Drone Link Assignment – Behavioral Extensions

Extend the Drone Link system to support flexible, runtime-adaptable behaviors and interaction control among drone modules and subsystems.

New Functional Requirements (Behavioral Patterns)

Dynamic Flight Behavior Switching

Different drones may require different flight behaviors such as aggressive maneuvering, energy-saving cruising, or stealth navigation. Implement a mechanism that allows selecting and switching between different flight behavior algorithms (that implement FlightBehavior) at runtime based on mission type or environmental feedback.

Module Status Broadcasting

Introduce a centralized event system where modules (e.g., battery, sensors, flight controller) can register to be notified when specific system states change (e.g., low battery, GPS lock, mission abort). Ensure new modules can be added or removed without affecting existing logic. Each module will implement ModuleInterface interface.

Mission State Management

Model the drone mission lifecycle using a state machine. The drone can be in states such as "Idle", "PreFlightCheck", "InMission", "ReturningHome", or "EmergencyLanding". Transitions between states should affect available actions and module behavior accordingly. Each state implement the MissionState interface that receives the drone context (a class that you should define)

Fault Handling Workflow

Introduce a modular error-handling chain where each handler (e.g., battery alert, obstacle detection, signal loss) processes mission-critical faults. If one handler cannot fully resolve the issue, it should pass it to the next in the chain for further evaluation.

Mission Snapshot and Restore

Enable the system to capture and store the drone's mission state (e.g., location, module status, configuration) at regular intervals or specific checkpoints. Implement functionality to roll back the drone to a previous state in case of failure or mission abortion.

Pre-Configured Mission Templates

Define abstract mission structures with invariant steps such as initialization, validation, execution, and shutdown. Allow subclasses or concrete missions (e.g., surveillance, delivery, terrain mapping) to implement mission-specific behaviors by overriding certain steps.

Command Queue and Replay

Implement a command queue for drone operations (e.g., takeoff, scan area, drop payload). Support queuing, undoing, and replaying commands. Enable mission controllers to review

and simulate previous command sequences for testing or training purposes. The mechanism will be designed based on the Command class.