CHAP-E: A Plan Execution Assistant for Pilots

J. Benton and **David Smith** and **John Kaneshige** and **Leslie Keely**

NASA Ames Research Center
Moffet Field, California 94035-1000
{j.benton,david.smith,john.t.kaneshige,leslie.keely}@nasa.gov

Abstract

Pilots have benefited from ever-increasing and evolving automation techniques for many decades. This automation has allowed pilots to handle increasingly complex aircraft with greater safety, precision, and reduced workload. Unfortunately, it can also lead to misunderstandings and loss of situational awareness. In the face of malfunctions or unexpected events, pilots sometimes have an unclear picture of the situation and what to do next, or must find and follow written procedures that do not take into account all the details of the particular situation. Pilots may also incorrectly assume the mode or state of an automated system and fail to perform certain necessary actions that they assumed an automated system would handle. To help alleviate these issues, we introduce the Cockpit Hierarchical Activity Planning and Execution (CHAP-E) system. CHAP-E provides pilots with guidance toward executing procedures based on the aircraft and automation system's state and assists pilots through both nominal and off-nominal flight situations.

Introduction

Piloting aircraft requires handling input from a variety of systems, including instruments that inform a pilot of the aircraft's state (e.g., airspeed, vertical speed, altitude, attitude, and heading). While automation has a long history of assisting pilots with handling this information, when malfunctions occur, sometimes multiple messages come from distinct systems, confusing a pilot and making it difficult to understand the next best course of action. A sad example of this occurred during the Air France 447 flight, which crashed, killing all passengers and crew. Coming from Rio de Janeiro, Brazil and going to Paris, France, the flight entered a large area of thunderstorm activity that resulted in both turbulence and ice crystals forming in the pitot tubes, which measure airspeed. Though the anti-ice system came on and a warning sounded, the pitot tubes iced over and no longer provided correct airspeed, causing the autopilot and auto-thrust systems to disengage. The aircraft began to roll from the turbulence, and the pilot overcompensated because the aircraft was now in a control mode that was more sensitive to roll input. Through a series of often disparate warnings and incorrect assumptions that followed (including stall warnings and presumed assumptions that the aircraft's autopilot would not allow a high angle of attack)[1], the aircraft stalled at 38000 feet, plunging into the ocean three and a half minutes later.

This tragic incident serves as an illustrative example of how messages from disparate systems, unclear procedures, and lack of basic data regarding the aircraft's automation state can cause serious issues for pilots and their flights. Numerous other examples exist, including American Airlines 268 and Turkish Airways 1951. Indeed, 55% of all major incidents are due to system malfunctions, and a primary reason those contributed to bad outcomes related to pilots' inability to accurately assess the nature of the failure (FAA 2013). Our objective is to help flight crews by providing a global picture of expected procedures given the aircraft state. Toward these ends, we seek to provide pilots with procedural guidance during flight, keeping track of the aircraft state and providing suggested procedures for pilots to follow.

More specifically, we are interested in the problem of real-time monitoring of all phases of airliner flight, and providing feedback to the pilots when actions are overlooked or are inappropriate, or when the conditions of flight are no longer in accordance with the objectives or clearance.[2] Traditionally, pilots have made use of written or electronic checklists to verify that appropriate actions have been performed and that the aircraft reached the proper state for each particular phase of flight. While these have served to standardize procedures and ensure that critical items have not been overlooked, checklists are both static and passive. For example, the pre-landing checklist confirms that the flaps are at the landing setting, the landing gear is down, the airspeed is in an appropriate range for the landing weight, the approach is stabilized, and the autobrakes are armed. It does not tell the pilots when to lower flaps, when to lower landing gear, what modes and settings to select for the autopilot, or whether the selected landing speed and flap settings are even appropriate for the runway length, wind conditions, and current runway braking action. In other words, the checklist helps confirm the state of the aircraft, but provides no guidance about when or how to achieve that state. This information is

[1] The angle of attack is the angle between the wing and airflow.

[2] This includes things like speed, altitude, descent/climb rate, autopilot mode and settings, route compliance, flap settings, fuel state, etc.

all buried in the pilot's training and expertise, and in procedures in the Pilot's Operating Handbook (POH). However, details can get overlooked when the crew is fatigued, the crew is overworked (e.g., due to weather conditions), or in the event of system failures.

To tackle this problem, we introduce the Cockpit Hierarchical Activity Planning and Execution (CHAP-E) system, a decision support and procedure display system. CHAP-E can display pre-defined plans (procedures) or interface with a situation-aware automated planner to generate and display appropriate procedures. In this way, it can be viewed as a planning, monitoring and execution assistant for the aircraft flight.

This paper focuses on the CHAP-E display and its use during flight. We designed the CHAP-E display to account for the constant movement that occurs during flight. Unlike some domains, during flight the state of the world changes continuously but predictably over time, depending upon action execution. This enables CHAP-E to determine how to safely execute a plan. The state information includes events such as reaching waypoints, instrument data (e.g., altitude and air speed) and aircraft configuration (e.g., flaps and landing gear positions). To predict these, CHAP-E uses an external simulator called the Trajectory Prediction System (TPS) (Kaneshige et al. 2014). It can use these predictions to determine the earliest time when a pilot may begin actions in the plan and the latest time an action can be executed for the plan to remain successful.

We begin by discussing work related to automation during flight and plan execution displays. We then follow with an overview of the CHAP-E system. Finally, we end with a discussion on future work.

Related Work

Automation systems have a long history in aviation. As (Billings 1996) points out, making flight more resistant and tolerant to error stands as the primary purpose of automation assistance in aviation. Despite this, little work has been done in assisting pilots by displaying procedures to them, ensuring their applicability during flight, and monitoring the execution of those procedures. Perhaps the closest match to CHAP-E is MITRE's Digital Copilot, which is designed for smaller single-pilot aircraft and informs the pilot of common mistakes and constraint violations during flight (MITRE 2016).

Other work on procedure displays has been implemented within the context of space missions. The NASA Autonomous Mission Operations (AMO) project used a user interface to track a spacecraft's life support system activity (Frank et al. 2015). It offered recommended activities based on the state of the spacecraft and current operating constraints. Personnel on the spacecraft forwarded the recommendation to flight controllers, and if approved by the flight controllers, the activity would be scheduled and displayed. It also used a system that helped the crew track the progress of plan execution. That system, called WebPD, was integrated with spacecraft systems to provide information about the spacecraft's state. The system then displayed serial procedures along with important relevant state information related to each step in the procedure (Stetson et al. 2015; Frank et al. 2013). A similar system, called the Procedure Integrated Development Environment (PRIDE), was implemented to assist in the development of procedures. The procedures are stored using the Procedure Representation Language (PRL). The procedure view component, PRIDE View, allows a user to follow a procedure step-by-step (Kortenkamp et al. 2008).

Another comparable system is RADAR (Vadlamudi et al. 2016), which assists in producing plans by generating landmarks and offering action suggestions based upon them. Unlike the other systems listed, RADAR uses PDDL (Fox and Long 2003).

Plan Execution Assistant

The design of CHAP-E centers around reducing human error and its potentially negative effects by providing decision support to human pilots. We define human error as action or lack of action taken by a human with unintended effects. Without knowing the intentions of a human actor, we cannot determine whether an error has taken place. A human must share the intention of their actions to identify an error. This makes detecting human error a difficult, complex problem. In our current version of CHAP-E we do not expect to identify all human errors. Instead, we assume a human pilot never intends to cause goal failure or violate important safety and operational constraints, which ties human intent to pilots maintaining safe flight. Fortunately, we can focus on negative effects assuming that a human will want to avoid making errors that would cause potential hazards or cause failure to achieve a given goal.

In this section, we present an overview of the CHAP-E system currently in development, with a particular focus on the approach and landing phases of flight. First, we discuss the CHAP-E plan representation, then how we can determine whether a plan may succeed without violating constraints. Finally, we discuss the CHAP-E display and its operation.

Plan Representation

CHAP-E uses hierarchical plans with causal links. The primitive actions in the plan can be simulated to ensure they do not violate important constraints. As seen in Figure 1, in the plan hierarchy, the highest level task is a flight from the departure airport to the destination airport, flight(from, to). This expands into a serial (via causal links) set of subtasks: ⟨FileFlightPlan(from, to), ObtainClearance(from, to), Taxi(rnwy) Fly(from, to), Taxi(gate), Shutdown⟩.[3] We can further break down the Fly(from, to) action into the phases of flight: ⟨Takeoff(from, rnwy), Climb, Cruise, Descend, Approach, Land(rnwy)⟩. Expanding the Approach phase, we have a set of primitive actions taken by the pilot. These individual actions have enabling safety conditions associated with them, such as a particular segment on the approach, an airspeed range, or an altitude range.

[3]We simplify the example by removing some parameters and tasks.

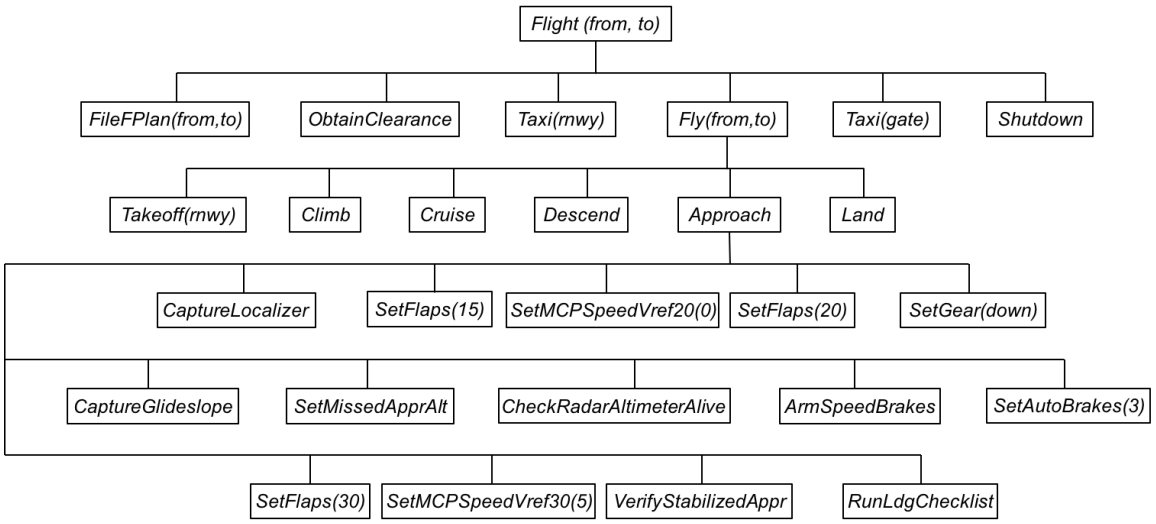

Figure 1: CHAP-E hierarchical plan with the approach phase expanded down to primitive actions

The hierarchical structure provides several advantages. First, much of the expansion cannot take place initially, because some of the parameters and constraints are not yet known. For example, we cannot always initially expand Taxi(rnwy), because the taxi route and the runway may have not been assigned yet. Before this expansion can take place, the initial part of the plan must be executed – we must file the flight plan, and obtain the taxi clearance. Similarly, the Departure, Cruise and Descent activities of the Fly subtask cannot be expanded until the aircraft obtains a route clearance. The Approach and Landing activities usually cannot be expanded until later in the flight when the approach and runway are assigned by Air Traffic Control (ATC) and accepted by the pilots. This will often depend on the traffic and weather conditions at the time of arrival. For example, the current wind conditions usually dictate which runways are viable, and ceiling and visibility constrain the approaches that are possible. This necessitates interleaving the hierarchical expansion of the plan and the plan's execution.

One of the challenges of representing these plans is that many of the actions are keyed off of particular events, rather than times. For example, a standard practice is to lower flaps to 20 degrees and lower the landing gear just before intercepting the glideslope (the vertical guidance for the aircraft) on an approach. Typically, this happens just outside of the Final Approach Fix, a designated waypoint about 5nm (nautical miles) from the end of the runway. The trouble is, there is some uncertainty about the exact time at which this event will occur, since it depends on the aircraft's exact speed and altitude, and on the wind conditions; if the aircraft is a bit faster than expected or a bit high, this event will occur sooner, if the headwinds are higher than expected, this event will occur a bit later. As a result, many of the actions in a CHAP-E plan are triggered off of events, rather than times or the completion of preceding actions. Frequently, these events involve reaching a particular waypoint or distance from a waypoint, reaching a particular altitude, or reaching a particular airspeed.

Figure 2 shows the plan and profile views for the ILS 28R approach into San Francisco.[4] The plan view gives a map-like picture of the approach as seen from above, with a transition to the approach starting at the waypoint ARCHI, intercepting the final approach course at the waypoint DUMBA, and continuing through the Final Approach Fix, the waypoint AXMUL, to the runway. The profile view shows a vertical slice of the altitude profile for the final segments of the approach.

Figure 3 shows a small fragment of a detailed plan for intercepting and flying this approach, for an aircraft beginning just east of the ARCHI waypoint. The plan contains three types of statements: *Events* that are expected to occur, *Actions* that the pilots must perform, and *Monitors*, which indicate conditions that must be maintained throughout some interval. Each event is characterized by a label, followed by the event. For example, the first event is that of crossing the waypoint ARCHI on the transition to the approach. The events ZILED, GIRRR, DUMBA, CEPIN, AXMUL, and RW28R also refer to the crossing of waypoints. The next five events are prefaced by `before!` conditions, which indicate a hard constraint that the event must occur before another event (otherwise the plan becomes invalid). The first of these is that we must have the clearance for the approach from ATC before crossing the ARCHI waypoint. The next three refer to the airspeed dropping below the maximum allowed value for a particular flap setting. The final two refer to the autopilot capturing the localizer and glideslope – the lateral and vertical guidance for the approach. Finally, A1500 and A1000 refer to the events when the altitude becomes less than 1500 ft and 1000 ft above the runway.

Actions are much like events, but these are things the pilots must do. These usually contain both hard and soft con-

[4]Note that waypoints follow a standard naming convention of five all-capitalized letters.

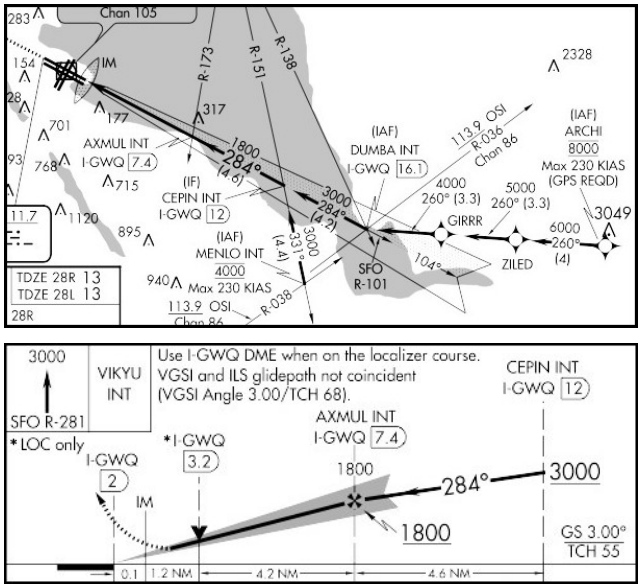

Figure 2: Plan and profile views for the ILS 28R approach into San Francisco

straints (preferences). For example, the first action says that after the clearance event, and before the GIRRR waypoint the pilots must arm the localizer in the autopilot, which allows the autopilot to follow the lateral guidance. This is a hard constraint, as indicated by the exclamation point (!) an the end of `between!`. There is also a soft constraint (preference) that this happen between ARCHI and ZILED (no exclamation point). The second action is similar, with a hard constraint window, and a soft constraint that the action happen before CEPIN. The third action is also prefaced by both hard and soft constraints and specifies a sequence of two events: setting the flaps to 20, and setting the autopilot speed window to the value Vref20. Like events, actions can have names, and the first of this sequence is named F20, which the subsequent action is conditioned on. The fourth action, lowering the landing gear, has a hard constraint that it must be performed before altitude 1500 and a soft constraint to do it after setting the flaps to 20, and before AXMUL. The final action sequence has a hard constraint and two soft constraints. This is needed in this case because we do not know, a priori, which of the events, Gear or AXMUL, will occur first, and we prefer that the action be done after both events have occurred.

Monitors are conditions that must hold over some period of time. For example, the first monitor states that the airspeed must always remain between the reference speed and the maximum speed for the particular flap setting being used at the time. The second and third monitors state that the localizer and glideslope must remain captured, and the final monitor states that the flaps must remain in the landing configuration. If any of these conditions are violated, the plan becomes invalid and must be revised.

There are a couple things worth noting about this plan:

- Most actions are conditioned on events, rather than on other actions. In many cases actions indirectly control when these events will occur, but there is some uncertainty, and it is the events that serve as constraints on when to perform the actions.
- Traditionally, "flexible" plans have been used to help deal with duration and time uncertainty. In a flexible plan, actions are partially ordered, and may be restricted to designated time windows. That is true here also, but the flexibility is expressed in terms of events that will manifest in terms of time windows. So, for example, the first action specifies that the localizer should be armed after the event where the clearance is obtained, and between the events of crossing ARCHI and GIRRR. These events will ultimately prescribe a time window for the action at execution time.
- Monitors are like overall conditions or durative goal conditions. They specify conditions that must hold between particular events. However, these monitors often span several low level actions. As a result, they are associated with (and come from) higher level tasks in the hierarchy. In the example above, they are associated with the Approach task instantiated with the San Francisco ILS 28R.
- The inclusion of both hard and soft constraint windows (discussed later) on actions is particularly important for

```
Events {
  ARCHI: cross(ARCHI) ;
  ZILED: cross(ZILED) ;
  GIRRR: cross(GIRRR) ;
  DUMBA: cross(DUMBA) ;
  CEPIN: cross(CEPIN) ;
  AXMUL: cross(AXMUL) ;
  RW28R: cross(RW28R) ;
  before![ARCHI] {CLR: start(Clearance = ILS28R.ARCHI)} ;
  before![ARCHI] { F5max: start(IAS <= Vmax5)} ;
  before![CEPIN] {F20max: start(IAS <= Vmax20)} ;
  before![AXMUL] {F30max: start(IAS <= Vmax30)} ;
  before![DUMBA] {LocCap: start(FMA-Lateral = LOC)} ;
  before![AXMUL] {GSCap: start(FMA-Vertical = GS)} ;
  A1500: start[Alt <= 1500AGL) ;
  A1000: start[Alt <= 1000AGL) ;
  ... }
  Actions {
  between![CLR,GRRR] & between[ARCHI,ZILED] <<ArmLocalizer>> ;
  between![LocCap,AXMUL] & before[CEPIN] <<ArmGlideslope>> ;
  between![F20max,AXMUL] & between[CEPIN,GSCap]
            <<F20: SetFlaps(20),SetMCPSpeed(Vref20)>> ;
  before![A1500] & between[F20,AXMUL] <<Gear: SetGear(Down)>> ;
  between![F30max,A1000] & after[Gear] & between[AXMUL,A1000]
            <<SetFlaps(30), SetMCPSpeed(Vref30+5)>> ;
  ... }
Monitors {
  throughout[CEDES, RW28R] {IAS in [Vref,Vmax]} ;
  throughout[LocCap, RW28R] {FMA-Lateral = LOC} ;
  throughout[GSCap, RW28R] {FMA-Vertical = GS} ;
  throughout[F30, RW28R] {Flaps = 30} ;
  ... }
```

Figure 3: Fragment of a detailed CHAP-E plan for the ILS 28R approach into San Francisco

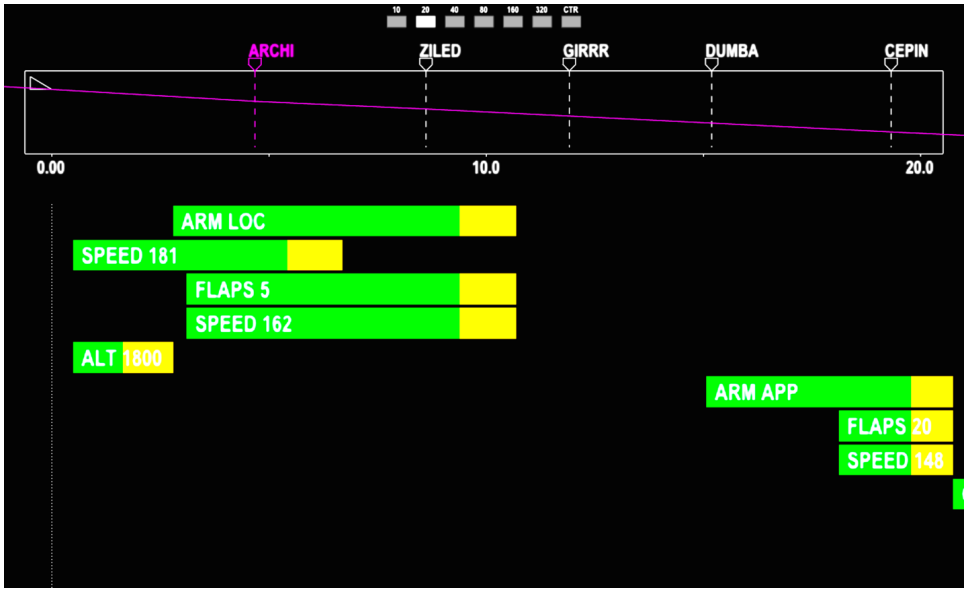

Figure 4: The CHAP-E display

monitoring pilots. Sometimes actions like lowering the landing gear could be performed much earlier, but it would be inefficient and noisy to do so. Standard procedure is to do it just before the final approach fix. The preferences therefore provide a more restrictive window where the actions should be performed, and allow us to warn the pilots when this does not happen by the end of the preference window. This allows for reasonable alerting, without becoming annoying for the pilots.

The plan fragment in Figure 3 contains approximately $1/4$ of the events, actions and monitors necessary for this particular example approach. The PLEXIL executive has been able to use the complete version of this plan to successfully approach and land a 777 at San Francisco in simulation, as well as to monitor pilot actions and warn when actions are not taken within the preference windows.

CHAP-E Display

The purpose of the CHAP-E display is to provide suggested flight procedures to a pilot for maintaining safe flight (see Figure 4). It consists primarily of a vertical profile and waypoint display, showing the expected vertical profile of the aircraft. Below that are the actions to be executed. Each action has an associated time window, representing when the pilot should perform it. The display moves horizontally as time passes.

Vertical Profile and Waypoints The CHAP-E vertical profile display includes waypoint information from the flight plan to provide a reference for the pilots. The profile shows the reference altitude of the aircraft given the current route. The aircraft is depicted as a small triangle at the upper left of this profile. Labels above the vertical profile show waypoint locations that the aircraft will reach when following the current flight plan. These provide a reference for the pilots; the labels will match instructions for airport-specific flight procedures, and may be referenced by air traffic control requests. They also help give scale for when a pilot should execute each action.

Actions and Time Windows Generally, a pilot should be allowed flexibility on when to execute actions in the plan. This means we depend on the pilots' training and habits so they may determine risk and action priority. To accommodate this aim, we display actions to the pilot in a gantt chart-like style indicating time windows for when the pilot should execute them. An example of an execution window is shown in Figure 5. Each window consists of five time points: an earliest start time (EST), preferred earliest start time (PEST), preferred start time (PST), preferred latest start time (PLST), and latest start time (LST). The display shows the preferred earliest start time, preferred latest start time and latest start time. The two end points, the earliest start time and latest start time, represent the interval in which the action may execute and the plan will still be valid. If executed outside of these times, the plan will likely fail. The preferred earliest and latest start times show when we prefer the pilot perform the action. The preferred start time is when we would ideally perform the action in a fully automated system. The execution times are given in terms of time relative to reaching waypoints during the flight plan and displayed in a gantt chart style.

To obtain the earliest and latest start times, CHAP-E can simulate the execution of the plan across varying start times for each action using an advanced simulation capable of capturing the physics and expected autopilot modes of the aircraft (see Figure 6 for a visual depiction of simulation) (Kaneshige et al. 2014). The simulation takes a series of commands and the time at which we expect the pilot to execute the commands, where each command cor-

responds to a pilot action in the plan. From this, it returns a per-second discretized profile of the state of the aircraft over the course of the displayed period. Using operational constraints, CHAP-E can determine whether the execution schedule would result in a safe flight and achievement of the goal (i.e., landing safely on a specified runway).

As indicated above, the display only shows the preferred earliest start time, the preferred latest start time and the latest start time. As discussed earlier, many actions, such as lowering the landing gear, can be performed very early without causing a plan to become invalid. Displaying the earliest start time may clutter the display with many long overlapping action boxes. To avoid that, we instead rely on preferred start times inferred from standard operating procedures, which usually suggest particular times for action execution. We use these to infer the preferred time points, and display the preferred earliest start time. The latest start time is more critical, so we display it. The preferred latest start time represents when we warn the pilot if an action has not yet been performed. For finding the preferred time points, we rely on domain knowledge for now, though we may explore other options to determine them automatically.

To show these time points, we use standard coloring often seen on flight displays. CHAP-E draws a green window between the preferred earliest start time and the preferred latest start time and an amber window between the preferred latest start time and the latest start time. If the preferred latest start time passes, a warning is spoken by CHAP-E with the action name (e.g., "gear down") and a status message is displayed at the bottom of the CHAP-E display. If the latest start time point passes, then a new plan will be displayed to recover.[5] When an action is executed by the pilot, it will be removed from the display, and the displayed action time windows below it will adjust their positions by moving up.

For the earliest and latest start times, we currently are using hard-coded time windows, but are implementing an initial technique for using the simulation to find them automatically. Our initial approach is a hill-climbing method, where we will first begin with a working schedule. Then, for each action, CHAP-E will simulate executing that action an arbitrary ϵ amount later. If the simulated trajectory continues to remain within specified operational limits and reach the goal, then the process repeats with another ϵ increase. Once a non-compliant simulated execution is found, the process begins with that action again simulating execution ϵ sooner than the original time point, and again repeating this process until a non-compliant simulated execution is found. The process then moves to the next action until all actions in the displayed plan can be simulated. In the case that not all actions are known due to lack of plan expansion, we will simulate as much into the future as possible. Note that this process is deterministic in nature and does not take into account potential exogenous events. It also treats the actions as being independent for purposes of computing the earliest and latest start times. In general this assumption is not valid. Performing an action like lowering flaps increases drag and causes the aircraft to slow down, which may change the time windows for subsequent actions and events triggered off of aircraft speed. As a result, when such an action is performed, subsequent windows may expand, shrink, or shift.

[5]Note that at the time of this writing, we are developing the planning mechanism for this step.

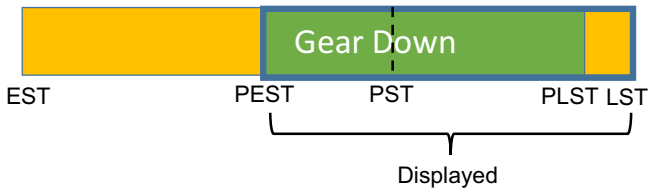

Figure 5: Time windows of an example "gear down" action, showing the earliest start time (EST), preferred earliest start time (PEST), preferred start time (PST), preferred latest start time (PLST), and latest start time (LST)

When an action is performed, CHAP-E must recognize this and remove the action from the display. This is more complex than it might first appear. For example, if the recommended action is to set the MCP-Speed to 163 knots, but the crew instead sets the speed to 165 knots, CHAP-E must determine whether this "unexpected" action still satisfies the necessary conditions for future actions that were to be achieved by the "recommended" action. If so, CHAP-E can remove the recommended action from the display. If not, CHAP-E will leave the recommended action(s), but must determine whether the unexpected action interferes with any conditions that need to be preserved in order for the plan to remain valid.

Challenges exist in continuously determining execution windows. The state of the flight continuously changes as the aircraft progresses. This means the time windows can change as unpredicted adjustments or pilot action (or inaction) occur. Though the simulation is relatively fast (approximately 60 milliseconds for a 5 minute projection), it may be impractical to rediscover time windows continuously. We're exploring several possibilities to mitigate this issue, including using courser-grained hill-climbing until important events occur (e.g., unexpected or missed pilot actions).

Discussion and Future Work

On commercial airliners, two pilots work in tandem during flight. One pilot, the *pilot flying* (PF), performs most of the direct piloting operations, including interacting with the aircraft controls. The other pilot, the *pilot monitoring* (PM), handles communication, runs checklists, and monitors the aircraft state, occasionally taking requests from the PF to perform certain actions on the aircraft. When the PF misses performing an action or fails to notice a significant change to the aircraft state, the PM notifies the PF.

Though CHAP-E initially has focused on assisting human pilots with plan execution via a display, we also have been exploring adding stronger automation characteristics to the system so it can handle both PF and PM tasks given a current plan. We plan to explore methods for the system to switch between PM and PF tasks. Currently, we can auto-

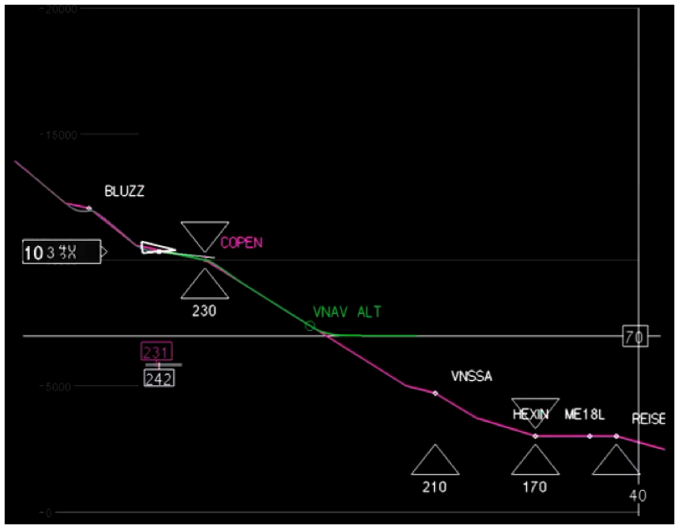

Figure 6: Visual depiction of plan execution simulation

mate flight completely by using the preferred start time as the time to execute actions. However, we require a mechanism for transferring flight control to CHAP-E for particular actions. This may happen automatically (for example, if the pilots appear to have missed performing a critical action) or upon request (for example, if the pilots become occupied and wish CHAP-E to perform specific actions). Our goal is to make the process intuitive to a pilot. We have begun considering using a touch-screen interface on CHAP-E to tap on actions that the pilots hope CHAP-E will perform, as well as methods of automatically determining safety-critical actions that may require CHAP-E's intervention when a pilot fails to perform them.

Acknowledgements

This work was supported by NASA Aeronautics's SASO/SECAT program.

References

Billings, C. E. 1996. Human-centered aviation automation: Principles and guidelines. Technical Report 110381, NASA Ames Research Center.

FAA. 2013. Operational use of flight path management systems.

Fox, M., and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20:61–124.

Frank, J. D.; Spirkovska, L.; McCann, R.; Wang, L.; Pohlkamp, K.; and Morin, L. 2013. Autonomous mission operations. In *IEEE Aerospace Conference*.

Frank, J. D.; Iverson, D.; Knight, C.; Narasimhan, S.; Swanson, K.; Scott, M. S.; Pohlkamp, K. M.; Mauldin, J. M.; McGuire, K.; and Moses, H. 2015. Demonstrating autonomous mission operations onboard the international space station. In *AIAA SPACE 2015 Conference and Exposition*.

Kaneshige, J.; Benavides, J. V.; Sharma, S.; Panda, R.; and Steglinski, M. 2014. Implementation of a trajectory prediction function for trajectory based operations. In *AIAA Atmospheric Flight Mechanics Conference*.

Kortenkamp, D.; Bonasso, R. P.; Schreckenghost, D.; Dalal, K. M.; Verma, V.; and Wang, L. 2008. A procedure representation language for human space flight operations. In *9th International Symposium on Artifical Intelligence, Robotics and Automation for Space i-SAIRAS*.

MITRE. 2016. The solo pilot gets a second set of eyes. https://www.mitre.org/publications/project-stories/the-solo-pilot-gets-a-second-set-of-eyes.

Stetson, H. K.; Frank, J. D.; Haddock, A.; Cornelius, R.; Wang, L.; and Garner, L. 2015. AMO EXPRESS: A command and control experiment for crew autonomy. In *AIAA SPACE 2015 Conference and Exposition*.

Vadlamudi, S.; Chakraborti, T.; Zhang, Y.; and Kambhampati, S. 2016. Proactive decision support using automated planning. Technical report, Arizona State University. https://arxiv.org/abs/1606.07841.