Marine Corps 4

Fellows: Divyam Khatri and Sam Thomas
Project Sponsor: Lieutenant Colonel Thomas Kline

NSIN

About the Team

Divyam Khatri

University of Texas - Dallas
2nd Year Double Majoring in Computer Science and Electrical Engineering

Sam Thomas

Georgia Institute of Technology
5th Year Computer Science Major

Background and the Problem

Aviation Logistics Division (ALD) advises the Commander, U.S. Marine Corps Forces Pacific, on aviation logistics, represents the Commander's interests across the Naval Aviation Enterprise, and provides support to subordinate organizations in order to ensure aircraft readiness.

The problem is that ALD relies too heavily on heuristics to predict how changes (in personnel, in supply delivery time, etc) affects aircraft readiness. ALD needs prediction tools: a discrete event simulator that models squadron maintenance.

The Solution

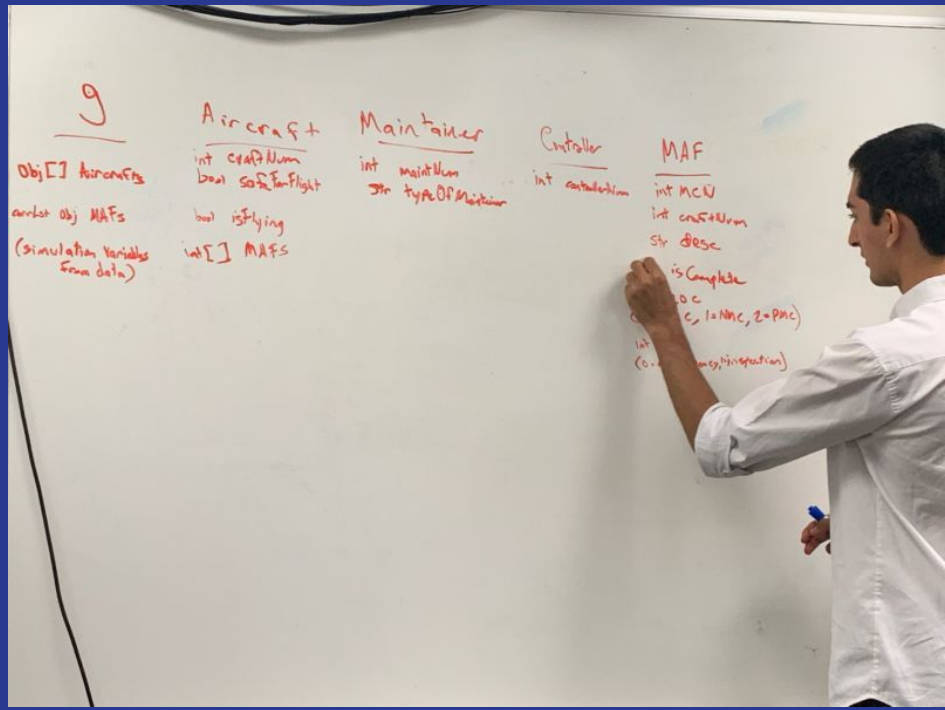
Our xForce team, along with our sponsor, are designing a simulator a simulation framework for Python. Python is open-source and available on the Marine Corps' network (a critically-important acceptance pre-requisite).

SimPy

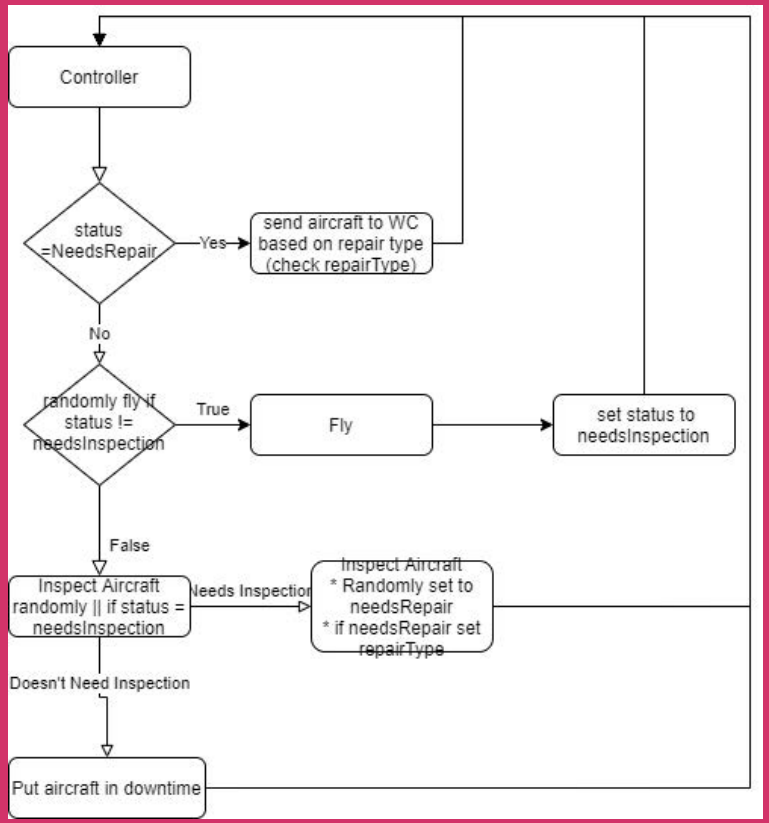
Work So Far

We learned the Maintenance Process through the other predictive tool our sponsor created: Maintenance Capacity Model (Microsoft PowerBI data analysis).

We traveled to Hawaii to interview squadrons directly and to work with our sponsor to design the simulation.



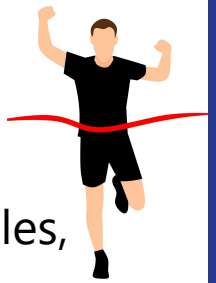
Simulation Design



V18SqdnMaintSimulation.py* | TRC.py | _g.py | _Aircraft.py

```
class Squadron_Model:
    def __init__(self, trial_number):
        self.env = simpy.Environment()
        self.aircraft_counter = 0
        self.controller = simpy.Resource(self.env,
            capacity=_g.gVars['numControllers'])
        self.flightlineMech = simpy.Resource(self.env,
            capacity=_g.gVars['numFlightlineMechs'])
        self.airframeMech = simpy.Resource(self.env,
            capacity=_g.gVars['numAirframeMechs'])
        self.aviTech = simpy.Resource(self.env,
            capacity=_g.gVars['numAviTechs'])
        self.pilot = simpy.Resource(self.env,
            capacity=_g.gVars['numPilots'])
        self.trial_number = trial_number
        self.mean_q_time_controller = 0
        self.mean_q_time_flightlineMech = 0
        self.mean_q_time_airframeMech = 0
        self.mean_q_time_aviTech = 0
        self.mean_q_time_flight = 0
        self.comebackthru = False
        self.results_df = pd.DataFrame()
        self.results_df["AC_ID"] = []
        self.results_df["Q_Time_Controller"] = []
        self.results_df["Q_Time_FlightLineMech"] = []
        self.results_df["Q_Time_AirFrameMech"] = []
        self.results_df["Q_Time_aviTech"] = []
        self.results_df["Total_Flight_Time"] = []
        self.results_df["Total_Flights"] = []
        self.results_df["Total_AF_Repair_Time"] = []
        self.results_df["Total_Avi_Repair_Time"] = []
        self.results_df["Total_FL_Repair_Time"] = []
        self.results_df["Total_Repair_Time"] = []
        self.results_df.set_index("AC_ID", inplace=True)
# Generate_AC creates a number of Aircraft objects and names them by counter
# Each aircraft is given a material condition state through random decisions
# Each aircraft is then given a flight decision to start
    def generate_ac(self):
        for i in range(_g.gVars['numAircraft']):
            self.aircraft_counter += 1
            ac_id = _Aircraft.Aircraft(self.aircraft_counter)
            #give some of the aircraft a material condition discrepancy
            #to discover during preflight inspection
            ac_id.fl_decision()
            ac_id.avi_decision()
            ac_id.af_decision()
            ac_id.flight_decision()
            self.env.process(self.Controller(ac_id))
            yield self.env.timeout(0)
# Controller considers material condition, flight schedule, and decides
# What each A/C next step in the process is
# if A/C has gripe, send to repair process
# if A/C on schedule to fly, send to preflight inspection
# if A/C not on schedule and no gripes, send to downtime
    def Controller(self, aircraft):
        start_q_controller = self.env.now
        with self.controller.request() as req:
            yield req
```

The Finish Line

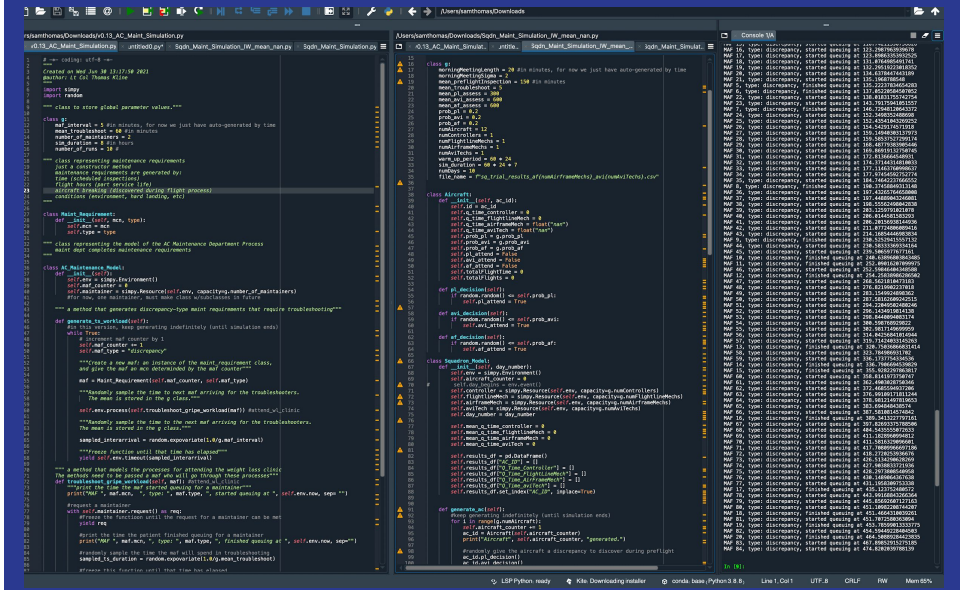


Our current model reads in variables, approximates the maintenance proces, and returns:

-mean time waiting for mechanics
-flight statistics per aircraft
-mean repair time by maint type

We are now working to:

-simulation interrupts (off-shift)
-supply parts wait time
-inputs by real data vs manual variables

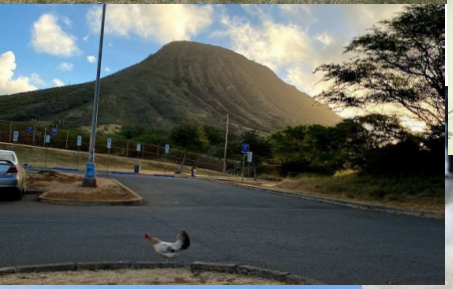


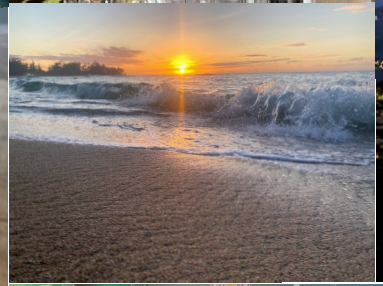
Fun Hawaii Pictures

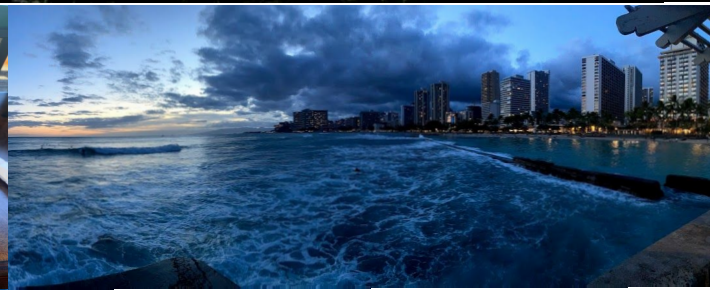






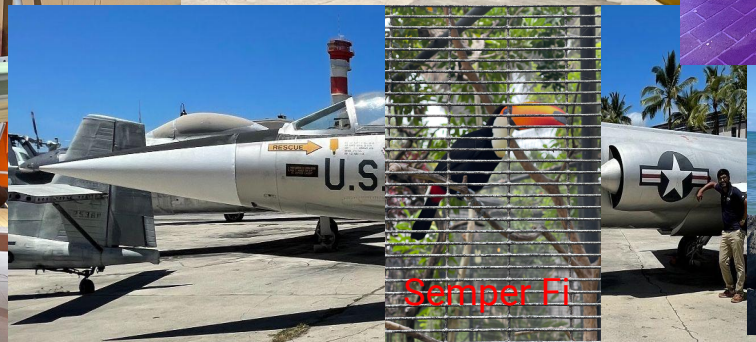





PLEASE
USE


KEEP OFF


RESCUE
U.S

Semper Fi

Mahalo for coming to our Demo!

Hang loose! 🤙