To appear in ICAPS 2018

CHAP-E: A Plan Execution Assistant for Pilots

J. Benton[*] and **David Smith**[**] and **John Kaneshige** [*] and **Leslie Keely**[*] and **Thomas Stucky**[*,+]
[*]NASA Ames Research Center, Intelligent Systems Division
{j.benton, john.t.kaneshige, leslie.keely}@nasa.gov

[**]david.smith@PSresearch.xyz

[+]SETI Institute
thomas.stucky@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 the 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 intuitive graphical guidance on what actions need to be performed and when they need to be performed based on the aircraft and automation state, and projection of this state into the future. This assists pilots in 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 Air France flight 447, which crashed into the Atlantic in 2009, killing all passengers and crew. Enroute from Rio de Janeiro to Paris, 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, causing them to malfunction. Though the anti-ice system came on and a warning sounded, the erroneous airspeed readings caused the autopilot and autothrottle systems to disengage. The aircraft began to roll from the turbulence, and the pilot overcompensated because the aircraft entered a control mode that made it more sensitive to roll input. Through a series of often disparate warnings and incorrect assumptions that followed (including stall warnings and presumptions 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 difficulties for pilots. Numerous other examples exist, including American Airlines flight 268 and Turkish Airways flight 1951. Indeed, 55% of all major incidents involve system malfunctions, and a primary reason those malfunctions contributed to bad outcomes was the pilots' inability to accurately assess the nature of the failure (FAA 2013) and determine an appropriate course of action. Our objective is to help flight crews by providing a global picture of expected procedures given the aircraft state. Towards this end, 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 flight from takeoff to landing, 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. This includes monitoring and providing guidance for things like speed, altitude, descent/climb rate, autopilot mode and settings, route compliance, flap settings, and fuel state. Traditionally, pilots have made use of written or electronic checklists to verify that appropriate actions have been performed and that the aircraft is in the proper state for the next phase of flight. While these ensure that critical items have not been overlooked, checklists are both static and passive. For example, the pre-landing checklist confirms the landing setting for flaps, the landing gear's state, the airspeed range for the landing weight, whether the approach is stabilized, and the autobrakes' state. 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 appropriate for the runway length, wind conditions, and current runway braking action. In other words, the checklist helps confirm

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

the state of the aircraft, but provides no guidance about when or how to achieve that state. This information is all buried in the pilot's training and expertise, and in procedures in the Pilot's Operating Handbook (POH). However, the crew can easily overlook details when fatigued or overworked due to adverse weather or 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 pilot procedures (plans) or interface with a situation-aware automated planner to generate and display appropriate procedures. It can be viewed as a planning, monitoring and execution assistant for aircraft flight.

Unlike many domains, during flight, the state of the world changes continuously over time. While this change is fairly predictable, it is also highly non-linear. For example, when the pilots set a lower airspeed in the autopilot, the throttles are retarded and the aircraft gradually slows down. However, this depends on how rapidly the aircraft descends and may occur at a variable rate. It depends on the drag deployed (flaps, landing gear, spoilers), the wind conditions, and the details on how the autopilot functions. To predict this continuous change, CHAP-E uses an external simulator called the Trajectory Prediction System (TPS) (Kaneshige et al. 2014). It repeatedly calls TPS to sample the range of possibilities and displays time windows to the pilots – the earliest time when the pilots should begin each action in the plan and the latest time each action can be executed for the plan to remain successful. Windows may have dependencies on one another; once CHAP-E fixes a window, the windows for remaining actions can shift, expand, or shrink. Uncertainty about wind conditions, aircraft dynamics, and autopilot response can also cause these windows to change.

This paper focuses on the CHAP-E system and its use during flight. Our main contributions in this work include: 1) characterizing the challenges related to planning in this domain, 2) the development of a plan language that facilitates the hierarchical encoding of event-driven plans and requirements, 3) use of a fast simulator to do real-time adjustment of the time windows for future activities, and 4) an intuitive graphical interface that assists pilots in recognizing what actions should be performed and the appropriate intervals for those actions.

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 for space missions. The NASA Autonomous Mission Operations (AMO) project tracked 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 encoded 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.

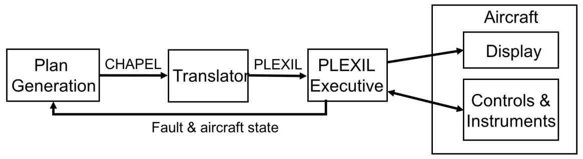


Figure 1: The architecture of CHAP-E

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 the goals implied by the flight plan and clearance.

The high-level architecture of CHAP-E is shown in Figure 1. CHAP-E plans are generated as CHAPEL, discussed below. To monitor plan execution and execute plans, we use the Plan Execution Interchange Language (PLEXIL) (Verma et al. 2006). A translator converts the CHAPEL plans into PLEXIL plans. The PLEXIL is read by a universal executive (UE) to interface with the aircraft. The PLEXIL executive can perform two tasks: plan monitoring and plan execution. Plan monitoring involves ensuring that the pilot executes the plan actions and prompting if actions are not initiated in a timely fashion, or requirements are violated. Plan execution automatically executes plan actions as an automated co-pilot. The CHAP-E display shows the plan as it is being executed. Currently, it is possible to manually switch between the two PLEXIL modes of execution.

Plan Representation

CHAP-E uses hierarchical plans to characterize the different tasks and primitive actions in an aircraft flight. Currently the hierarchies are handcrafted based on the phases of flight. This helps us represent well-defined conditions and requirements for each phase. As seen in Figure 2, the highest level task is a flight from the departure airport to the destination airport, flight(from, to). This expands into a sequence of subtasks: ⟨FileFlightPlan(from, to), ObtainClearance(from, to), Taxi(rnwy) Fly(from, to), Taxi(gate), Shutdown⟩.[2] 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, such as setting speeds for the autopilot, setting flaps, lowering landing gear, and running checklists.

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 wind conditions usually dictate which runways are active, and ceiling and visibility constrain the approaches that are possible. This necessitates interleaving the hierarchical expansion of the plan and the plan's execution.

Hierarchical structure has been commonly used in many planning applications. Typically, a decomposition for a task consists of partially or totally ordered sets of subtasks or primitive actions. In some hierarchical planning approaches, metric temporal constraints between subtasks/actions are also allowed. At worst, these constraints among subtasks/actions can be captured using a Simple Temporal Network. However, for CHAP-E the nature of the constraints among subtasks is more complex. In particular, many of the actions are keyed off of particular events, rather than times. For example, during an approach, the 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). 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 or reaching an altitude airspeed.

A second source of complexity is that for a plan to remain valid, conditions must often hold or be maintained between events. If these conditions are ever violated, additional contingent actions must be performed. For example, during a particular segment of a descent or approach, the aircraft speed might need to remain in a particular range. However, this may not be possible because of the rate of descent required. Adding additional drag, in the form of speedbrakes (spoilers), can be used as a contingency action to rectify the constraint violation and keep the aircraft speed within the desired window.

Figure 3 shows a procedure from a Continental 777 flight manual for flying a CAT III ILS instrument approach procedure. Figure 5 shows a small fragment of CHAP-E's detailed plan decomposition for flying this kind of approach instantiated to the ILS 28R approach into San Francisco International Airport. Figure 4 shows the published approach procedure for this approach, consisting of a plan and profile view. 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.[3] The profile view shows a vertical slice of the altitude profile for the final segments of the approach.

The plan fragment in Figure 5 for intercepting and flying this approach assumes the aircraft begins just east of the ARCHI waypoint. The plan contains four types of statements: *Events* that are expected to occur, *Actions* that the pilots must perform, *Requirements*, which indicate conditions that must be maintained throughout some interval, and *Contingencies*, which indicate corrective actions if requirements are not met. 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, 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 two 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

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

[3]Waypoints are always five capital letters.

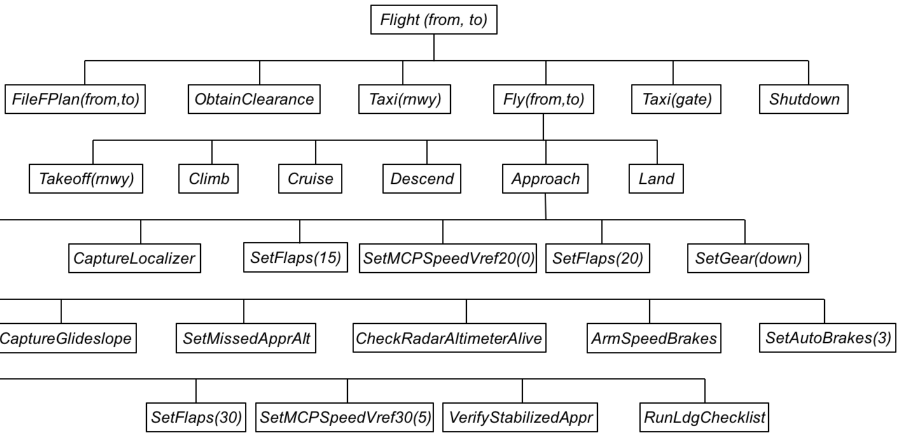
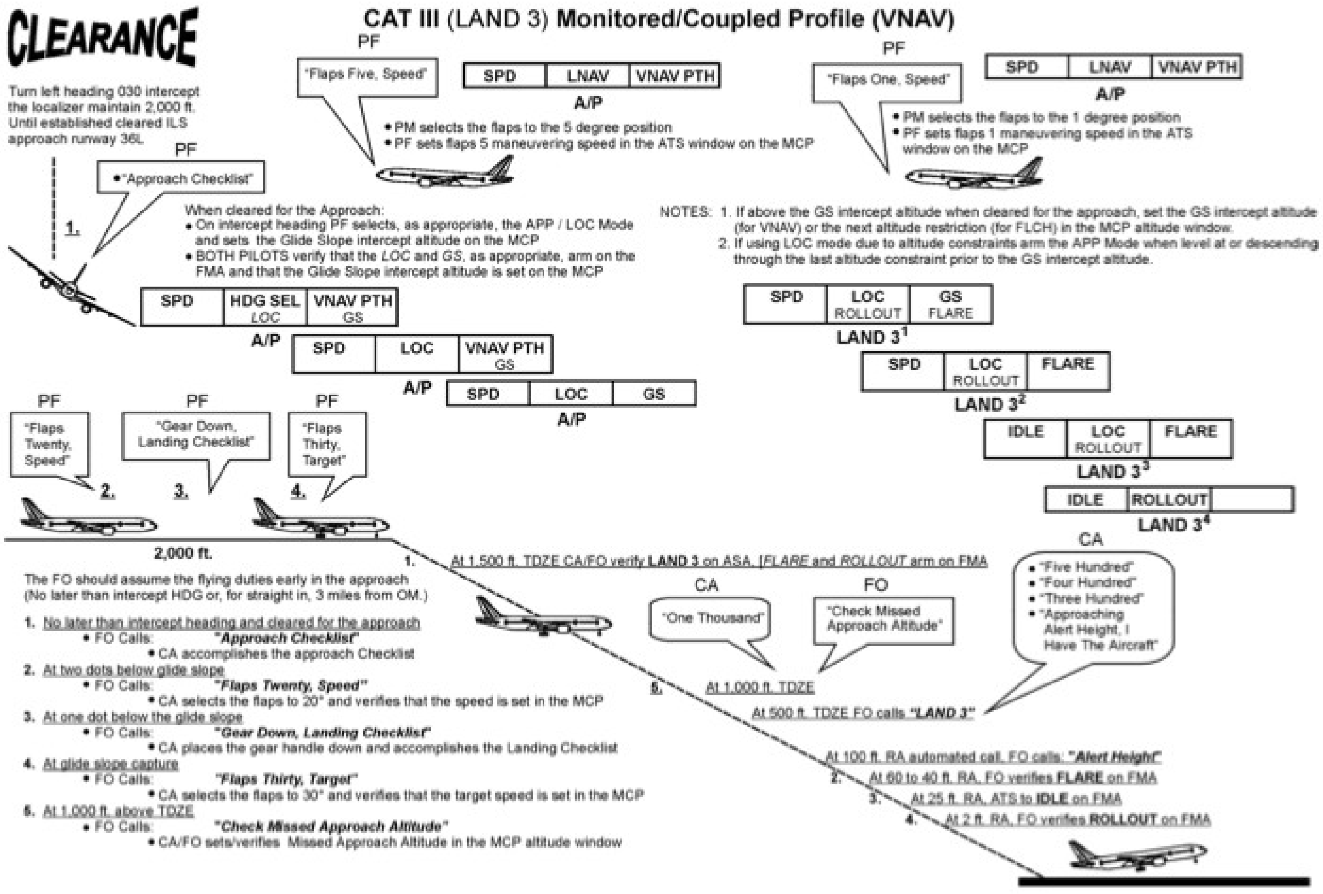


Figure 2: CHAP-E hierarchical plan with the approach phase expanded down to primitive actions



Turn left heading 030 intercept the localizer maintain 2,000 ft. Until established cleared ILS approach runway 36L

- PM selects the flaps to the 5 degree position
- PF sets flaps 5 maneuvering speed in the ATS window on the MCP

- PM selects the flaps to the 1 degree position
- PF sets flaps 1 maneuvering speed in the ATS window on the MCP

When cleared for the Approach:
- On intercept heading PF selects, as appropriate, the APP / LOC Mode and sets the Glide Slope intercept altitude on the MCP
- BOTH PILOTS verify that the *LOC* and *GS*, as appropriate, arm on the FMA and that the Glide Slope intercept altitude is set on the MCP

NOTES: 1. If above the GS intercept altitude when cleared for the approach, set the GS intercept altitude (for VNAV) or the next altitude restriction (for FLCH) in the MCP altitude window.
2. If using LOC mode due to altitude constraints arm the APP Mode when level at or descending through the last altitude constraint prior to the GS intercept altitude.

The FO should assume the flying duties early in the approach (No later than intercept HDG or, for straight in, 3 miles from OM.)

1. No later than intercept heading and cleared for the approach
   - FO Calls: ***"Approach Checklist"***
     - CA accomplishes the approach Checklist
2. At two dots below glide slope
   - FO Calls: ***"Flaps Twenty, Speed"***
     - CA selects the flaps to 20° and verifies that the speed is set in the MCP
3. At one dot below the glide slope
   - FO Calls: ***"Gear Down, Landing Checklist"***
     - CA places the gear handle down and accomplishes the Landing Checklist
4. At glide slope capture
   - FO Calls: ***"Flaps Thirty, Target"***
     - CA selects the flaps to 30° and verifies that the target speed is set in the MCP
5. At 1,000 ft. above TDZE
   - FO Calls: ***"Check Missed Approach Altitude"***
     - CA/FO sets/verifies Missed Approach Altitude in the MCP altitude window

1. At 1,500 ft. TDZE CA/FO verify **LAND 3** on ASA, [*FLARE* and *ROLLOUT* arm on FMA

5. At 1,000 ft. TDZE

At 500 ft. TDZE FO calls ***"LAND 3"***

At 100 ft. RA automated call, FO calls: ***"Alert Height"***

2. At 60 to 40 ft. RA, FO verifies **FLARE** on FMA
3. At 25 ft. RA, ATS to **IDLE** on FMA
4. At 2 ft. RA, FO verifies **ROLLOUT** on FMA

Figure 3: Procedure for flying an ILS CAT III approach from a Continental 777 flight manual

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 constraints (preferences indicated by CHAPEL constraints *without* a `!` character). For example, the first action says that after the clearance event, and before the CEPIN waypoint the pilots must arm the localizer in the autopilot, which allows the autopilot to capture and follow the lateral guidance. This is a hard constraint, as indicated by the exclamation point (`!`) at the end of `between!`. There is also a soft constraint (preference) that this happen between ZILED and CEPIN (no exclamation point) and a further ideal preference indicating that CHAP-E would choose to do this action at ZILED. The second action is similar, with a hard constraint window, and a soft constraint that the action happen before CEPIN, with an ideal time of LocCap (when a guidance signal called the localizer is received). 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 action in the next line 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 again has a hard constraint, a soft constraint and an ideal time.

Requirements are conditions that must hold over some period of time. For example, the first requirement 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 requirements state that the localizer and glideslope must remain captured, and the final requirement states that the flaps must remain in the landing configuration over the specified interval.

The Contingencies provide a simple fix (action) if the first requirement is violated. This prevents the plan from becoming invalid. In situations like that of Air France 447, discussed in the introduction, a contingency would involve warning pilots that their actions (e.g., by putting the aircraft at a high angle of attack) have put jeopardized the flight and to inform them of corrective measures.

The plan fragment in Figure 5 contains approximately 20% of the events, actions, requirements and contingencies needed for this particular example approach. The PLEXIL executive has been able to use the complete version of this plan to successfully intercept, 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. A more general hierarchical version of this plan has been used to fly transitions and approaches to other airports and runways.

Planning for CHAP-E

We would like CHAP-E to automatically provide procedures for pilots for as many situations as possible so it can become a dependable piloting aid for both nominal and off-nominal scenarios. For this reason, we have been actively

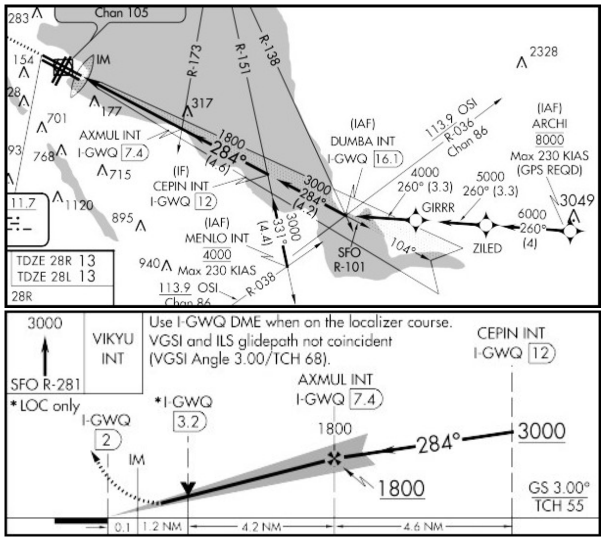


Figure 4: Plan and profile views for the ILS 28R approach into San Francisco

working toward applying automated planning techniques to generate CHAPEL plans automatically and on-the-fly as new situations arise. As it is, the development of CHAP-E has necessitated that we capture standard operating procedures and established air traffic constraints for a variety of circumstances. While capturing these procedures and constraints offers its own set of demands, automatic generation of plans for our domain presents even greater challenges:

- Event-based actions. Most actions are conditioned on events, rather than on other actions. These events may be exogenous and known, but exactly when they will occur is unknown. Though actions may indirectly control when events occur, through varying aircraft speed, for example, uncertainty in environmental conditions and precise pilot actions, prevents off-line timepoint prediction. Traditionally, "flexible" plans have been used to help deal with duration and time uncertainty. Flexible plans have partially ordered actions and may be restricted to designated time windows. We express our flexibility in terms of events that will manifest as time windows.
- Re-establishment contingencies. We allow pilots to perform actions outside of the established plan. In some cases, these actions may have no ill effects on the plan (i.e., the pilot actions maintain the causal link requirements). In some cases, however, pilots might perform actions that clobber a causal link. In these cases, the causal link must be reestablished. In generating plans offline for reestablishment, we need to follow these guidelines: (1) The reestablishment should not require removal of any tasks or requirements in the remainder of the plan, (2) the contingency must be able to reestablish the causal link prior to any action requiring it, (3) the reestablishment of the causal link should not interfere with any of the current tasks. Contingencies have been sometimes discussed in terms of generalized planning (Srivastava, Immerman,

```
Events {
 ARCHI: cross(ARCHI) ;
 ZILED: cross(ZILED) ;
 CEPIN: cross(CEPIN) ;
 AXMUL: cross(AXMUL) ;
 RW28R: cross(RW28R) ;
 before![ARCHI] {CLR: start(Clearance = ILS28R.ARCHI)} ;
 before![CEPIN] {F20max: start(IAS <= Vmax20)} ;
 before![AXMUL] {F30max: start(IAS <= Vmax30)} ;
 before![CEPIN] {LocCap: start(FMA-Lateral = LOC)} ;
 before![AXMUL] {GSCap: start(FMA-Vertical = GS)} ;
 A1500: start[Alt <= 1500AGL) ;
 A1000: start[Alt <= 1000AGL) ;
 ... }
Actions {
 between![CLR,CEPIN] & between[ZILED,CEPIN] & at[ZILED]
           <<ArmLocalizer>> ;
 between![LocCap,AXMUL] & before[CEPIN] & at[LocCap]
           <<ArmGlideslope>> ;
 between![F20max,AXMUL] & between[CEPIN,GSCap] &
  at[CEPIN] <<F20: SetFlaps(20),SetMCPSpeed(Vref20)>> ;
 before![A1500] & between[F20,AXMUL] & at[F20]
           <<Gear: SetGear(Down)>> ;
 between![F30max,A1000] & between[Gear,A1500] &
  at[AXMUL] <<SetFlaps(30), SetMCPSpeed(Vref30+5)>> ;
 ... }
Requirements {
 R1: throughout[START, RW28R] {IAS in [Vref,Vmax]} ;
 throughout[LocCap, RW28R] {FMA-Lateral = LOC} ;
 throughout[GSCap, RW28R] {FMA-Vertical = GS} ;
 throughout[F30, RW28R] {Flaps = 30} ;
 ... }
Contingencies {
 Unless R1 <<Speedbrakes>> ;
 ... }
```

Figure 5: Fragment of a detailed CHAP-E plan for the ILS 28R approach into San Francisco

and Zilberstein 2008). Generalized plans are goal-directed solutions to planning problems that can be applied on any deterministic environment within the domain. Generalized planning has typically focused on including loops in plans, but also involves reasonable contingencies.

- Requirements across several low-level actions. Requirements are like overall conditions or durative goal conditions. They specify conditions that must hold between particular events. However, these requirements often span several low level actions. As a result, they are associated with (and come from) higher level tasks in the hierarchy.
- Continuous, non-linear dynamics. Prediction of the aircraft position, its true airspeed, expected autopilot mode, and a number of other dynamically changing attributes (e.g., fuel levels), requires modeling continuous, nonlinear quantities. Pilot actions control most of these quantities. As an obvious consequence, a planner cannot precisely predict event time without modeling these. This also makes it difficult to predict potential contingencies that might be required. Though the modeling language PDDL+ can encode continuous, nonlinear processes, the language does not allow for task hierarchy (Fox and Long 2006). It also is not easy to encode combinations of preference-based and constraint-based. The planner `PluReal` can handle PDDL+-style nonlinear, continuous change by using a Satisfiability Modulo Theories (SMT) encoding (Bryce 2016). We have considered using encodings such as this for our domain.
- Preference-based and constraint-based event windows. The inclusion of both hard constraint and soft preference windows on actions is particularly important for monitoring pilots. For example, lowering the landing gear could be performed early, but this would be inefficient and noisy. Instead, pilots normally perform these tasks according to standard operating procedures. In the case of landing gear, this should occur just before reaching the final approach fix, the beginning of the last segment of flight when coming into an airport. Preferences provide a more restrictive window to perform actions and allow us to warn pilots when they do not perform actions by the end of the preference window. This allows reasonable alerting without becoming annoying for pilots.

PLEXIL for Plan Execution

The Plan Execution Interchange Language (PLEXIL) was originally developed for spacecraft operations. It operates over a state machine, where each node in the state machine contains the “content” of what can be executed. Nodes can be hierarchical, providing varying levels of abstraction. Content can include other nodes, commands and data lookups. Each node has a set of start conditions, specifying when the execution of an action should begin, and a set of end conditions, specifying when the execution should finish. Nodes can also contain loops, specifying repeat conditions. Pre- and Post-conditions exist on each node to ensure that activities can execute properly.

To execute a plan, PLEXIL monitors exogenous events, e.g., the state of the aircraft or passing over a particular waypoint, to use as triggers for control input. Plan monitoring works by examining the instruments and controls of the aircraft. A plan monitor has the same tasks but operates on them differently. Instead of executing a task by solely examining exogenous events, a plan monitor looks for endogenous events (i.e., actions of the pilot) to inform the pilot when a task passes the preferred latest start time. In this case, it enters a warning node, informing the pilot that the task must be completed. If the pilot does not then perform the task before the latest start time, PLEXIL enters into an error state. In this case, it can either trigger a contingency or trigger a process to find a new procedure.

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 6). It consists primarily of a vertical profile and waypoint display, showing the expected vertical profile of the aircraft. Actions are displayed below that. Each action has an associated time window showing when the pilot should perform it. The display moves horizontally over time.

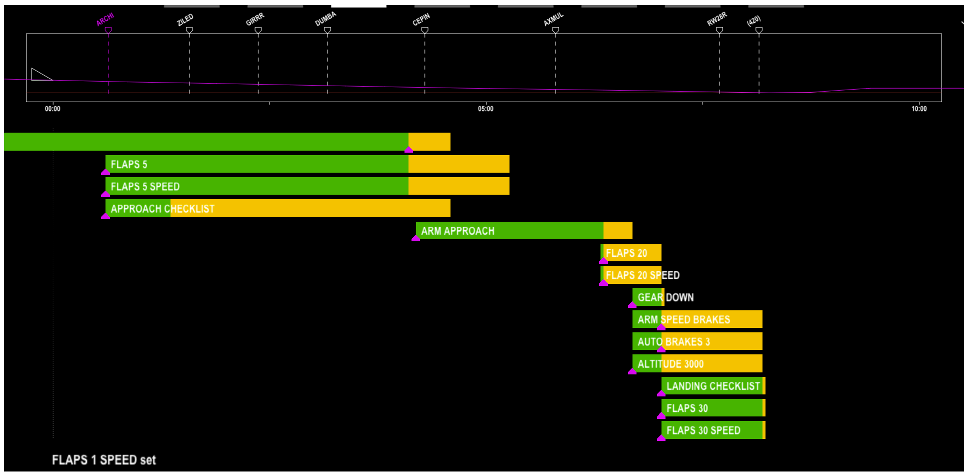


Figure 6: The CHAP-E display

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 and vertical trajectory 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 path. These provide a reference for the pilots; the labels will match waypoints on published departure, enroute, arrival, and approach procedures, and may be referenced in air traffic control clearances. 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 7. 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). Initially each of these corresponds to an established constraint or preference, as defined in the plans we generate (see Figure 5). The display shows the PEST, PST, PLST, and LST. The two end points, the EST and LST, represent the interval in which the action must execute for the plan to remain valid. If executed outside of these times, the plan will likely fail. The PEST and PLST show when we prefer the pilot perform the action. The PST is when we would ideally perform the action in a fully automated system.

Many actions, such as lowering the landing gear, can be performed very early without causing a plan to become invalid.[4] Displaying the EST can clutter the display with many

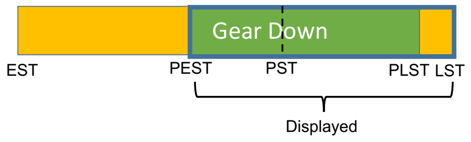


Figure 7: Time windows of an example "gear down" action, showing the earliest start time (EST), preferred earliest start time (PEST), preferred start time (PST), preferred latest start time (PLST), and latest start time (LST)

long overlapping action boxes. To avoid that, we only display PESTs PLSTs, and LSTs. We warn the pilots if an action has not yet been performed by the PLST. For finding the preferred time points, we rely on domain knowledge from standard operating procedures such as that shown in Figure 3. EST and LST points are updated through simulation and sampling as described below.

To show these time points, we use standard coloring often seen on flight displays. CHAP-E draws a green window between the PEST and the PLST and an amber window between the PLST and the LST. A magenta arrow shows the recommended PST. If the PLST 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 contingency actions will be introduced if available, or a new plan will be generated and displayed.[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.

Rescheduling Time Windows To obtain the earliest and latest start times, CHAP-E uses an advanced simulation capable of capturing the physics and expected autopilot modes

[4]In extreme instances where additional drag is necessary to descend quickly, it might be necessary to extend gear early.

[5]This planning mechanism is still under development.

of the aircraft using a tool called the Trajectory Projection System (TPS) (Kaneshige et al. 2014). Figure 8 shows a visual depiction of this simulation with the most recent predicted trajectory of the aircraft shown in green. The green text shows the predicted mode changes, and the white text, though difficult to see, shows the expected actions to be performed by the pilot. The magenta line represents the reference trajectory (the trajectory expected according to the approach and landing procedures).

The TPS provides a fast-time simulation of simplified aircraft dynamics (Kaneshige et al. 2014; Shish et al. 2016). A 4D trajectory (position and time/speed) is extrapolated based on the current state of the aircraft, weather conditions, and simplified models of control laws and mode transition logic heuristics for the flight management system, autopilot system, and autothrottle system. The state of the aircraft is modeled along with commands issued by the flight systems, such as the bank angle, flight-path angle, and thrust. TPS then propagates the state forward in time using a basic forward Euler method to predict the orientation, velocity, and position of the aircraft. For CHAP-E, the trajectory prediction system was enhanced to include support for inputting command changes at specific times (e.g., the `at` preference in the plans) in the future, including input configuration (e.g., flaps, gear settings) or automation mode (e.g., autopilot, autothrottle) settings. The TPS then 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). If it does not, the windows are adjusted by a fixed amount and TPS is re-run on the new schedule. This process is repeated periodically during the flight at a configurable interval (every second in our current system).

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).

Plan Execution When an action is performed, CHAP-E will recognize this and remove the task 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. 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.

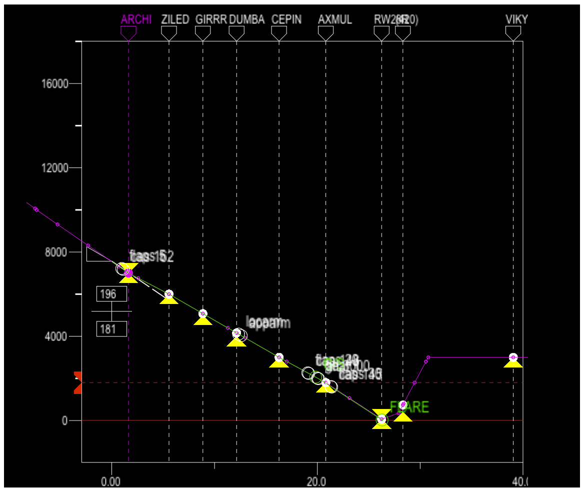


Figure 8: Visual depiction of plan execution simulation

Conclusion & Future Work

In this paper, we presented a pilot assistance tool that displays piloting procedures. The tool, called CHAP-E, gives pilots windows for beginning each task and provides an interval when, according to standard operating procedures, the pilot should execute each task. The end goal of the tool is to provide a pilot quick and accurate procedural guidance during flight. We recognized several qualities of the piloting domain that made planning for piloting guidance difficult. The domain has an event-based nature, where actions must execute relative to events instead of time points, as is typical in temporal planning. Some of the events are exogenous, others are directly or indirectly related to actions taken by the pilot. Plans should also include re-establishment contingencies, where if causal links become violated, a short contingency plan would provide guidance for the pilot to maintain the same procedure. The domain also contains continuous, nonlinear dynamics, and constraint and preference windows. The plans must maintain flexibility so that they can be rescheduled. For future work, we have begun exploring methods of plan synthesis. We discussed the difficulties with performing this task. We are recently interested in performing "pilot-in-a-box" simulations, where CHAP-E can be used to rapidly simulate expected pilot actions for prototyping of aircraft and air traffic scenarios. Toward this goal, we have incorporated an automated planner called the Autonomous Constrained Flight Planner (ACFP) onboard that finds new, alternative flight plans taking inclement weather and other dynamic features and events into account (Sadler et al. 2016). CHAP-E can then execute procedure sets appropriate for the new flight plans.

Acknowledgements

Thank you to Kimberlee Shish for her assistance and discussions. 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.

Bryce, D. 2016. A happening-based encoding for nonlinear PDDL+ planning. In *AAAI Workshop on Planning for Hybrid Systems*.

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

Fox, M., and Long, D. 2006. Modelling mixed discrete-continuous domains for planning. *Journal of Artificial Intelligence Research (JAIR)* 27:235–297.

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.

Sadler, G.; Battiste, H.; Ho, N.; Hoffmann, L.; Johnson, W.; Shively, R.; Lyons, J.; and Smith, D. 2016. Effects of transparency on pilot trust and agreement in the autonomous constrained flight planner. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, 1–9.

Shish, K.; Kaneshige, J.; Acosta, D.; Schuet, S.; Lombaerts, T.; Martin, L.; and Madavan, A. N. 2016. Aircraft mode and energy-state prediction, assessment, and alerting. *Journal of Guidance, Control, and Dynamics* 40(4):804–816.

Srivastava, S.; Immerman, N.; and Zilberstein, S. 2008. Learning generalized plans using abstract counting. In *Proceedings of the Twenty-Third Conference on Artificial Intelligence, AAAI*, 991–997.

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.

Verma, V.; Jonsson, A.; Pasareanu, C.; and Iatauro, M. 2006. Universal-executive and PLEXIL: Engine and language for robust spacecraft control and operations. In *AIAA Space Forum*.