collection in RflySim.

Include: Case configuration, versions, preparation, baseline, trigger, acquisition, termination, recovery, and evaluation.

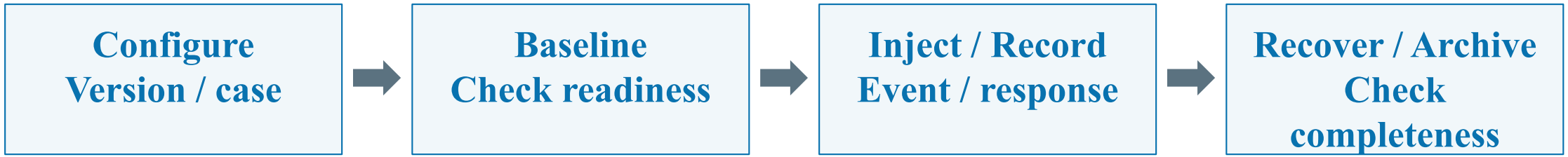
Timing: Do command time, model-effect time, and log time use the same clock?

Exceptions: How should a run be labeled if takeoff fails, logs are missing, or the fault effect is unconfirmed?

Review: Sections 7.3 and 7.8.3; textbook Sec. 7.1.5.2. Examples e9_data_collect and e18_FaultLogTimestamp: [installation directory]/RflySimAPIs/7.RflySimPHM/0.ApiExps/

7.9.2 Fault Modeling and Interfaces: Automated Data Collection

Exercise 7: Record the Start, Events, End, and Recovery



Save: Case ID, random seed, versions, units, timestamps on a common time base, commands, and raw logs.

Stop: Use time, mission, or protection thresholds. Retain timeouts and failures from valid runs in the evaluation.

Check: Evidence that the fault took effect, healthy controls, and the recovered state. Map different clocks before comparing times.

7.9.2 Fault Modeling and Interfaces: Accelerometer Noise

Exercise 8: Set Accelerometer Noise Gain to 0.5 in SITL

Exercise 8: In RflySim software-in-the-loop (SITL; SIL in the textbook), set accelerometer noise gain to 0.5 and explain the interface protocol.

Directory: [installation directory]/RflySimAPIs/7.RflySimPHM/1.BasicExps/e4_SensorFault/
Read Readme.pdf first.

Check: Distinguish interface parameter p (called noise gain in Readme) from noise-branch multiplier k. Verify the model version, fault ID, and slot.

Submit: Message fields, trigger and recovery steps, and three-axis acceleration comparisons for healthy, injected, and recovered segments.

Review: Sections 7.3.1 and 7.7; textbook Secs. 7.1.5.1 and 7.2.1 (fault modeling and injection framework).

7.9.2 Fault Modeling and Interfaces: Accelerometer Noise

Exercise 8: Select a Fault ID and Set Its Parameter Slot

```
silInt = [123542] + [0] * 7
silFloat = [0.5, 0.0] + [0.0] * 18
mav1.sendSILIntFloat(silInt, silFloat)
```

Layer	This Example
Message	Check value 1234567897; vehicle ID; 8 integers; 20 float32 values
Parameter slots	Integer slot 0: 123542 (accelerometer noise); floating-point slot 0: p = 0.5
Model channel	UDP port 30100 + 2(ID − 1); direct input to the model fault module

7.9.2 Fault Modeling and Interfaces: Accelerometer Noise

Exercise 8: Separate Input p from Noise Multiplier k

$$k = 0.2 + 0.8p = 0.2 + 0.8 \times 0.5 = 0.6$$

Symbols: k is the noise-branch multiplier; p is interface parameter in floating-point slot 0 (noise gain in Readme).

Healthy branch: The noise input is multiplied by 0.2. Record three-axis baselines under matched operating conditions.

Course convention: Both the textbook and Readme call p the noise gain. With p = 0.5, k = 0.6.

Alternative interpretation: To obtain k = 0.5, set p = (0.5 − 0.2) / 0.8 = 0.375.

Recovery: For this model, send parameter 0 with the same ID to restore amplitude. Reinitialize the model before a new independent case.

7.9.2 Fault Modeling and Interfaces: Accelerometer Noise

Exercise 8: Prepare, Run, and End e4_SensorFault

Directory: [installation directory]/RflySimAPIs/7.RflySimPHM/1.BasicExps/e4_SensorFault/
Start with Readme.pdf and FaultInjectAPITest.py.

Prepare: In MATLAB 2022B or later, run Init.m → Build MulticopterModel.slx → right-click and run GenerateModelDLLFile.p to generate the DLL.

Inject: Run MulticopterModelSITL.bat as administrator → open Python38Run.bat → set silFloat[0:2] = [0.5,0] → python FaultInjectAPITest.py.

End: After about 50 s, the script prints Sim End and exits. Press any key in the SITL command window to close CopterSim, QGC, and RflySim3D.

Note: The SITL script overwrites the same-named DLL in CopterSim's model directory and restarts any running CopterSim.

7.9.2 Fault Modeling and Interfaces: Accelerometer Noise

Exercise 8: Confirm the Event, Then Compare the Logs

5 s Arm	→ **20 s** Inject p = 0.5	→ **20–50 s** Observe hover response	→ **About 50 s** Print Sim End

Observation: With parameter 30, the Readme example loses stable attitude and position hold. At p = 0.5, k = 0.6; a weaker effect is expected. Use log comparisons to judge it.

Logs: Convert ulg files under [installation directory]/Firmware/build/px4_sitl_default/instance_1/log/ to csv. Compare accelerometer_m_s2[0]–[2] in _sensor_combined_0.csv before and after injection.
If the fault module fails to compile, replace it with the matching library version. MATLAB 2022B or later is required to generate the DLL.

Chapter Outline

Exercises and Classroom Discussion

Chapter 7 Health Management and Safety Assessment

7.1 Health Management Fundamentals

7.2 Unified Fault Modeling

7.3 Safety Reinforcement and Fault Data

7.4 Fault Hazards and Safety Assessment

7.5 Diagnosis and Prognostics

7.6 Safety Decisions and Improvement

7.7 Fault Injection Interfaces

7.8 Experiments and Validation

7.9 Exercises and Classroom Discussion

3 Units in This Section

7.9.1 Concepts and Metrics

Textbook: Sec. 7.4, Exercises 1–5

7.9.2 Fault Modeling and Interfaces

Textbook: Sec. 7.4, Exercises 6–8

7.9.3 Safety Plan Design

Textbook: Sec. 7.4, Exercises 9–10

7.9.3 Safety Plan Design: Hierarchical Decisions

Exercise 9: Divide Work Between a Behavior Tree and RL

Exercise 9: Design a hierarchical safety architecture combining a behavior tree with reinforcement learning (RL). Explain their coordination.

Conditions: Diagnosis may raise false alarms, observations may be stale, and a learned policy may propose infeasible actions.

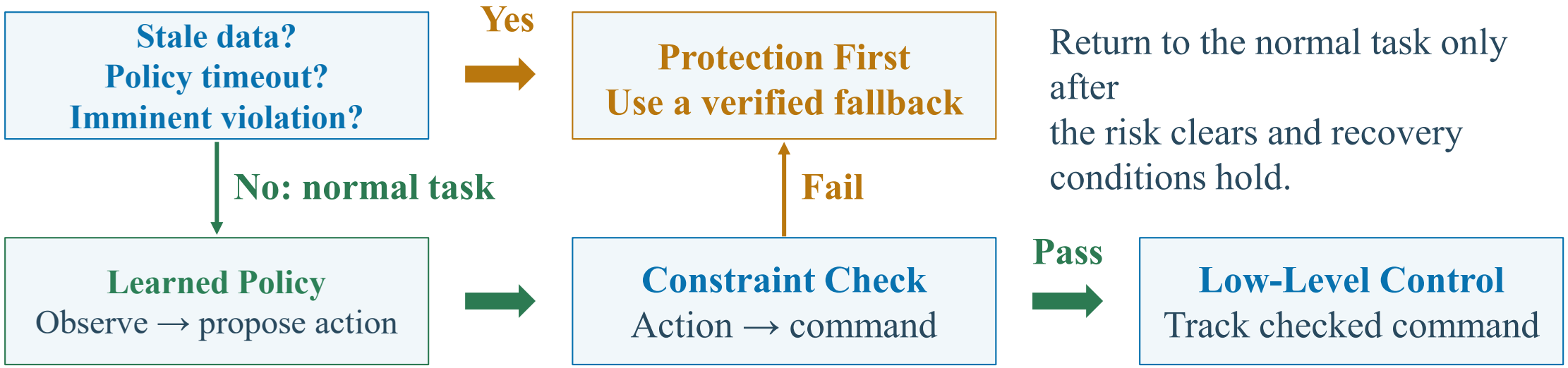
Specify: Priorities and inputs/outputs for normal tasks, constrained policies, and protective actions.

Validate: Report constraint violations, switching delay, and false-trigger costs as well as mission completion rate.

Review: Section 7.6; textbook Sec. 7.1.5.5 (training safety decision algorithms).

7.9.3 Safety Plan Design: Hierarchical Decisions

Exercise 9: Constraints Determine When a Policy May Act



The behavior tree sets priorities and switches branches. Hysteresis, dwell time, and recovery conditions prevent chattering.

Report violations, switching delay, and false-trigger costs alongside completion rate.

7.9.3 Safety Plan Design: Combined-Disturbance Tests

Exercise 10: Separate Wind and Payload-Offset Effects

Exercise 10: Plan an automated test of a gust combined with a payload offset. Define key metrics.

Variables: Wind strength, direction, and duration; payload mass, offset direction, and distance.

Controls: Why retain four groups—neither factor, wind only, offset only, and both?

Evaluate: Compare tracking, control reserve, mission outcomes, and potential consequences for the surroundings.

Review: Sections 7.2 and 7.3; textbook Sec. 7.1.3.2 (environmental and structural faults) and Sec. 7.1.5.2.

7.9.3 Safety Plan Design: Combined-Disturbance Tests

Exercise 10: Test Main Effects, Then Their Combination

Group	Wind	Payload Offset	Purpose
A: Baseline	Absent	Absent	Normal error and control reserve
B: Wind	Present	Absent	Effect of wind alone
C: Offset	Absent	Present	Effect of center-of-gravity shift
D: Combined	Present	Present	Interaction and worst cases

Report together: Tracking error, saturation duration, constraint violations, completion rate, and response time.
Repeat random-condition tests and report statistical intervals.

Chapter Summary: Common Mistakes

Review Similar Concepts and the Order of Operations

7.1: Symptom ≠ cause; current health ≠ mission safety; one simulation layer ≠ complete evidence.

7.2: Command ≠ rotor speed ≠ thrust coefficient; bias, noise, and timing have different effects.

7.3: Cases ≠ completed runs; command time ≠ effect time. Split flights before creating windows.

7.4: Score ≠ accident probability; MTBF is an average, not a device lifetime.

7.5: Exclude labels from inputs; predict RUL from history only. Censoring ≠ failure.

7.6 / 7.8: Mode accepted ≠ motion controlled; model credibility ≠ safety probability.

7.7: Send ≠ effect; stop ≠ reset. Zero IDs do not clear latched model faults.

Chapter Summary

Health Management Informs Safety Decisions and Actions

Concepts: Health management examines internal state and future capability; safety assessment examines external risk. OSA-CBM's six PHM functions turn data into advice.

Methods: Fault models and data → risk assessment (FHA, FMEA, FTA, and DAL) → diagnosis and prognosis → safety decisions and system regression tests → injection interfaces and simulation-to-real validation.

Scope: Simulation and offline results apply to stated models, versions, and conditions. MTBF, DAL, and ALARP address different questions and are not interchangeable.

Watch for: Anomaly versus fault; health versus safety; accuracy versus a reliable closed loop; input gain p versus branch multiplier k; time-threshold labels versus fault ground truth.

Next—Chapter 8: Multimodal Perception and Intelligent Decision-Making brings more reliable observations into safety decisions.