Design Document - SimJoby

Ray Torres – April 2025

Purpose

The program simulates the operation of eVTOL aircraft manufactured by five companies, each with different operational characteristics. The program records data such as flight time, faults, distance travelled, passenger miles, and charging statistics. Full requirements are given at the end of this document.

Overview

I implemented significant "Extra Credit" beyond the minimum problem statement:

(1) Users can configure the number of aircraft, duration of the simulation, number of charging stations, and duration of time-step.

(2) I implemented the simulation in two parts, a *Basic Simulation* and a *Full Simulation*.

- The *Basic Simulation* calculates the statistics per vehicle *type* as specified in the original Problem Statement. This simulation assumes that a charger is immediately available as soon as the battery is depleted.
- The *Full Simulation* provides a full simulation framework by iterating over time steps and simulating flights, charger usage and availability. This tracks the time duration an aircraft is waiting for a charger and the charging time. The simulation can be run with a variable number of aircraft and charging stations and the total time waiting to charge can be determined.
  This allows evaluation of the optimum number of chargers for any given fleet size. Statistics are given per each aircraft, not aircraft type.
  This framework allows for the addition of more advanced and comprehensive simulation capabilities.

Key Features

- **Simulation Initialization**: Users specify the number of aircraft, simulation duration and number of chargers.
- **Vehicle Behavior**: Each vehicle simulates flight, monitors battery depletion, and charges as needed.
- **Statistical Tracking**: Flight statistics are calculated and displayed.
- **Singleton Design**: The Simulation class is implemented as a singleton to ensure global simulation control.
- **Document-View design pattern**, separates data management, program processing, and user interface.

- **Object-Oriented Programming**: Classes encapsulate data and methods for modularity and reusability. The EVTOL class can be extended to add new vehicle types with unique characteristics without modifying the core simulation logic.
- **Robust User Input**: Validates user input and provides default values if invalid.
- **Scalability**: Fleet generation is dynamic, allowing simulations with any number of vehicles.

System Architecture

State Machine

Each aircraft maintains its current state, implemented as an enumeration.
The aircraft sequences through the states:
`FLYING -> WAITING_TO_CHARGE -> CHARGING -> FLYING.`
(There is also an `IDLE` state that could be used in the future for aircraft maintenance, passenger loading /unloading, etc.)

Classes

The application uses a modular architecture divided into the following key components:

---

1. EVTOL Class

This class represents an individual eVTOL vehicle and encapsulates the following properties and behaviors:

- **Attributes:**
  - Manufacturer name
  - Cruise speed, battery capacity, charging time, and energy usage
  - Passenger capacity and fault probability
  - Runtime statistics (distance traveled, faults, passenger miles.)
- **Key Methods:**
  - SimulateFlying: Models the vehicle in flight, updating distance, battery level, and fault occurrences.
  - SimulateCharging: Models the charging process, updating the battery level and charging time.
  - SimulateWaitingToCharge: Accumulates time spent waiting for a charger.

2. SIMULATION Class (Singleton)

Manages the entire simulation process, from initializing parameters to running and displaying results. Model eVTOL behavior, including energy consumption, charging, and fault probabilities.

- **Responsibilities:**
    - Create and manage the fleet of vehicles.
    - Allocate chargers and track their availability.
    - Simulate vehicle behavior over the specified time duration.
    - Compute and display statistics.

- **Key Methods:**
    - Initialize -Sets up the simulation.
    - run: Executes the simulation by advancing time steps and managing each vehicle's state.
    - displayResults: Presents the simulation results.
    - createRandomFleet: Generates a fleet with random distributions of vehicle types.

---

3. VIEW Class (Singleton)

Handles all user interactions, following Document-View design pattern.

- **Responsibilities:**
    - Display welcome and goodbye messages.
    - Collect user input for simulation parameters (duration, number of vehicles, number of chargers.).
    - Ask if the user wants to rerun the simulation.

---

4. TestSimJoby Class

A preliminary framework for unit testing.

---

5. DOCUMENT Class (Singleton)

Placeholder for future enhancements, such as logging results to files or maintaining user simulation history. Currently, this class has not been implemented.

Workflow

1. **User Input:**
   - The View class collects simulation parameters (e.g., duration, number of vehicles, chargers) from the user.
2. **Simulation Initialization:**
   - The Simulation class creates a fleet of vehicles and sets up the required parameters.
3. **Simulation Execution:**
   - The Simulation class iterates through time steps.
   - Each EVTOL object updates its state and statistics based on available chargers and energy consumption.
4. **Display Results**
   - At the end of the simulation, results are aggregated and displayed, providing insights per manufacturer.

Future Enhancements

1. **Logging and Reporting**: Use the Document class to save simulation results to a file for future reference.
2. **Multithreading**: Implement each aircraft flying in its own thread. This would allow simulation of more complex behavior and interactions.
3. **Class inheritance**: Each manufacturer's vehicle could be implemented in its own class derived from EVTOL. Given the current requirements, inheritance does not offer significant advantages because each vehicle type can be specified by initialization parameters. If the vehicle types had different functionality, the EVTOL class could provide virtual functions that could be overridden by the base classes to provide manufacturer-specific functionality.

Problem Statement

There are five companies developing eVTOL aircraft. The vehicle produced by each manufacturer has different characteristics. Six distinct properties are laid out in the below table:

Company Name	Alpha Company	Bravo Company	Charlie Company	Delta Company	Echo Company
Cruise Speed (mph)	120	100	160	90	30
Battery Capacity (kWh)	320	100	220	120	150
Time to Charge (hours)	0.6	0.2	0.8	0.62	0.3
Energy use at Cruise (kWh/mile)	1.6	1.5	2.2	0.8	5.8
Passenger Count	4	5	3	2	2
Probability of fault per hour	0.25	0.10	.05	.22	.61

You will simulate using these vehicles for 3 hours. Of course, your simulation should take much less time than that. 20 total vehicles should be deployed, and a random number of each type of vehicle should be used (with the total between all five types being 20).

There are only three chargers available for all 20 vehicles! A single charger can only be used by one vehicle at a time. Assume the chargers can charge the batteries in the *Time to Charge* time listed for each vehicle.

Keep track of the following statistics *per vehicle type*:
- average flight time per flight
- average distance traveled per flight
- average time charging per charge session
- total number of faults
- total number of passenger miles.

For example, if there are 2 vehicles carrying 4 passengers on a vehicle that cruises for 1 hour at 100 mph, total number of passenger miles is 2 * 4 * 1 * 100 = 800.

Assume that:
- Each vehicle starts the simulation with a fully charged battery
- Each vehicle instantaneously reaches Cruise Speed
- Each vehicle is airborne for the full use of the battery and is immediately in line for the charger after running out of battery power.