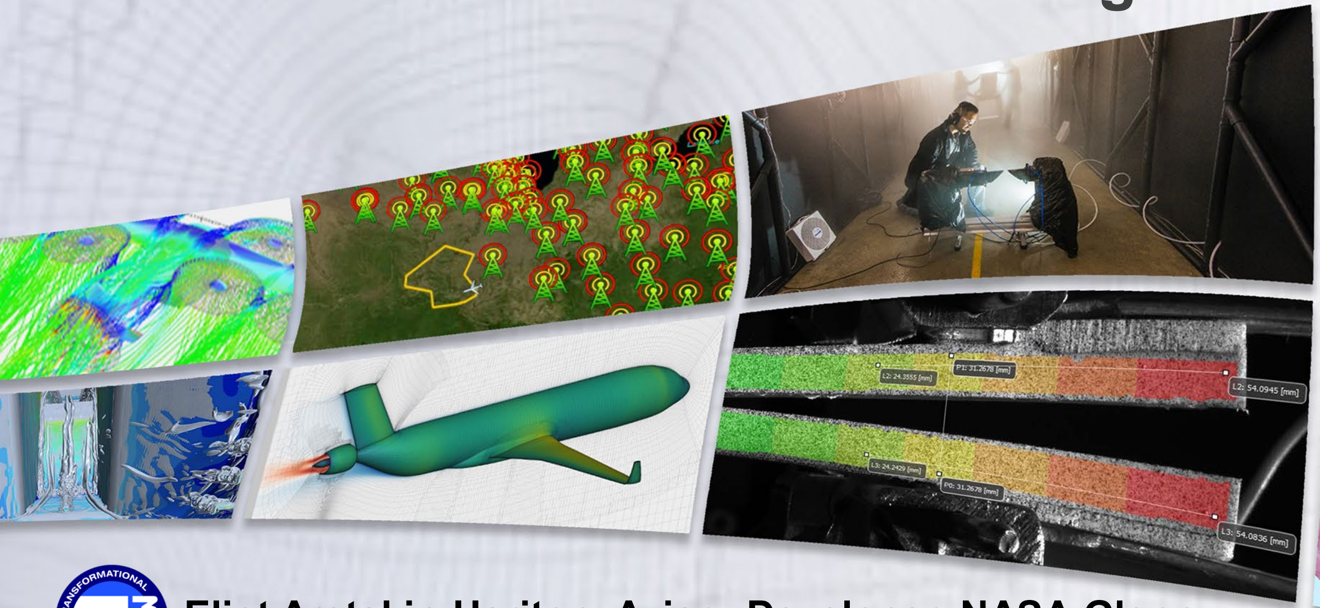
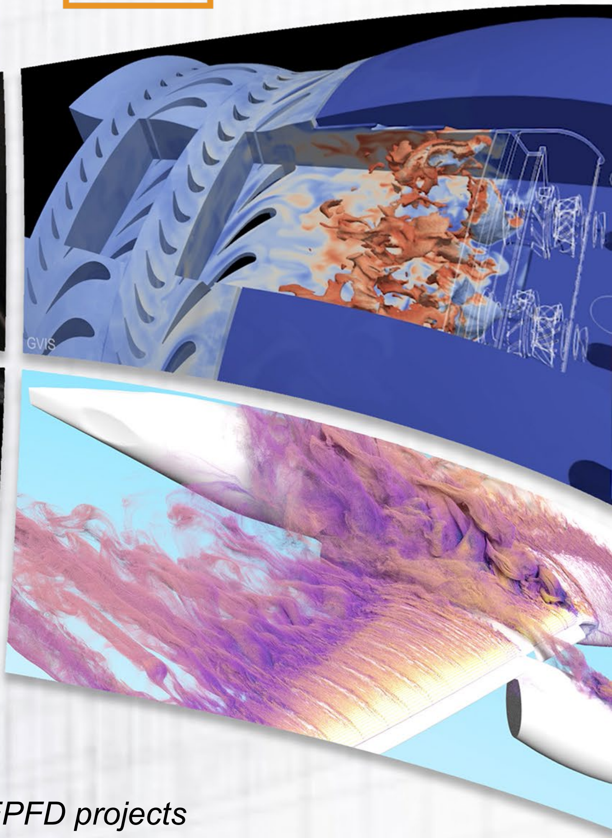
National Aeronautics and Space Administration

NASA's Aviary Takes Flight:

A Public Software for Aircraft Design





Eliot Aretskin-Hariton, Aviary Developer, NASA Glenn

Jason Kirk, Aviary Lead Developer, NASA Langley

www.nasa.gov

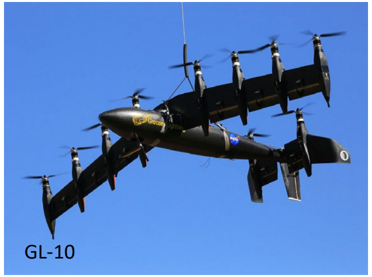
Aviary activities have been co-funded by the T³, AATT, and EPFD projects

Presentation Roadmap

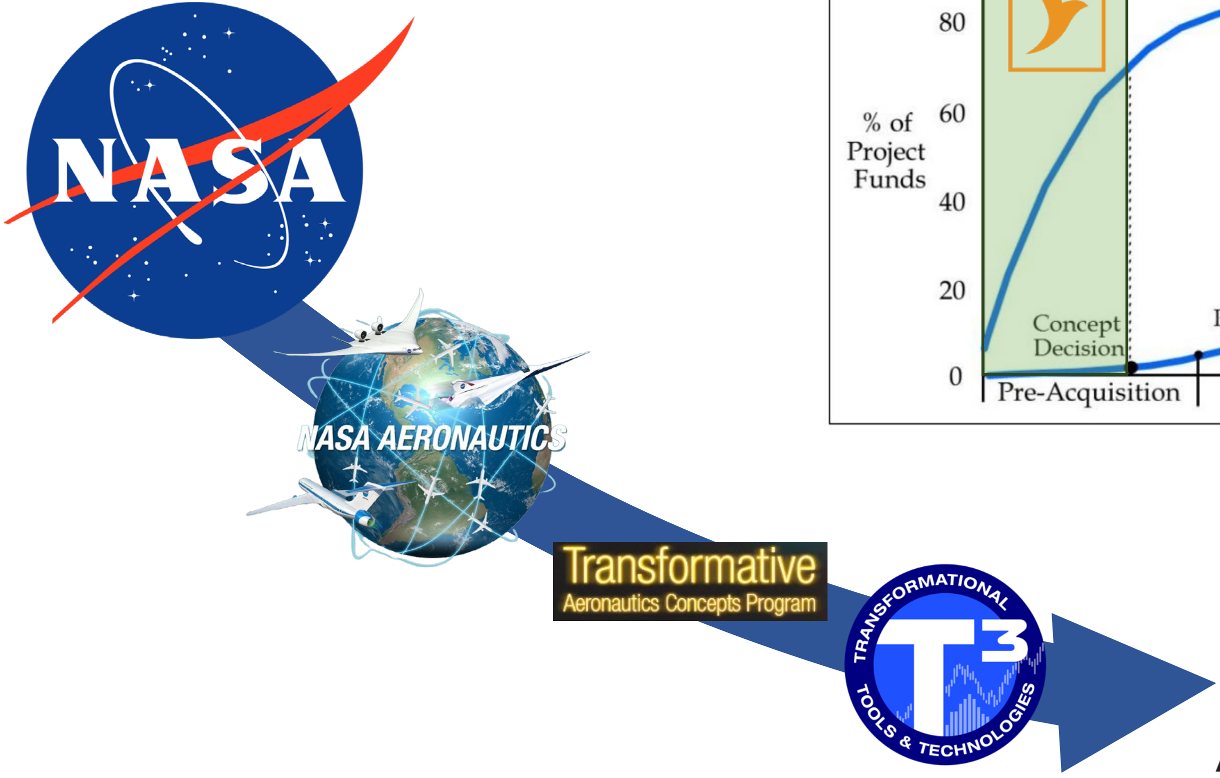
- NASA Aeronautics Goals
- Aviary meets these needs
- How Aviary works
  - The structure of Aviary
  - The user’s interaction with Aviary
- How Aviary has been used at NASA
  - Partnership
  - Future work
- How YOU can get involved

NASA Aeronautics Goals

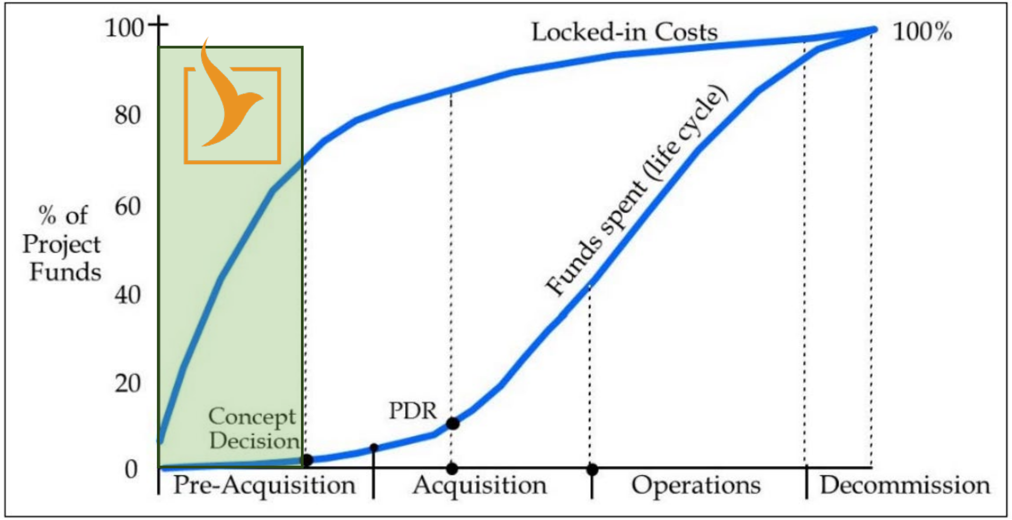
“NASA's Aeronautics programs focus on research, development, and testing of aviation technology advancements that will benefit humankind and retain U.S. leadership in a vital manufacturing and transportation sector.” - Robert Pearce, NASA Associate Administrator



Where does Aviary fit in?



AVIARY



Aviary's Areas of Transformation

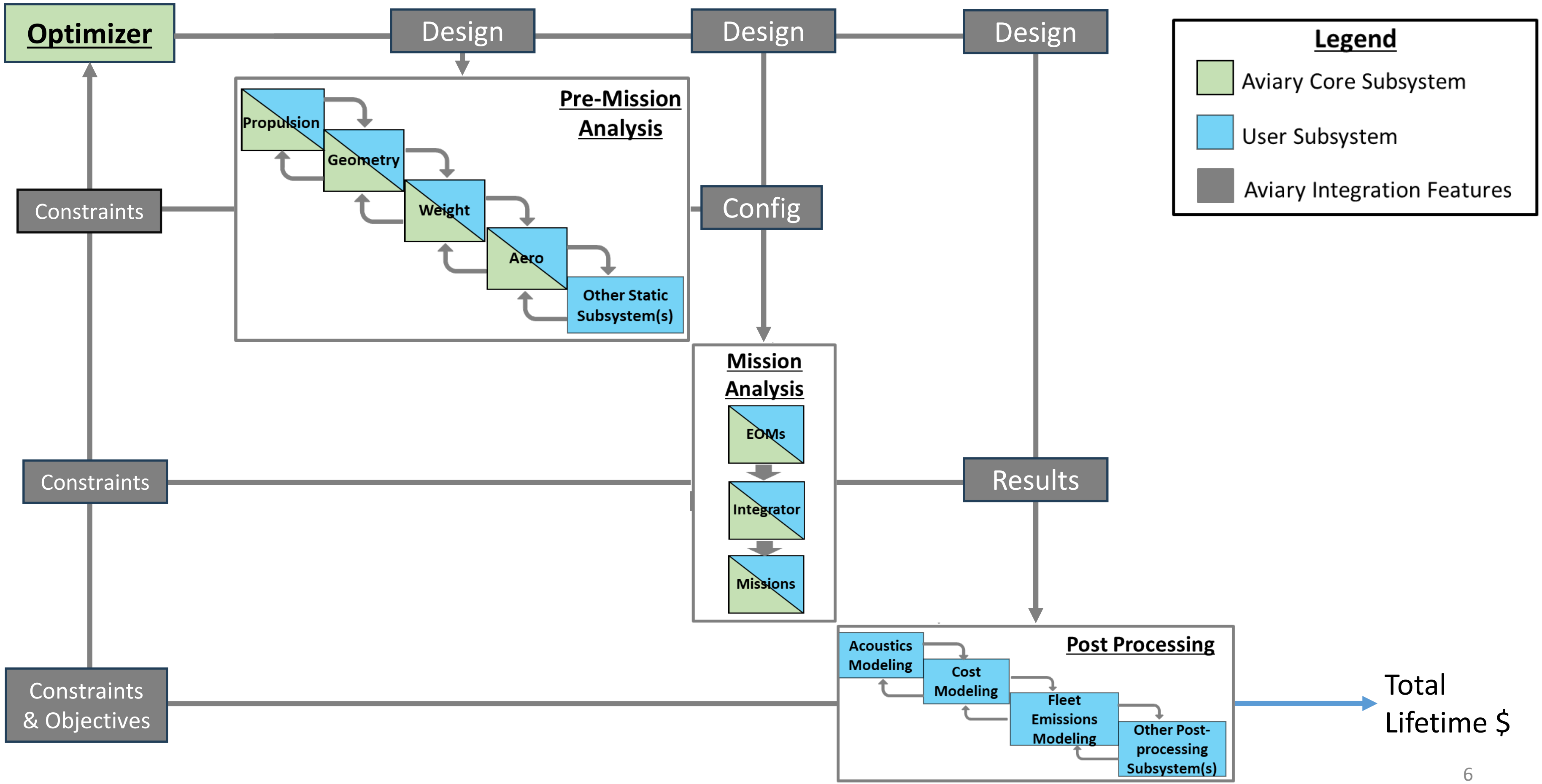
Revolutionary

- Enables coupling of disciplines and trajectories previously unrealizable
- Consolidation and modernization of multiple closed-source legacy aircraft design tools
- Allows specialists to specialize - "Bring Your Own Subsystem, we have an example aircraft"
- Optimize complex aircraft designs for high-level objectives

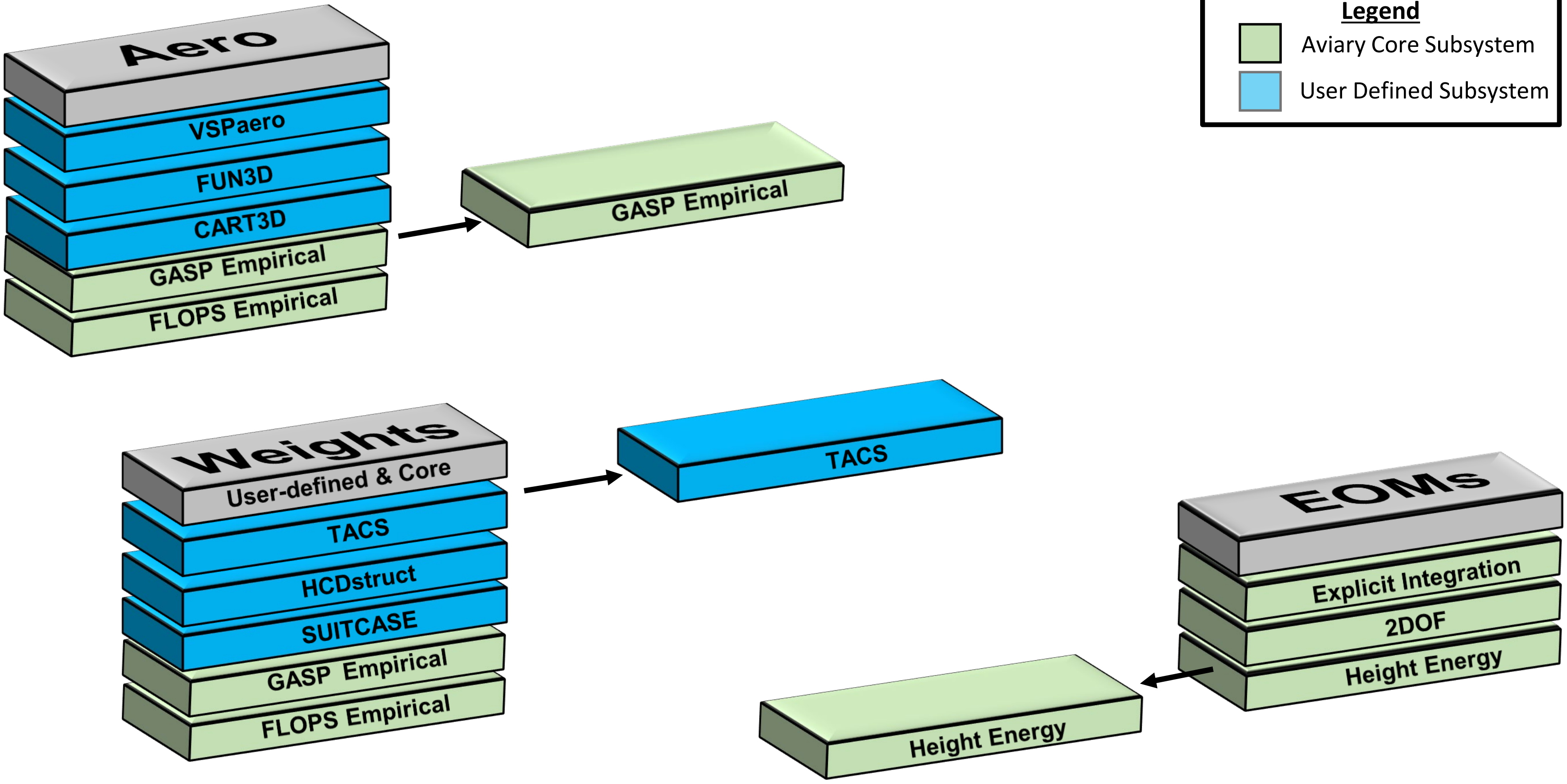
Inspires innovation

- Open source, free model and result sharing
- Documentation/ Examples/ Video Tutorials
- Introduces Students to complex system modelling
- Connect new libraries (UQPCE)
- Bridges Academia and Industry
- Pathway to High-fidelity

The Details: Structure of Aviary

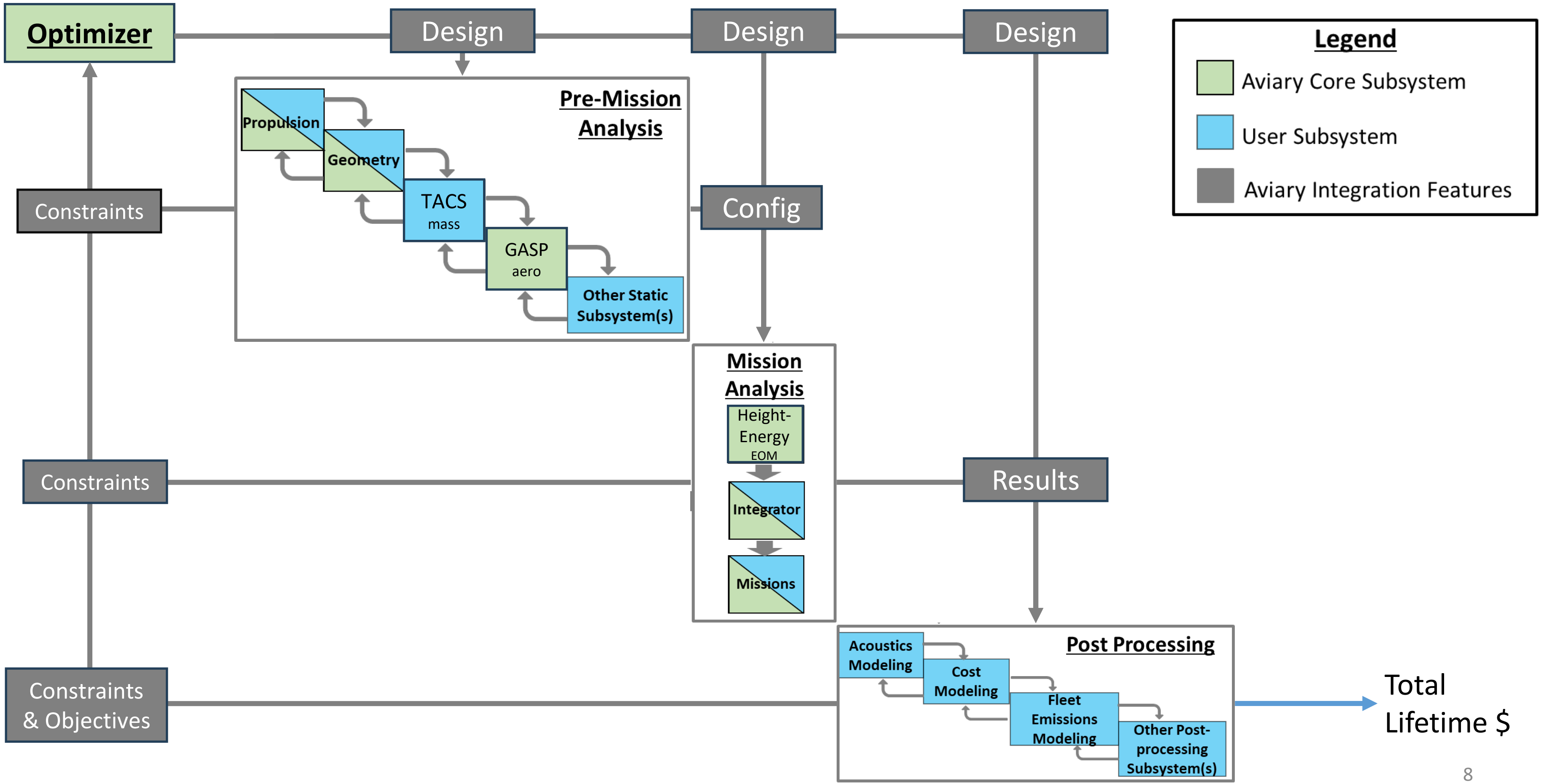


The Details: User Selecting Disciplines



Legend
- Aviary Core Subsystem
- User Defined Subsystem

The Details: Structure of Aviary



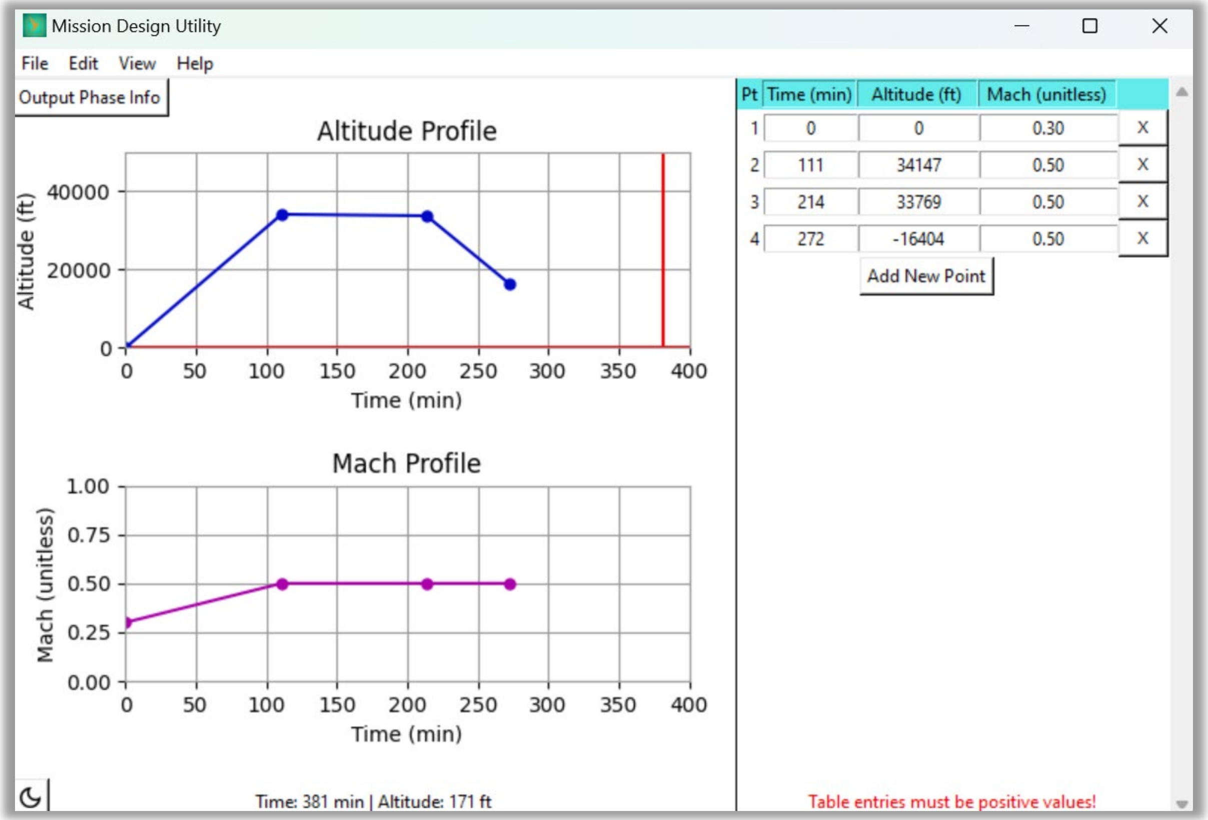
User Interaction with Aviary

Aviary's user interface is designed as a series of "layers" that build on each other

- Each additional layer becomes more complex and provides more capability & flexibility
- Accommodates users of all experience levels

Level 1 (input file)	Little to no coding required; provide input file and execute
Level 2 (Python script with helper functions)	Swap subsystems or make small changes to analysis setup
Level 3 (fully scripted, custom analysis)	Custom Python scripts to set up and optimize designs; introduce complex components or subsystems

Level 1: Draw a Mission



- Used to generate input files
- Allows for visualization of the mission profile
- Validation to ensure that the trajectory is physically valid
- Processing of the points to reduce the risk of numerical problems

Level 1: Select Your Aircraft .csv

```
# Input Values
aircraft:crew_and_payload:cargo_mass,10040,lbm
aircraft:crew_and_payload:num_passengers,180,unitless
aircraft:crew_and_payload:passenger_mass_with_bags,200,lbm
aircraft:engine:data_file,models/engines/turbofan_23k_1.deck,unitless
aircraft:engine:reference_sls_thrust,28690,lbf
aircraft:furnishings:mass,11192,lbm
aircraft:fuselage:aisle_width,24,inch
aircraft:fuselage:num_seats_abreast,6,unitless
aircraft:fuselage:pilot_compartment_length,9.5,ft
aircraft:fuselage:seat_pitch,29,inch
aircraft:fuselage:seat_width,20.2,inch
aircraft:fuselage:tail_fineness,3,unitless
aircraft:fuselage:wetted_area_factor,4000,unitless
aircraft:horizontal_tail:area,0,ft**2
aircraft:horizontal_tail:aspect_ratio,4.75,unitless
aircraft:horizontal_tail:form_factor,1.25,unitless
aircraft:horizontal_tail:mass_coefficient,0.232,unitless
```

Included Models

- Commercial Single-Aisle, 180pax, conventional
- Single-Aisle, 96 pax, conventional
- N3CC: single-aisle, 165 Pax, 2035 technology
- Turboprop freighter: 31k lb payload, turboprop, all-electric variant in progress
- Blended Wing Body (BWB), in progress
- Community Models*

Understanding Your Results

Aviary Dashboard for aircraft_for_bench_FwFm

Model | Optimization | Results | Subsystems

Mission Summary | Trajectory Results Report | Aviary Variables | Timeseries Mission Output Report | Aircraft 3d model

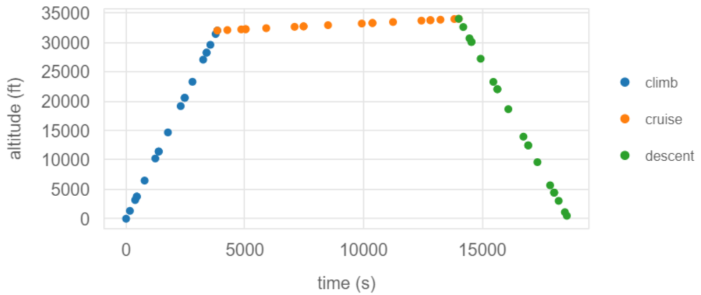
This is one of the most important reports produced by Aviary. It will help you visualize and understand the optimal trajectory produced by Aviary. Users should play with it and try to grasp all possible features. This r
timeseries tab, users can select which phases to view. Other features include hovering the mouse over the solution points to see solution value and zooming into a particular region for details, etc.

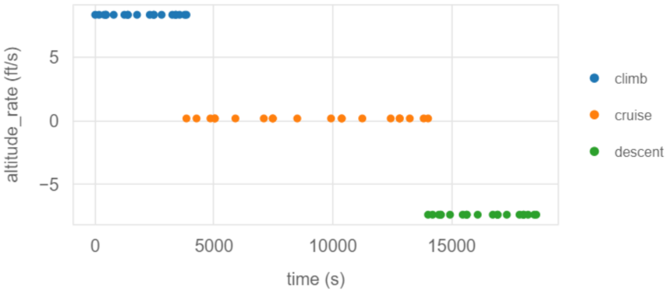
Results of aircraft_for_bench_FwFm
Creation Date: 2024-07-02 11:22:35

Timeseries | climb parameters | cruise parameters | descent parameters

Display data: Solution | Simulation

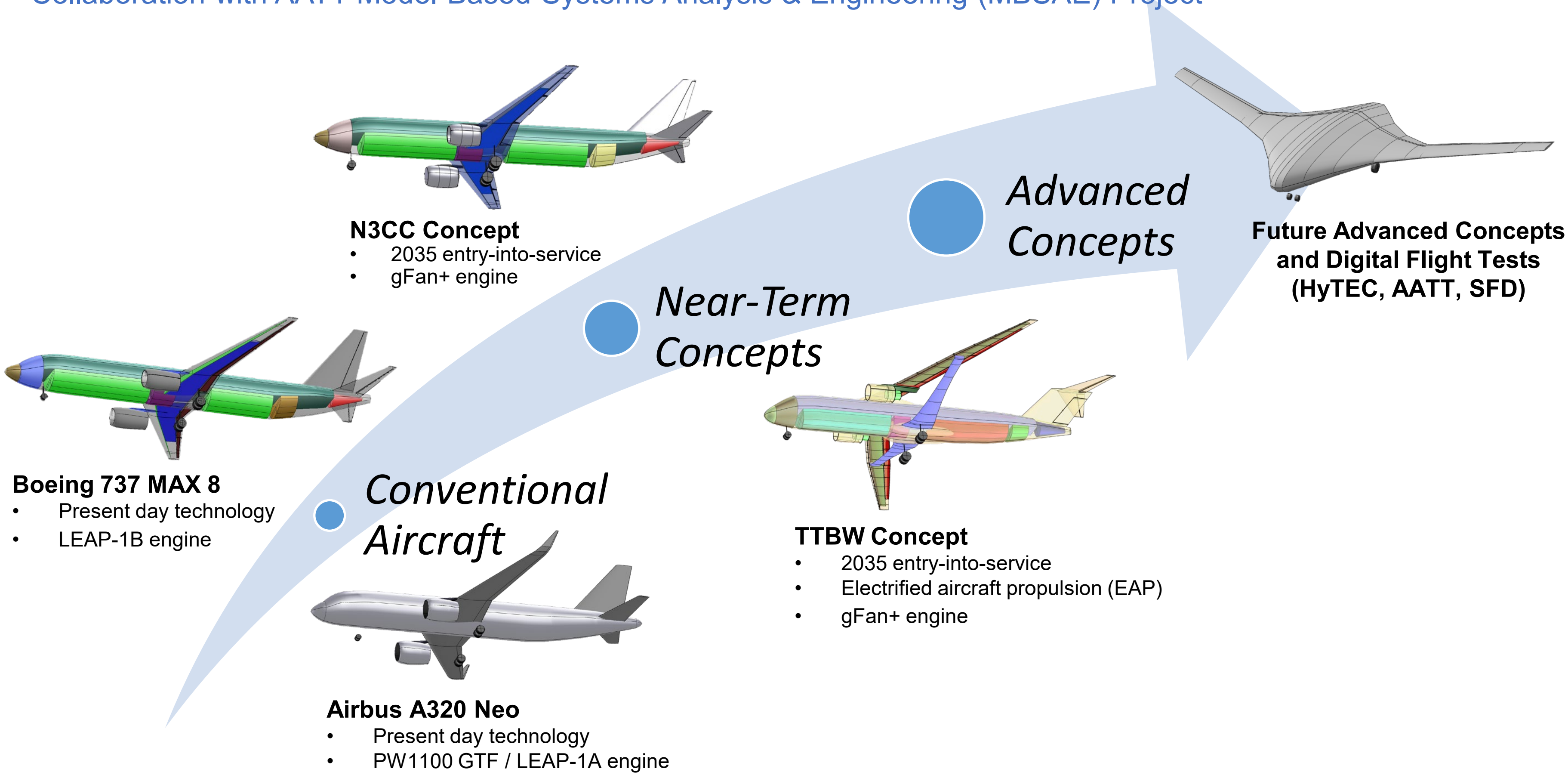
Plot phases: climb ✕ | cruise ✕ | descent ✕





Applications Within NASA

Collaboration with AATT Model-Based Systems Analysis & Engineering (MBSAE) Project



Partners and External Users

Industry

BOEING

NORTHROP GRUMMAN

ELECTRA

GE Aerospace

NRA Funded Research

Potential FLOPS Replacement

Academia

Georgia Institute of Technology

UNIVERSITY OF MICHIGAN

UC DAVIS UNIVERSITY OF CALIFORNIA

VIRGINIA TECH

WICHITA STATE UNIVERSITY

Using Aviary for courses

Future Areas of Work

- Blended Wing Body Support
- All-electric and hybrid-electric example aircraft
- 6DOF Mission EOMs
- National Airspace Simulation Digital Twin Integration
- Examples Integration with Uncertainty Quantification Tools (UQPCE)
- Improving ease of Aviary’s ability for Design of Experiments

Summary

- NASA Aeronautics Goals
- Aviary meets these needs
- How Aviary works
  - The structure of Aviary
  - The user’s interaction with Aviary
- How Aviary has been used at NASA
  - Partnership
  - Future work
- How YOU can get involved

AVIARY

Try Out Aviary Today!

AVIARY

SCAN HERE

Email agency-aviary@mail.nasa.gov to connect with the Aviary team

Aviary can also be installed through GitHub:
https://github.com/OpenMDAO/Aviary
Or the Python package manager:
“pip install aviary”

The Aviary Team

Thank you to all our current and former members!

Current Members

- Jason Kirk (LaRC)
- Eliot Aretskin-Hariton (GRC)
- Ken Moore (GRC)
- Xun Jiang (LaRC)
- Herb Schilling (GRC)
- Chris Bennett (LaRC)
- Kaushik Ponnapalli (GRC)
- Nathan Perreau (LaRC)

Past Members

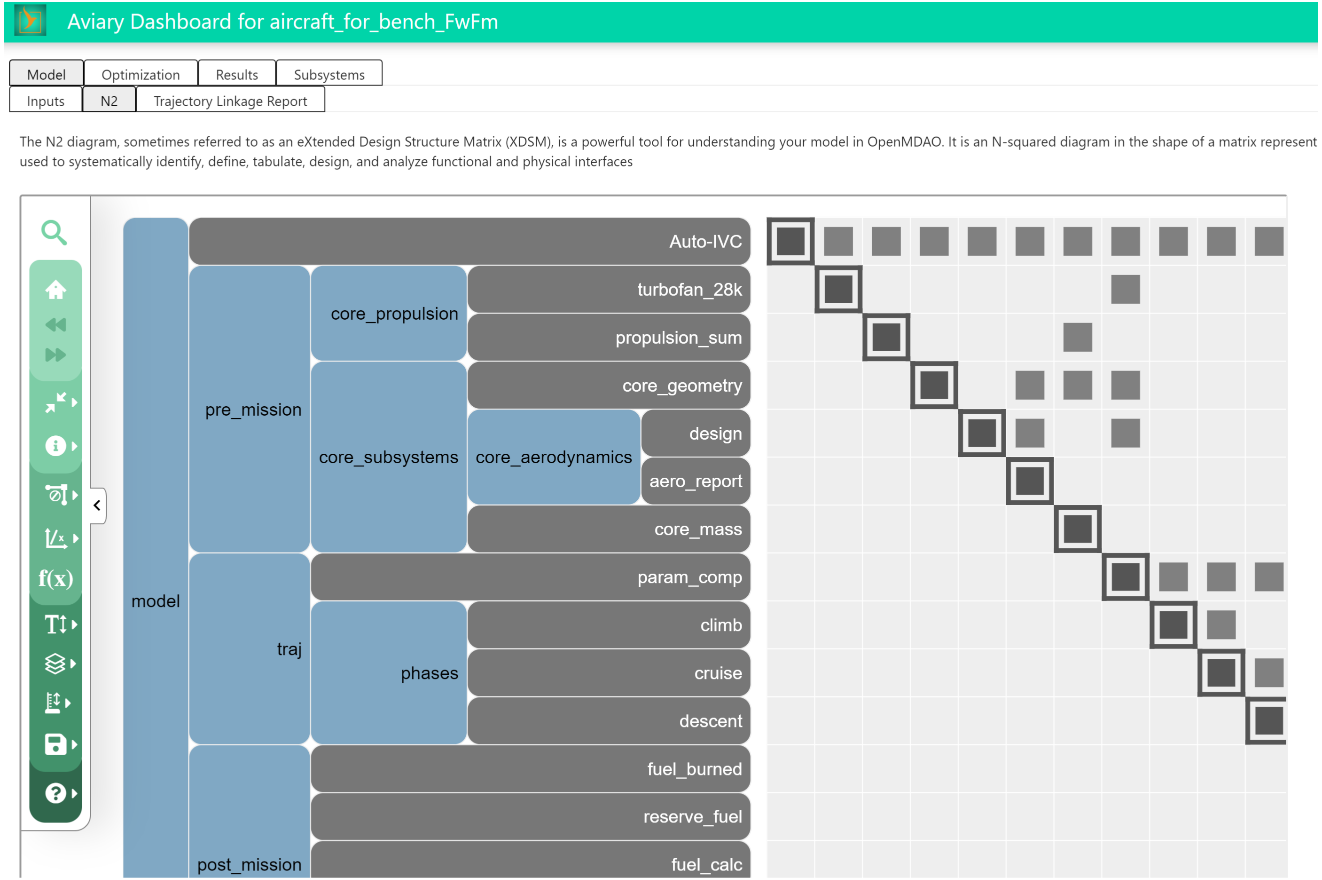
- Darrell (DJ) Caldwell (LaRC)
- Jennifer Gratz (GRC)
- John Jasa (GRC)
- Carl Recine (ARC)
- Kenny Lyons (ARC)
- Ben Margolis (ARC)
- Samara Murri (formerly LaRC)
- Erik Olson (formerly LaRC)
- Janet Ross (LaRC)
- Dahlia Pham (ARC)
- Jeff Chapman (GRC)

Current and Past Advisors

- Ben Phillips (LaRC)
- Eric Hendricks (GRC)
- Justin Gray (formerly GRC)
- Rob Falck (GRC)
- Jeff Bowles (ARC)
- Joseph Garcia (ARC)

Aviary activities have been co-funded by the T^3, AATT, and EPFD projects

Understanding your Results



Understand, debug, and parse results with interactive reports

- High-level summary of final aircraft design
- Detailed optimization reports
- Detailed reports from each subsystem
- Plots and figures of aircraft trajectory and time-dependent variables

Adding External Subsystems

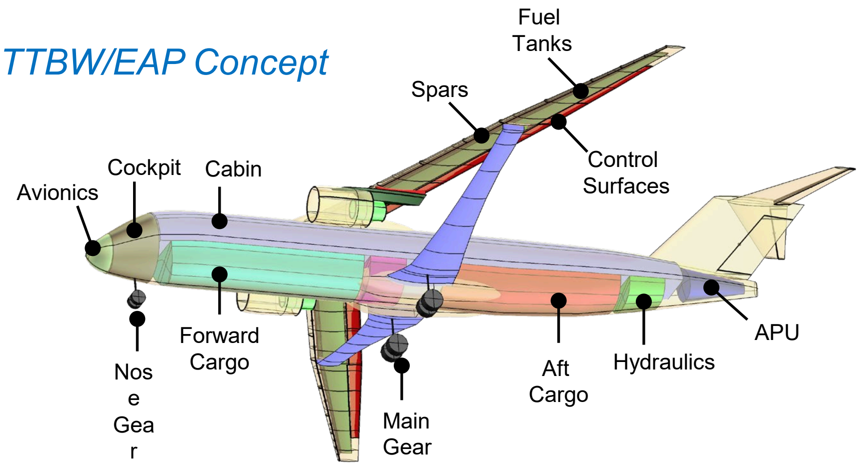
- User-defined modules need to tell Aviary what to expect from your system:
  - The states you want to integrate across the mission
  - Any new variables your system needs
  - The constraints, parameters, and design variables
- Aviary provides the `SubsystemBuilderBase` object, which you use to create your builder

```python
def build_pre_mission(self):
    '''
    Build an OpenMDAO system for the pre-mission computations of the subsystem.

    Required for subsystems with pre-mission computations.

    Used in level3.py to build the pre-mission system.

    Returns
    -------
    pre_mission_sys : openmdao.core.System
        An OpenMDAO system containing all computations that need to happen in
        the pre-mission (formerly statics) part of the Aviary problem. This
        includes sizing, design, and other non-mission parameters.
    '''
    return om.Group()

def get_states(self):
    '''
    Return a dictionary of states defined by this subsystem.

    Required for subsystems with mission-based dynamics.

    Note that there must be outputs provided by the user's model that provide
    the time derivative of each state. The convention is to name the output
    f"{state_name}_rate", where `state_name` is the name of the state variable.

    Use in the phase builders (e.g. cruise_phase.py) when other states are added to the phase.

    Returns
    -------
    states : dict
        A dictionary where the keys are the names of the state variables
        and the values are dictionaries with the following keys:

        - 'units': a string indicating the units of the state variable
        - any additional keyword arguments required by Dymos for the state variable.
    '''
    return {}
```

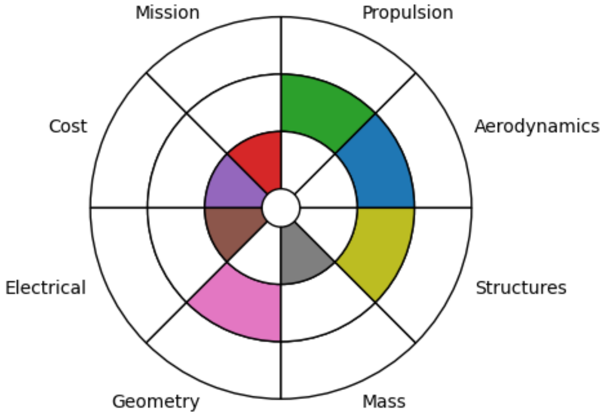
Using Aviary to Model the TTBW

- Non-proprietary *TTBW Tech Collector* developed by NASA to allow future vehicle technology studies, open publication of results, and easier collaboration with parties outside of NASA.
- External subsystems used to model geometry, aerodynamics, and propulsion
- EAP technology demonstration for FY23 includes an electric climb assist

TTBW/EAP Concept



TTBW Discipline Model Fidelity

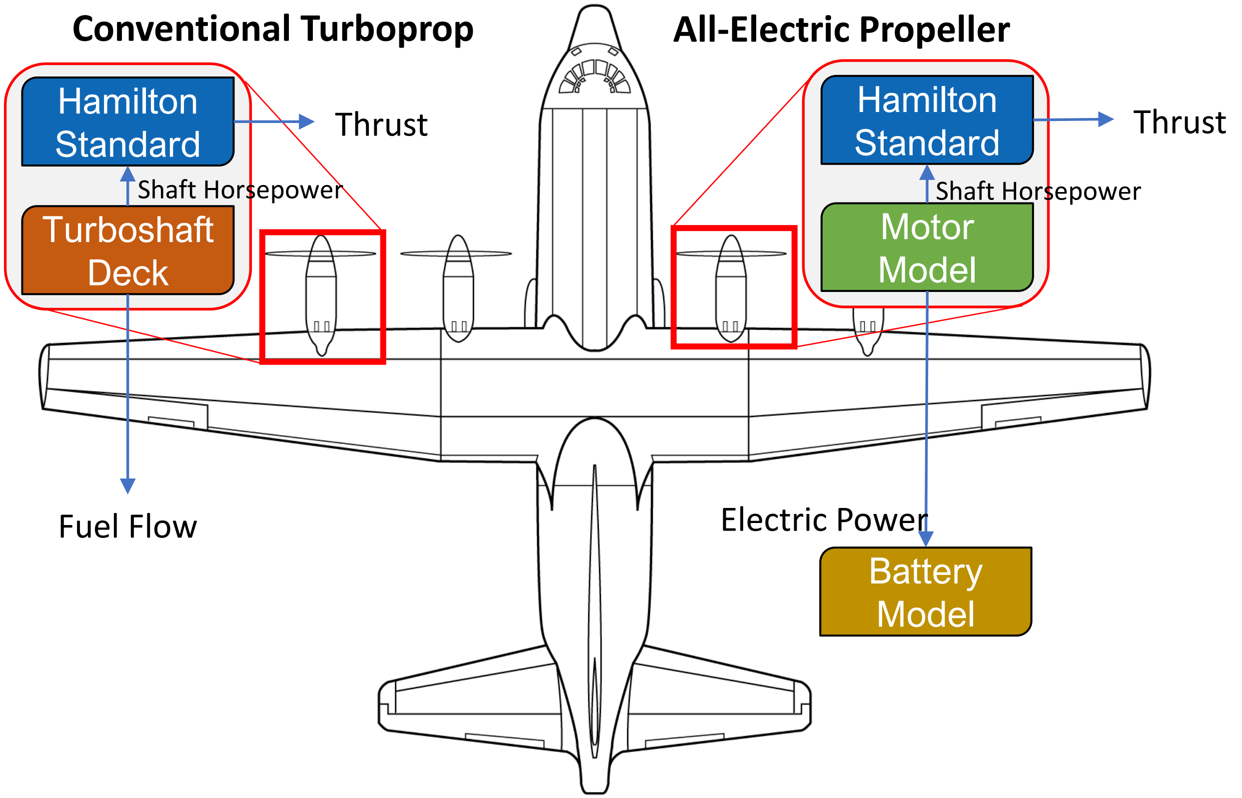


Using Aviary to Model Electrified Vehicles

Conceptual Electrified Freighter

- Hybrid-electric, based on C-130
  - Outer pair of turboprops replaced with all-electric driven propellers
  - Legacy tools struggle to model this aircraft
    - Throttle split between fuel and electric propulsors must be optimized
- Aviary can model this vehicle out-of-the-box
  - Structural weight and aerodynamic performance calibrated to publicly available data
  - Multiple unique engine models
    - Conventional turboshaft
    - All-electric propeller
  - Propeller modeling using Hamilton Standard
  - Simple motor and battery models included

Using Aviary to Model Electrified Vehicles



- New engine model "packages" for conventional and all-electric propulsors with propellers
- Reserve mission implementation with flexible definition (fuel and electric power use)