

Autonomous Cognitive Agent Flying Aircraft in X-Plane

Milestone 1 Progress Evaluation

Team Members:

Avery Mackey:	amackey2023@my.fit.edu
Dylan Ostolaza:	dostolaza2024@my.fit.edu
Zachary Reese:	zreese2023@my.fit.edu
Brianna Marshall:	bmarshall2022@my.fit.edu

Faculty Advisor:

Dr. Bhattacharyya: sbhattacharyya@fit.edu

Client:

Dr. Bhattacharyya, Department of Electrical Engineering and Computer Science:

Progress of current milestone:

Task	Completion %	Avery Mackey	Dylan Ostolaza	Zachary Reese	Brianna Marshall	To do
Investigate Tools	100%	25%	25%	25%	25%	None
Demos	100%	0%	0%	100%	0%	None
Technical Challenges	100%	25%	25%	25%	25%	None
Requirement Document	100%	0%	0%	0%	100%	None
Design Document	100%	0%	0%	100%	0%	None
Test Plan	100%	0%	100%	0%	0%	None

Discussion of each accomplished task (and obstacles) for the current Milestone:

- Investigate Tools
 - All of the tooling for this project was selected by previous contributors to this project, with our primary obstacle being adapting to the choices they made and learning how to properly use the tools already in use. The primary coding languages are Java and SOAR, the simulation environment is X-Plane with XPC and CCAT being used to link the agent to the environment. The only addition to the system so far is the inclusion of the apt.dat file which is a database that holds all of the information on airports simulated in X-Plane.
- Demos
 - Made a small demo video of the entire CCAT-agent system in action. Right now, the top level agent simply gets up to altitude and then cruises in a straight line towards the destination, only for demo purposes. Figuring out how to write SOAR production rules was a little tricky for this, but not too difficult. Setting up the system as a whole was relatively simple, though.
- Technical Challenges
 - The team has successfully addressed the three primary technical challenges identified at the beginning of the project. Through research and hands-on development, we have gained a working understanding of the SOAR architecture, including its production-rule system, memory model, and development workflow, allowing us to begin implementing and testing cognitive agent behaviors. We have also developed a sufficient understanding of X-Plane and relevant aircraft systems to interpret flight data, identify equipment failures, and evaluate their effects on aircraft behavior. Finally, we have familiarized ourselves with the existing codebase and the communication between SOAR, Java, and X-Plane, enabling us to extend the framework while preserving its existing functionality. Overcoming these initial challenges has provided the technical foundation necessary to move forward with implementing and evaluating the agent's ability to adapt flight plans in response to multiple aircraft equipment failures.
- Requirement Document
 - The document first involved communicating with team members and the faculty advisor to elicit major features/functions to identify the requirements. Because the project is an extension, other prerequisites involved examining the existing software and investigating the documents

and tools to determine what needed to be expanded or modified. Additional research was conducted to determine the specific types of equipment failures that could occur on the aircraft model. A challenge with the document was interpreting the IEEE guideline to fit the scope of the project.

- Design Document
 - The system for this project was already relatively in place, so the system design for this project was relatively straightforward. We have one novel addition to the system, being the way that the system is able to find the nearest n airports so the aircraft can safely divert. The apt.dat file is a database within X-Plane 12 that stores data about every airport available and is available to be read with a plugin. The only real hurdle with this document was finding an effective method for locating airports. An external API was considered, but then it was realized that X-Plane has a database tracking all of this, so we will just add to the CCAT java program a class that tracks the n closest airports.
- Test Plan
 - During this milestone, we developed the testing plan for the ACAFAX Flight Equipment Failure Agent. We created test cases for the initial flight plan, equipment failure initiation, failure detection, agent response, and flight route modification. We also identified expected outputs for each test to make sure the system can properly generate and validate flight plans, detect equipment failures, respond to failures, and modify routes when necessary. One of the main obstacles was breaking the overall system into specific functions and determining how each function should be tested, especially for situations involving multiple failures or normal sensor fluctuations that should not be identified as actual failures.

Discussion of the contribution of each team member to the current Milestone:

- Avery Mackey
 - Coordinated meetings, met with several flight students and instructors to discuss most common equipment failures in various aircraft, and researched SOAR production rules. Speaking with people in flight was illuminating, particularly in realizing the breadth of potential causes for any particular component failure.
- Dylan Ostolaza

- I was responsible for creating the test plan for this milestone. I organized the testing into the five core functions of the ACAFAX system: initial flight planning, equipment failure initiation, failure detection, failure response, and flight route modification. For each function, I developed test cases with specific procedures and expected outputs to explain how the system should be tested and what results should occur.
- Zachary Reese
 - Recorded all of the demo videos, got the entire system running, and created the system design document. The demo videos were relatively straightforward, as that only required a video of X-Plane itself to see agent output, and a second video of the GUI for the CCAT program to see the agent controls. The design document was straightforward as well, as we are building on an existing system, with novel features. It was interesting to comb through the existing codebase for all of this, and try to learn a little more about the SOAR production rules to make the demo for the high level orchestrator agent. Right now in the demo video, the agent simply flies to 5000 feet and then makes a straight line for its destination airport.
- Brianna Marshall
 - Organized and refined the requirements gathered in the SRS format. This included explicitly defining the major features, reviewing existing documentation, asking clarifying questions, and mapping the project needs against the SRS guidelines. Additionally, distinguishing the work needed for the functional, non-functional, and interface requirements.

Plan for the next Milestone:

Task: <i>Implement, test, and demo...</i>	Avery Mackey	Dylan Ostolaza	Zachary Reese	Brianna Marshall
Triggering and communicating with at least one aircraft equipment failure from X-Plane to the SOAR agent	10%	10%	70%	10%
The agent's ability to follow a complete flight plan in the X-Plane simulation with no	10%	10%	10%	70%

complications				
SOAR rules that recognize the selected failure and determine an appropriate response based on the aircraft's remaining capabilities	70%	10%	10%	10%
The agent issuing the appropriate commands in response to the failure	10%	70%	10%	10%

Discussion of each planned task for the next Milestone:

- *Implement, test, and demo...*
 - Triggering and communicating with at least one aircraft equipment failure from X-Plane to the SOAR agent
 - To do this, we will need to implement several additions to the existing system. First, we need to add to the CCAT program a class that is able to manipulate datarefs to simulate equipment failure. There are specific datarefs for sensors, engine throttle, etc that we can manipulate through CCAT and the XPC plugin. Next, we will need to add SOAR production rules that check these datarefs for any failures. Finally, we will need to add a way to trigger these failures, or potentially make them happen at random, so the agent is unprepared for it. This can be implemented via the GUI. We can add a button to trigger an equipment failure to the existing GUI. The flow for this becomes a button press in the GUI, CCAT manipulates the dataref for the sensor/equipment, and the agent detects it.
 - The agent's ability to follow a complete flight plan in the X-Plane simulation with no complications
 - Review the current codebase and X-Plane functionality to understand how flight plans and waypoints are represented and passed between the simulation and the agent. Add any additional logic that might impede this. Monitor the aircraft during the flight to confirm it is following the correct route by testing multiple flight plans and recording any issues encountered during normal operations. Record a demonstration of the aircraft completing a full flight plan.

- SOAR rules that recognize the selected failure and determine an appropriate response based on the aircraft's remaining capabilities
 - Implementation will involve developing rules that identify failures using data received from X-Plane, evaluate their effects on aircraft functionality, and select appropriate actions to maintain safe flight. Testing will be conducted through simulated failure scenarios in X-Plane to verify that the agent correctly identifies each failure and selects an appropriate response given the aircraft's operational limitations. Demonstrate the completed functionality by introducing a selected equipment failure during a simulated flight and showing how the SOAR agent recognizes the failure, evaluates the aircraft's remaining capabilities, and determines an appropriate course of action.
- The agent issuing the appropriate commands in response to the failure
 - To do this, we will need to define all expected responses for each equipment failure, identify all commands that the agent can send to X-Plane, and implement any missing command logic that's passed from the agent to the aircraft via the Java application. When the response is chosen, we will verify that the agent's commands are sent properly, and that X-Plane executed each command as expected. All commands will be tested with their associated equipment failures, and any unusual results will be recorded for further debugging.

Date(s) of meeting(s) with Client during the current Milestone:

- September 21 at 9:30 a.m.
- September 24 at 3:30 p.m.

Client/Faculty Advisor feedback on each task for the current Milestone:

- Investigate Tools
 -
- Demos
 -
- Technical Challenges
 -
- Requirement Document
 -
- Design Document
 -
- Test Plan
 -

Faculty Advisor Signature:_____

Date:_____

Evaluation by Faculty Advisor

Faculty Advisor: Detach and return this page to Dr. Chan (HC 209) or email the scores to pkc@cs.fit.edu

Score (0-10) for each member: circle a score (or circle two adjacent scores for .25 or write down a real number between 0 and 10)

Avery Mackey	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Dylan Ostolaza	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Zachary Reese	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Brianna Marshall	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10

Faculty Advisor Signature:_____

Date:_____