Simulation and Flight Test Environments for the TASAR Traffic Aware Planner

Terique L. Barney,[1] Matthew C. Underwood,[2] and Bill K. Buck[3]
NASA Langley Research Center, Hampton, VA, 23681

The Traffic Aware Planner (TAP) software is a flight deck decision support tool that enhances the flight crew's ability to make flight-optimizing route change requests while airborne. The software provides conflict-free, optimized trajectory suggestions during en route flight to produce time- and fuel-savings compared to the current trajectory. The TAP software requires evaluation in an operational environment with real pilot users to validate projected benefits. To this end, a set of developmental test environments have been developed to mature the software and mitigate technical risk prior to entering operational evaluation. The unique attributes of each test environment were leveraged to provide a range of purpose- and case-dependent TAP software tests. This paper describes the elements of a testing environment, discusses several environments of varying fidelity used to test the TAP software, and provides a review of two case studies highlighting the vital role testing played in the TAP software development process.

I. Introduction

In today's operational environment, the flight crew may request amendments to their current route to achieve a variety of objectives such as saving time or fuel. However, the flight crew often makes these requests with limited situation awareness of constraints and restrictions that would affect the proposed route change [1]. Sometimes these requests are denied by air traffic control (ATC) due to conflicts with nearby traffic, airspace constraints, or convective weather. Repeatedly denied requests may discourage pilots from making future requests that could improve flight efficiency. The ability of the flight crew to make better-informed requests to optimize the flight, especially as air traffic continues to increase in volume and complexity, is a potential avenue for improving operations within the national airspace system.

Traffic Aware Strategic Aircrew Requests (TASAR) is an operational concept that empowers the flight crew to make better-informed reroute requests, thereby increasing the likelihood of ATC approval [2]. The TASAR concept is enabled by the Traffic Aware Planner (TAP) software, a flight deck decision support tool hosted on an Electronic Flight Bag (EFB) platform that provides pilots with time- and fuel- saving trajectory alternatives compared to the current active route [3]. The TAP software incorporates up-to-date aircraft state data from onboard avionics, as well as the latest position of surrounding traffic, the most recent wind forecast, and the most recent convective weather forecast, in order to calculate deconflicted candidate trajectory modifications that improve upon the current active route. By using this software during flight, the aircraft operator is able to propose user-preferred trajectories to ATC that may reduce the duration or overall operating cost of that flight.

The National Aeronautics and Space Administration (NASA) partnered with Alaska Airlines to conduct an operational evaluation of the TAP software. This activity focused on validating the projected time and fuel benefits accrued by using the software in an operational environment. Discovering and correcting software issues prior to deployment of the software to Alaska Airlines was essential to the success of the operational evaluation and the overall mission of technology transfer to industry. Accordingly, a test program utilizing a series of test environments of varying fidelity was developed to prepare the TAP software for the operational evaluation. These test environments provided a platform to exercise the software in a wide range of test conditions, thereby facilitating rapid detection of software issues and reducing risks.

This paper discusses the environments used to conduct simulation and flight testing as a critical step toward validation of TAP software in a relevant operational environment. A number of different test environment options are

[1] Aerospace Engineer, Crew Systems and Aviation Operations Branch, M/S 152.
[2] Aerospace Engineer, Crew Systems and Aviation Operations Branch, M/S 152, AIAA Member.
[3] Aerospace Engineer, Crew Systems and Aviation Operations Branch, M/S 152, AIAA Senior Member.

described, including an operational test environment, a flight test aircraft, desktop flight simulator, as well as other simulation environments and tools suitable for rapid turnaround testing of developmental software builds. This paper also provides background information on components inherent to these test environments, provides descriptions of a tool that enables playback of previously recorded data, and concludes by discussing two case studies that utilized several of the test platforms to discover software issues and test the implemented solutions.

II. Test Environment Elements

Test and evaluation of the TAP software system is enabled by the integration of three essential elements into each test environment (see Fig. 1): the TAP system installation (described in section II.A), connectivity to real or simulated avionics data (described in section II.B), and connectivity to internet data (described in section II.C). Avionics and internet data could either be live or recorded data. The test configurations described in this paper combine the test environment elements in a variety of ways in order to evaluate TAP's technical performance and to identify areas to improve its robustness.

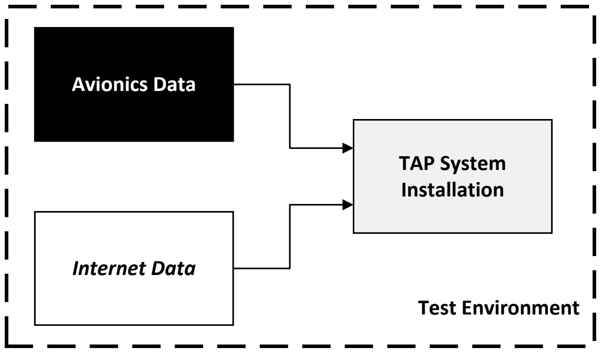

Fig. 1 Elements of TAP System Test Environment.

A. TAP System Installation

The TAP software aspect of the system installation consists of four primary components shown in Fig. 2: the TAP Display, TAP Display Adapter, TAP Engine, and External Data Server (EDS). TAP software system components are initiated using the TAP startup services, which sends commands to the other three components to begin their connections and processing when the user starts the TAP Display. While the basic configuration shown in Fig 2 contains one instantiation of TAP, the expected configuration of operational use includes two instances of TAP: one set for the Captain, and the other set for the First Officer. Each instance consists of a standalone set of three of the primary TAP components: a TAP Display, TAP Display Adapter, and TAP Engine, with a single EDS shared by both instances. The components may all run on a single computer, or may be distributed across various networked hardware.

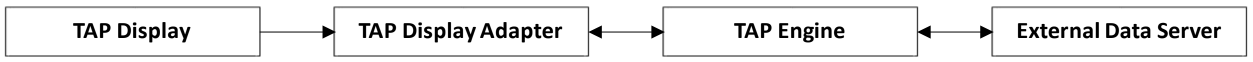

Fig. 2 TAP Software System Component Interaction.

The TAP Display is a human-machine interface that facilitates pilot interaction with the software. It displays optimized trajectory solutions, conflict characteristics, fuel and time outcomes, ownship route information, and additional information about the state of the system. The TAP Display interacts with the TAP Engine through the TAP Display Adapter (TDA), which aids communication between the TAP Display and TAP Engine. The TDA re-synchronizes connections between the TAP Display and TAP Engine after the TAP Display hardware goes into 'sleep' mode or the TAP Display application is pushed to the tablet background during device operations. Additional information for the TAP Display and TDA is provided in Ref. [4].

The TAP Engine performs all of the processing necessary to generate route optimization advisories. It reads and reformats data inputs for internal use and responds to pilot inputs on the TAP Display affecting processing. The TAP Engine also sends optimized solutions, convective weather and Special Use Airspace (SUA) polygons, three-dimensional (3D)-gridded winds, route data, and other information to the TAP Display for presentation to the user. The EDS uses internet connectivity to gather the external data required for TAP to conduct its processing, and provides it as input to the TAP Engine. This external data enhances the validity of TAP's optimization options by enabling information relevant to the ownship's operating environment to be considered. Further detail on the EDS is provided in section II.C and Ref. [4].

B. Avionics Data

The TAP Engine requires data from various avionic systems, such as ownship state, route, and Automatic Dependent Surveillance-Broadcast (ADS-B) In traffic data, and is currently designed to accept this information in a Simple Text Avionics Protocol (STAP) formatted data feed. An Aircraft Interface Device (AID) receives and decodes ARINC 429 and ARINC 717 data from aircraft avionics. An ARINC 834 server running on the AID outputs the avionics data to the TAP Engine in a STAP format. As currently implemented, the TAP Engine must receive its aircraft avionics data from an ARINC 834 server.

Alternatively, avionics data may originate from a flight simulator capable of providing avionics data to TAP in the proper format, such as the NASA Langley Aircraft Simulations for Traffic Operations Research (ASTOR). Each instantiation of ASTOR is a medium-fidelity, networked, and interactive desktop simulation of a commercial transport aircraft and its flight deck systems [5]. ASTORs were leveraged to enable robust and rigorous testing prior to testing on an aircraft. ASTOR simulations enable TAP software testers to execute commands on the simulated aircraft using a flight deck interface, while simultaneously operating the TAP Display. An ASTOR uses an emulation of ADS-B In/Out, simulates airspace conditions such as traffic and wind, and provides aircraft state data in a syntax similar to the ARINC 429 data protocol. Additionally, each ASTOR has a software emulation of the AID called a Data Concentrator Emulator (DCE). Connections between the DCE and TAP Engine mimic real-world connections between the AID and TAP Engine which, when used with emulated traffic data, allows assessment of TAP's technical performance to occur in simulation environments. Further information regarding ASTOR and the simulation capability is provided in Ref. [6].

C. Internet Data

The EDS periodically downloads and processes data obtained via airborne internet. These data include 3D-gridded wind and temperature data directly from the National Oceanic and Atmospheric Administration (NOAA), SUA activation schedules from the Federal Aviation Administration (FAA), and convective weather data from a NASA-hosted Ground Data Server (GDS) [7]. GDS gathers convective weather data from a commercial weather service provider and converts it into convective weather polygons for use by the TAP Display and TAP Engine. EDS makes these data available to instances of TAP (e.g. Captain and First Officer) that are connected to it. The trajectory optimization algorithms in each TAP instance considers potential conflicts to the active route obtained from this data. For more information on weather and atmospheric data obtained via the internet and used by TAP, see Ref. [7].

III. TAP Test Environments

Although Alaska Airlines' aircraft were used for evaluating the TAP software in an operational testing environment [8], installation of the TAP software onto vendor hardware was a non-trivial certified process with limited opportunities for rapidly and frequently deploying updated versions of the software. Therefore, the TAP software code delivered to Alaska Airlines had be thoroughly tested to ensure it was stable and of sufficient reliability to accurately assess the usability and benefit in the Alaska Airlines operational environment [9]. User-based testing in developmental test environments was performed to accomplish this goal. These test environments enabled researchers to:

1) introduce a range of flight conditions and test configurations unavailable in the operational environment,
2) increase control over test variables that impacted TAP technical performance,
3) verify the functionality of TAP using requirements-based testing in simulation and flight test environments,
4) build confidence in the exhaustive coverage of performance risk areas and the accuracy of risk levels identified in test results, and
5) evaluate TAP for technical acceptance based on researcher and Subject Matter Expert (SME) inputs.

Developmental test environments described in this section were used to prepare TAP software builds for deployment for operational evaluation. The Alaska Airlines aircraft was the operational environment to which all other remaining developmental test environments are referenced. A tabulated comparison of the components of the Alaska Airlines aircraft, NASA HU-25A, Simulation Test Lab, and Virtual Machines (VM), as well as the TAP Playback tools, is provided in Table 1. A check mark (✓) in the table represents an exact match to the component installed on the Alaska Airlines aircraft listed in the same row. Test environments with components distributed across hardware (three left columns with test environments) and components co-located on hardware (two right columns) are separated by a bold vertical line. Comparisons of these systems are provided in detail in Ref. [10].

The test environments are listed from left-to-right in order of decreasing fidelity from the operational environment, as well as increasing convenience and frequency of usage for testing. The sequence in which each test environment was utilized is not fixed, and varied depending on the purpose of the testing.

Table 1 Similarities and Differences between Alaska Airlines Aircraft, NASA HU-25A, Simulation Test Lab, Virtual Machines, and TAP Playback Tools.

	Alaska Airlines aircraft	NASA HU-25A	Simulation Test Lab	Virtual Machines	TAP Playback Tools
Captain TAP Software System	Gogo ACPU2 Processor (TAP Engine)	Windows Laptop (TAP Engine, TDA)	Linux Box 1 (TAP Engine, EDS)	N/A	N/A
	UTAS TIM Processor (TDA)	Direct Wi-Fi connectivity from iPad to TDA	✓		
	Apple iPad (TAP Display)	✓	✓		
First Officer TAP Software System	UTAS AID2 Processor (TAP Engine, EDS)	Windows Laptop (TAP Engine, EDS)	Linux Box 2 (TAP Engine)	N/A	N/A
	UTAS TIM Processor (TDA)	Windows Laptop (TDA)	✓		
	Apple iPad (TAP Display)	✓	✓		
Aircraft State and Route Data Capability	UTAS AID2 ARINC 834	✓	ASTOR DCE Emulated ARINC 834	ASTOR DCE Emulated ARINC 834	Recorded aircraft data
Internet Capability	Gogo ACPU2 Internet Terminal ~10 MB per second	Cobham Aviator 200 System ~200 KB per second	Ground-based Internet ~1 GB per second	Ground-based Internet ~1 GB per second	Recorded internet data

A. Alaska Airlines' Aircraft

Alaska Airlines installed the TAP system (hardware and software) on three commercial transport aircraft to support the operational evaluation [11]. In this configuration, depicted in Fig. 3a, the TAP Display runs on each of two Apple[4] iPad EFB interfaces. These EFB interfaces each connect through a UTC Aerospace Systems (UTAS[5]) Tablet Interface Module (TIM) to an avionics processor hosting the TAP Engine. The TIMs also have processors that run the TDA software for its client TAP Display – one for the Captain (Capt. in Fig. 3a), and one for the First Officer (F.O. in Fig. 3a). On Alaska Airlines' aircraft, the Captain's instance of the TAP Engine runs on a Gogo Airborne Control Processor Unit-2 (ACPU2) and the First Officer's instance of the TAP Engine runs on a UTAS AID generation 2 (AID2). The AID2 connects the aircraft's avionics bus to the EFB, providing ADS-B In, ownship state data and route information to each TAP Engine using ARINC 834 STAP. The connection between avionics data buses and the AID2 (see Fig. 3a) allows read-only access to avionics data, ensuring TAP has no effect on safety-critical systems while obtaining necessary data for optimized trajectory generation. The EDS (shown on the AID2 in Fig. 3a), which can run on either the AID2 or ACPU2, receives data collected from external sources via airborne internet provided by the ACPU2.

B. NASA Langley HU-25A Guardian Aircraft

The NASA Langley HU-25A Guardian aircraft was chosen as a developmental test environment for final testing before deployment to Alaska Airlines [10]. The HU-25A has a broad flight envelope representative of commercial airline operations. The HU-25A test platform architecture (shown in Fig. 3b and described further in Ref. [10]) allows greater flexibility and control over the test conditions and scenarios than the operational environment.

[4] The use of trademarks or names of manufacturers in this report is for accurate reporting and does not constitute an official endorsement, either expressed or implied, of such products or manufacturers by the National Aeronautics and Space Administration.

[5] UTAS has changed its name to Collins Aerospace since the procurement of hardware used in the operational evaluation and developmental testing.

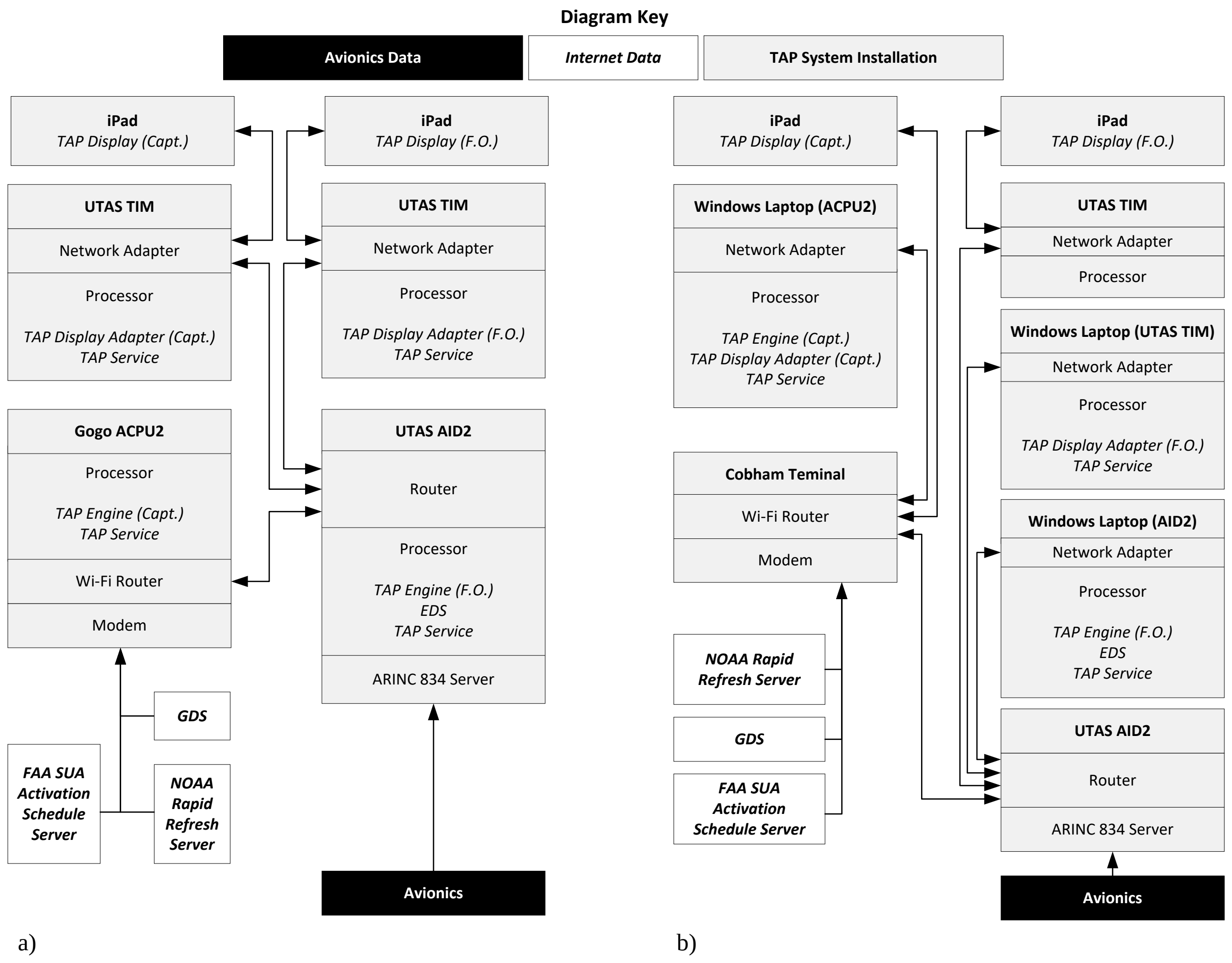

Fig. 3 Alaska Airlines Aircraft (a) and NASA HU-25A (b) TAP System Configurations.

The flexible, extensible, and re-configurable architecture of the HU-25A Guardian [10] closely emulates the Alaska Airlines operational test environment depicted in Fig. 3b. However, there are some notable and intentional differences between each system configuration, as seen in Table 1. For example, the processors used on the HU-25A to run the TAP Engine and TDA are on Windows laptops instead of the Linux-based ACPU2, TIM, and AID2 systems on each Alaska Airlines' aircraft. The rationale behind this decision was to allow rapid and frequent deployment of the TAP software to each computing platform. An additional benefit mechanism of using laptops is mobility. The hardware hosting the TAP software system on the HU-25A can be packed up and transported between the Simulation Test Lab and the HU-25A with ease. This enabled preliminary ground testing to be carried out using the same equipment used on the aircraft.

Furthermore, the TAP Display could be hosted on either Windows Surface Pro tablets or Apple iPads. The Surface Pro tablets enabled additional capabilities useful during development, including consolidation of output data and testing of TAP software on an alternate EFB interface.

C. Simulation Test Lab

The Simulation Test Lab shown in Fig. 4 enables developmental testing of TAP's basic functionality (e.g. software component communication, functional requirements, etc.) using the same EFB interfaces (i.e. Apple iPads) and UTAS TIM hardware used in the Alaska Airlines aircraft environment. Two Linux computers in the Simulation Test Lab environment emulates the ACPU2 (hosting the Captain's TAP Engine) and AID2 (hosting the First Officer's TAP Engine) processors in the operational environment. An ASTOR hosted on a Windows computer produces simulated aircraft avionics data (ownship state, route and traffic information), and a DCE hosted on the same computer outputs those data to the TAP Engine. The Linux computer emulating the ACPU2 processor hosted the EDS (see Table 1 in section III.E). Data between all components is transferred via a wired local area network.

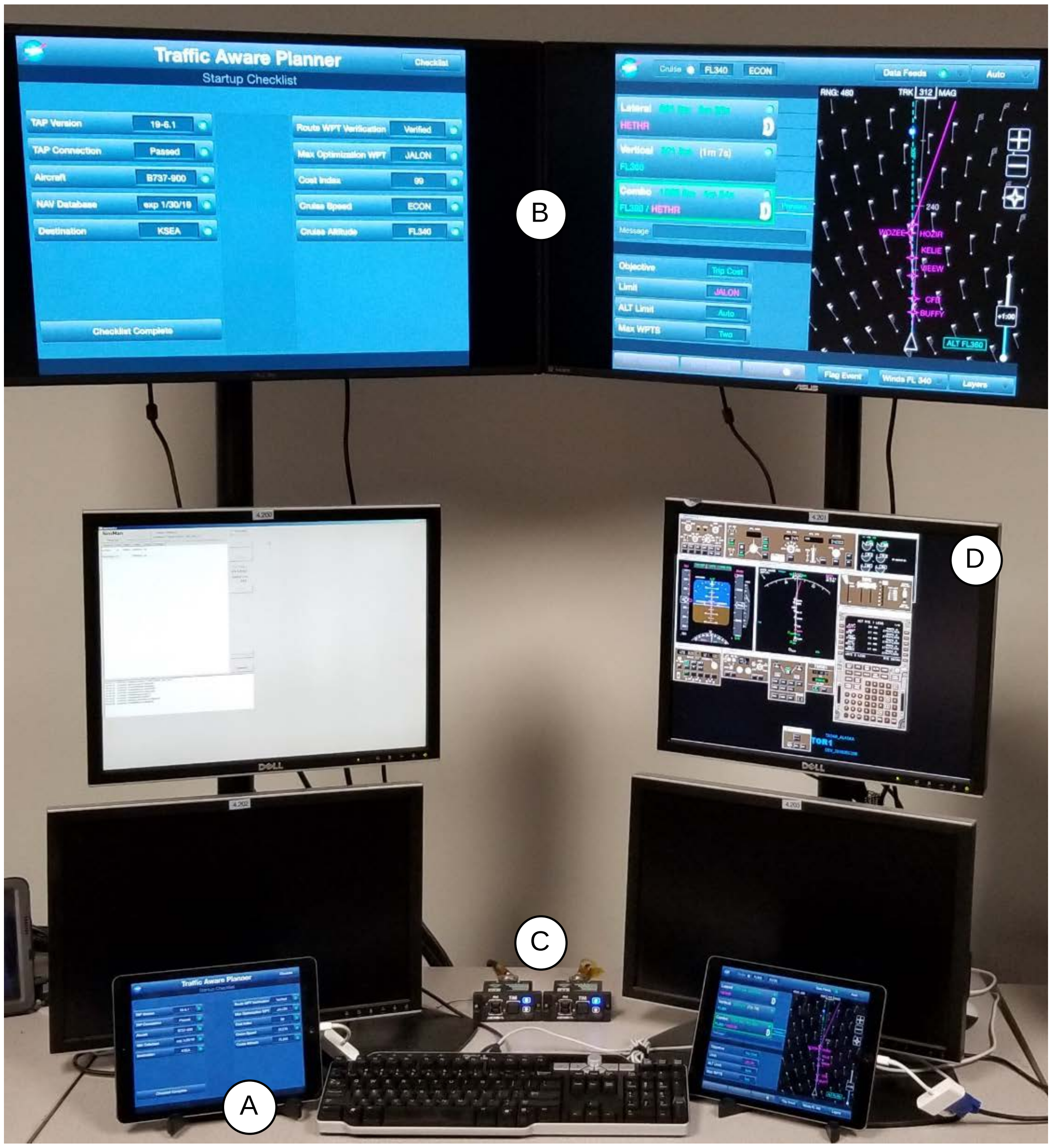

Fig. 4 Example of Simulation Test Lab Setup.

In Fig. 4, the Captain and First Officer iPad screens (label A) showing the TAP Display were duplicated onto larger monitors (label B) for demonstrations. Each iPad was connected to either the ACPU2 or AID2 emulator via the UTAS TIM connection (label C). The interactive ASTOR (label D) running in real-time on a Windows computer generated simulated avionics data. The ACPU2 and AID2 emulators hosting the TAP Engine and EDS are not shown in Fig. 4.

The Simulation Test Lab provided a fast and convenient opportunity to evaluate TAP software performance and verify its functionality with repeatable simulated inputs in an environment similar to the operational environment. This environment was useful in enabling researchers and SME pilots to uncover TAP related issues more quickly and cost-effectively. These issues included incomplete data transfers, network connectivity issues, improper display artifacts, and inadequate generation of weather and traffic-conflicting trajectory alternatives. Researchers and testers captured issues before prematurely deploying resources into higher fidelity flight test environments, and reported the issues to developers to be debugged and refined in later developmental software builds.

D. Virtual Machines

VM developmental test environments were useful in conducting functional testing and quick-look assessments of TAP software developmental builds before deploying them to higher fidelity environments. Many user-level TAP issues that would have impeded test and evaluation of higher value technical performance areas were identified with

VMs and quickly addressed in a repeatable manner. Furthermore, the VMs allowed testers to evaluate various TAP system configurations on a low-risk testing environment. VM instances could also be created and terminated with relative ease and control, thus simplifying overhead project cost management while providing testing flexibility.

VMs were the most flexible of all of the test environments, allowing testers to use a variety of test configurations and TAP component locations for functional-level testing of TAP software external to the Simulation Test Lab. Amazon Web Services GovCloud was used to generate VM environments capable of providing avionics and internet data to the TAP software. VMs enabled rapid turnaround testing by providing immediate user access to the latest software builds and other dedicated remote machines on the same network. This facilitated collaborative implementation and evaluation of the TAP software with software developers. VMs were also assigned to multiple users permitting concurrent test runs, sharing of live content, collaborative troubleshooting efforts, or multi-user experiments.

E. TAP Playback Tools

TAP playback tools were test mechanisms that allowed testing of previously recorded runs of the TAP software in a variety of computing environments. These tools granted testers increased flexibility, repeatability, and control over test conditions affecting TAP software behavior. The TAP software playback tools were useful in analyzing anomalies observed during testing conducted in aircraft test environments, as well as the reproduction of key test scenarios during live demonstrations. These tools enabled timely playback of the data collected to observe TAP software behavior in a variety of ways without re-running the software on an aircraft or in a simulator.

Three types of playback tools were available for testing. The interactive playback tool used the same flow of avionics and internet data obtained by the TAP software during a previous run and allowed a user to provide dynamic inputs to the TAP software and observe outputs in real time. Non-interactive playback with a display were mechanisms through which the user could view previously recorded interactions with the TAP Display. Non-interactive playback without display were used as a TAP Engine debugging feature to show the information flow between TAP system components without visual playbacks.

IV. Case Studies

This section presents two case studies that illustrate how a combination of test environments were used to quickly discover issues with the behavior and functionality of the TAP software, and test the effectiveness of solutions implemented to address them. The first case study highlights a behavioral issue with how the TAP software handles avoidance polygons after the limit waypoint. The second case study describes a user issue regarding the flight crew's requirement to manually track and input cruise altitude and cruise speed settings in the TAP software, which led to the implementation of new functionality. Each example presents the issue, discusses how the issue was identified, and describes how the implemented solutions for each issue were tested in several environments.

A. TAP Trajectory Solutions Inhibited by Downstream SUA Impacting the Active Route

The TAP software probes prospective trajectory solutions for conflicts (e.g., traffic, convective weather, or SUAs) and suppresses any solutions that that have one. For hazards defined by avoid-area polygons (convective weather and SUAs), the TAP software considers the lateral bounds (including any lateral buffer if appropriate), valid altitudes, time of applicability, and look-ahead time when assessing solutions for conflicts. The look-ahead time is defined as the individually specified time horizon after which the conflict is ignored by the TAP Engine (e.g. a conflict with convective weather is ignored if it occurs more than two hours in the future). To deconflict proposed trajectories, the TAP Engine generates solutions with up to two, off-route, named waypoints, which rejoins any named waypoint along the active route up to the limit waypoint. Beyond the limit waypoint, the active route cannot be changed by the TAP software. The last waypoint is set by default to the limit waypoint; however, the user may manually change it and set any route waypoint as the limit.

During an HU-25A test flight, with a developmental version of the TAP software, the route passed through the buffered lateral bounds of an SUA scheduled to be active and located after the limit waypoint. The TAP software correctly identified this as a conflict and was therefore unable to produce any trajectory solutions for the entirety of the flight, to include the portions of the flight prior to the limit waypoint. This situation revealed an issue that, if uncorrected and deployed to the Alaska Airlines aircraft environment, could have led to the occasional loss of valuable opportunities to evaluate TAP.

An example of this issue is presented in Fig. 5. The figure shows the map portion of the TAP Display, with the direction of travel of the aircraft (current ownship position is off-screen) from the upper-right to the lower left of the image. The TAP-generated trajectory (dashed cyan line) rejoins the active route (solid magenta line) at the limit waypoint, DEQUE. An area of active SUAs (polygons) are shown, each with an undepicted lateral buffer. In this example, the undepicted lateral buffers of the SUAs overlapped the active route causing a conflict (polygons highlighted in yellow). The TAP software could not provide a trajectory solution resolving the conflict because it was after the limit waypoint.

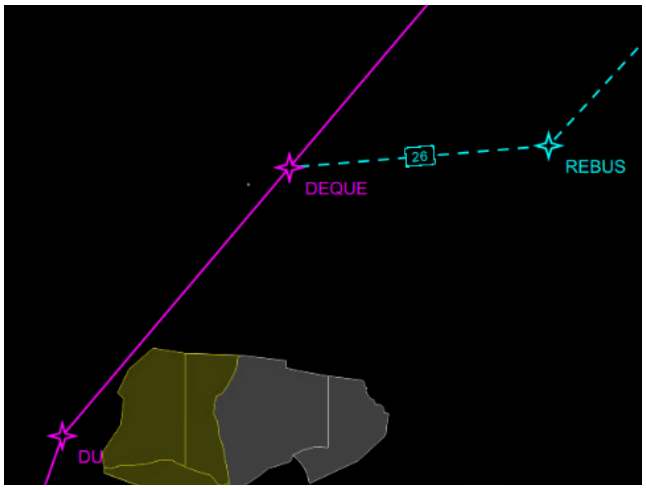

Fig. 5 SUA Lateral Buffer Conflict shown on TAP Display.

The test environments described in this paper were used to verify several TAP software updates that mitigated portions of this issue. The first mitigation was to remove the lateral buffer around the SUAs. The lateral buffer provided minimal benefit from a conflict resolution viewpoint, and created a potential source of confusion due to improper depiction in the TAP Display. Secondly, because the TAP software was designed for en route operations, an altitude filter was applied that ignored SUAs with ceilings below FL180. Finally, the SUA look-ahead time, which previously considered all conflicts with SUAs regardless of activation time, was set to 180 minutes.

The interactive TAP playback tool was utilized in VM environments to test the impacts of removing the SUA lateral buffer, adding the altitude filter, and modifying the SUA look-ahead time in updated software builds of TAP, using flights where this issue occurred. The Simulation Test Lab was used to verify iteratively these mitigations before they were evaluated on the HU-25A, which enabled the delivery of the TAP software on time and increased confidence that the software performed as intended.

B. Design, Development, and Testing of Semi-Automated Cruise Settings

The second case study examined the design and development of algorithms to track and autonomously set the cruise altitude and cruise speed parameters in the TAP Display. These parameters are used by TAP's trajectory generator to compute trajectory solutions. These values typically change in flight, but are not available to the TAP software via the ARINC 834 STAP feed. Therefore, the flight crew had to manually update the cruise altitude and cruise speed setting on the TAP Display anytime those parameters changed.

The TAP software monitors the current aircraft state against these values and illuminates an indicator light (see Fig. 6) to notify the flight crew if the settings in TAP are misaligned with the actual aircraft state data. The indicator light is: green (a) if the cruise parameters and the state data are the same, yellow (b) if the aircraft is in level steady flight but not at the altitude or airspeed entered in the TAP display, or grey (c) if the aircraft is in a climb or descent. Incorrect settings for either cruise altitude or cruise speed number lead to inaccurate predictions of fuel and time outcomes, thus causing inaccurate and imprecise optimization solutions. Furthermore, feedback from SMEs during flight-testing on the HU-25A indicated the pilot's workload due to maintaining the TAP cruise altitude and speed settings would be unacceptable in an operational evaluation.

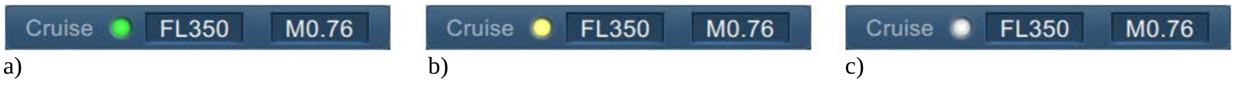

Fig. 6 TAP Display Cruise Settings Panel.

Two algorithms were designed, implemented, and tested to allow the TAP Display to autonomously track, infer, and set the cruise parameters — one for the cruise altitude and one for the cruise speed. Each of these algorithms operate in a continuous loop once the aircraft is above a configurable altitude.

Once these algorithms were implemented in the TAP Display software, initial testing focused on level cruise flight using the VM test environment. These initial tests were structured to identify gaps in the logic and set baseline parameters used to tune the algorithms. Testing then progressed to the HU-25A, where the tests centered on identifying operational issues with the implementation of the logic, such as precision errors in cruise setting calculations or undesired trends during aircraft maneuvers.

Testing in the HU-25A environment uncovered a logic gap and unintended behavior in the first version of the cruise altitude algorithm during the initial climb-out. If a user set the correct target altitude during the initial climb-out (Fig. 7, line with open squares), the automation caused the cruise altitude setting in the TAP software (Fig. 7, line with filled circles) to descend from the user setting in 1000-foot increments until it matched the aircraft's barometric altitude (Fig. 7, gray line) while it was still climbing. When the altitude setting in the TAP software reached the altitude of the climbing aircraft, the logic increased the altitude in 1000-foot increments to remain aligned with the aircraft altitude. This behavior also manifested during step climbs or descents at cruise altitude, especially if the altitude change was greater than 4,000 feet. Therefore, the logic was modified to prevent the algorithm from changing the cruise altitude setting in cases where the barometric altitude was converging towards the target cruise altitude.

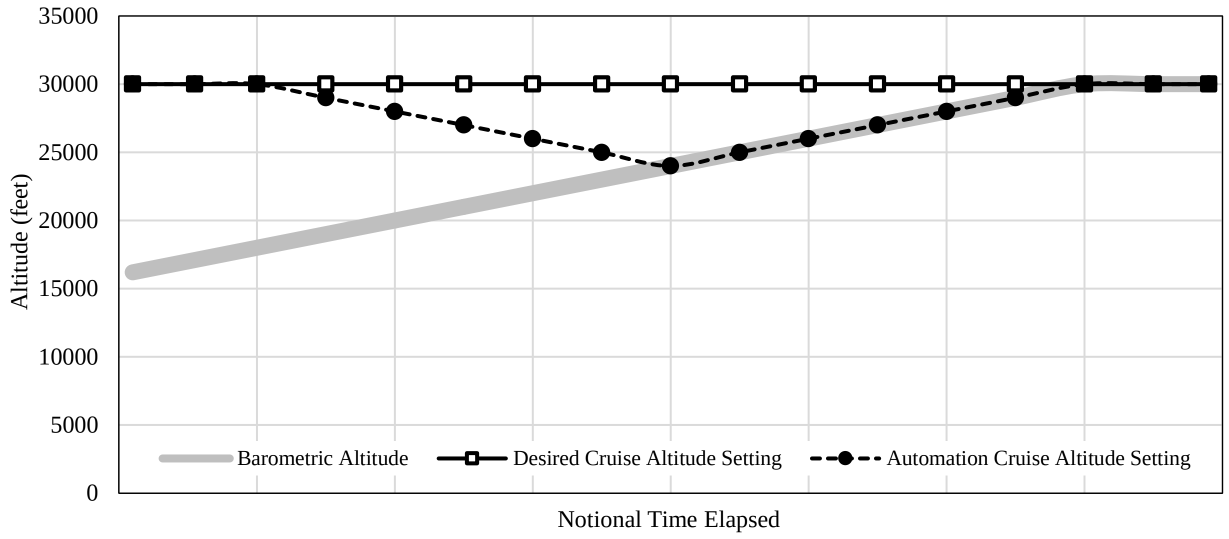

Fig. 7 Notional Visualization of the Performance of Cruise Altitude Logic during Initial Climb-out.

During a subsequent flight on the HU-25A, a second logic gap was identified in the updated version of the cruise altitude algorithm. On that particular flight, the aircraft was instructed to level off after takeoff at 4,000 feet for approximately 3 minutes before climbing to the desired cruise altitude of FL300. Due to the duration of the intermediate level off, the cruise altitude algorithm correctly inferred that the aircraft was in a steady-state error condition (cruise altitude setting in TAP was FL300 while the aircraft was steady and level at 4,000 feet) and modified the cruise altitude setting in TAP accordingly to 4,000 feet. Additionally, during descent the cruise altitude algorithm correctly inferred that the aircraft's altitude was diverging from the cruise altitude set in TAP, and modified the cruise altitude accordingly by decreasing it in 1,000-foot increments until the TAP application automatically shut down at 10,000 feet. The algorithm performed as designed in both occurrences; however, from an operational perspective, neither behavior was desired. The logic was modified to include a check to determine if the barometric altitude of the aircraft was greater than a configurable altitude parameter before the algorithms would begin to operate. The altitude parameter is currently set to FL300.

After the final version of the cruise altitude and cruise speed algorithms were implemented and tested in each of the developmental test environments, they were delivered to Alaska Airlines. The SMEs who initially raised a concern regarding a potential increase to the pilot's workload due to the manual entry of the cruise settings in TAP were able to exercise the revised software during the Alaska Airlines operational evaluation. They provided positive feedback that the new functionality was acceptable and corrected the deficiency, that is, the correct cruise settings only had to be set once which resolved the workload concern and reduced a potential source of pilot error. In addition to correcting the deficiency, the SMEs felt the revised algorithm increased confidence in the calculated fuel and time savings since the trajectory predictions now used the correct aircraft cruise altitude and speed values.

V.Conclusion

This paper presented the operational and developmental test environments used to conduct simulations and flight tests as part of the TAP software evaluation with Alaska Airlines. Test environments included Alaska Airlines aircraft, a NASA flight test aircraft, a Simulation Test Lab, Virtual Machine environments, and TAP playback tools. Background information on TAP system components inherent to these test environments was provided, and two case studies that illustrated the impact the various test environments had on the success of testing efforts were described.

Convenient access to each purpose-built environment at different points throughout software testing was vital to TAP software development, namely the identification and mitigation of software and operational issues prior to higher fidelity deployment. Simulation test environments enabled convenient and rapid turnaround testing of the TAP software, while the NASA HU-25A flight test environment enabled purposeful replication of the operational test environment for higher fidelity testing and technical acceptance of the TAP software. These test environments were able to reduce technical and programmatic risk, culminating in a successful deployment of the TAP software to Alaska Airlines.

References

[1] Henderson, J., Idris, H., and Wing, D. A., “Preliminary Benefits Assessment of Traffic Aware Strategic Aircrew Requests (TASAR)”, AIAA 2012-5684, AIAA Aviation Technology, Integration, and Operations Conference; Indianapolis, IN, 2012.

[2] Ballin, M. G. and Wing, D. J., “Traffic Aware Strategic Aircrew Requests (TASAR)”, AIAA-2012-5623, AIAA Aviation Technology, Integration, and Operations Conference; Indianapolis, IN, 2012.

[3] Woods, S. E., Vivona, R. A., Roscoe, D. A., LeFebvre, B. C., Wing, D. J., and Ballin, M. G., “A Cockpit-based Application for Traffic Aware Trajectory Optimization”, AIAA 2013-4967, AIAA Guidance, Navigation, and Control Conference; Boston, MA, 2013.

[4] Woods, S. E., Vivona, R. A., Henderson, Jeffrey H., “Traffic Aware Planner for Cockpit-based Trajectory Optimization” AIAA-2016-4067, AIAA Aviation Technology, Integration, and Operations Conference; Washington, DC, 2016.

[5] Palmer, Michael T., Ballin, Mark G., “A High-Performance Simulated On-board Architecture to Support Traffic Operations Research”, AIAA-2003-5452, AIAA Modeling and Simulation Technologies Conference; Austin, TX, 2003.

[6] Finkelsztein, Daniel, et al., “Airspace and Traffic Operations Simulation for Distributed ATM Research and Development”, AIAA-2005-6488, AIAA Modeling and Simulation Technologies Conference and Exhibit, San Francisco, CA, 2005.

[7] Lewis, T. A., Burke, K. A., Underwood, M. C., and Wing, D. J., “Weather Design Considerations for the TASAR Traffic Aware Planner”, AIAA Aviation Technology, Integration, and Operations Conference; Dallas, TX, 2019.

[8] Wing, D. J., Burke, K. A., Henderson, J., Vivona, R. A., and Woodward, J., “Initial Implementation and Operational Use of TASAR in Alaska Airlines Flight Operations”, AIAA 2018-3043, Aviation Technology, Integration, and Operations Conference, AIAA Aviation Forum; Atlanta, GA, 2018.

[9] J. Henderson, “Annualized TASAR Benefit Estimate for Alaska Airlines Operations”, NASA/CR-2015-218787, Hampton, 2015.

[10] Underwood, M. C., Lewis, T. A., Barney, T. L., “In-Flight Evaluation of the Traffic Aware Planner on the NASA HU-25A Guardian Aircraft”, AIAA Aviation Technology, Integration, and Operations Conference; Dallas, TX, 2019.

[11] Wing, D. J., Burke, K. A., Ballard, K., Wilson, S. R., Henderson, J., Woodward, J., “Initial TASAR Operations Onboard Alaska Airlines”, AIAA Aviation Technology, Integration, and Operations Conference; Dallas, TX, 2019.