Design Considerations for a Variable Autonomy Executive for UAS in the NAS

Michael Lowry, Anupa Bajwa, Thomas Pressburger, and Adam Sweet
Intelligent Systems Division, NASA Ames Research Center, Moffett Field, CA 94035 USA

Charles Fry, Michael Dalal, and Johann Schumann
SGT, NASA Ames Research Center, Moffett Field, CA 94035 USA

Deborah Dahl
USRA, NASA Ames Research Center, Moffett Field, CA 94035 USA

Gabor Karsai and Nagabhushan Mahadevan
Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN 37235

This paper describes research targeted towards an autonomy executive (AOS) for UAS in the National Air Space (NAS). The project goal is to incrementally provide the knowledge and intelligence onboard a UAS to safely fly in the National Air Space, eventually autonomous from remote human ground crews and communicating directly with air traffic control. Longer-term, the goal is to provide the capability for pilotless air vehicles such as air taxis that will be key for new transportation concepts such as air mobility-on-demand. For both of these targeted applications, AOS is incorporating artificial intelligence capabilities that operationally meet human pilot competencies. Even when autonomy is achieved from a remote human ground crew, AOS will have variable degrees of autonomy with respect to air traffic control (ATC), just as human pilots do now. AOS has the capability of interacting in natural language with ATC, as well as through data link protocols. AOS can adapt to varying levels of autonomy and control directed by ATC in standard and relaxed FAA phraseology– from being vectored moment by moment, to accepting broad directives such as following a specified aircraft or sighting and avoiding traffic. AOS can autonomously manage contingencies such as vehicle systems degradations and failures. It incorporates a decision maker that takes information from multiple diagnostic reasoners, disambiguates (if needed) sensor results to specific failures using active mode changes, then projects forward the impact of the degradation on the nominal plan. If the nominal plan is no longer viable, then alternative plans are formulated, and subsequently selected and executed, including abort options.

I. Nomenclature

DF = a leg type for waypoint to waypoint navigation, part of ARINC 424
HIL = Hardware Integration Laboratory
IVHM = Integrated Vehicle Health Management
MTL = Metric Temporal Logic
NAS = National Air Space
SIL = Software Integration Laboratory
UAS = Unmanned Aircraft System
UTM = UAS Traffic Management
VI = a leg type for intercept to course, part of ARINC 424

II. Introduction

The Autonomy Operating System (AOS) is an open flight software platform for smart UAVs.

AOS has as its foundations NASA's core flight executive and core flight software (cFE/cFS). Core flight software is an open source middleware package that originates in NASA's small spacecraft flight software community, providing the capability for interoperation of reusable apps [1]. In that regard, it is similar to Apple's IOS for smartphones. Typical reusable apps for small spacecraft flight software include command and data handling, thermal management, attitude pointing, and electrical power management. The AOS project has determined that cFE/cFS can be ported to the UAS domain and serve as a robust and certifiable platform for UAS flight software.

A small business, Windhover Labs, is developing a UAS flight software product line based on cFE/cFS suitable for visual and beyond visual line of sight applications such as precision agriculture and package delivery [2]. These applications are anticipated for airspace below 400 feet that will be operated according to UAS Traffic Management (UTM) protocols currently under testing [3]. The research described in this paper is targeted towards an autonomy executive for UAS in the National Air Space, and longer-term towards pilotless air vehicles carrying passengers. This requires more intelligence and piloting capabilities than lower-level drones. In both UTM and UAS in the NAS, piloting capabilities are currently provided by remote ground crews. AOS is aiming for incrementally achieving autonomy from remote ground crews.

Human pilots fly under the regulatory authority of the FAA. Under VFR regulations, a pilot could fly from the west coast of the US to the east coast under 18,000 feet in class G and E airspace, and never communicate with air traffic control. The only communication required would be with other aircraft at un-towered airports. Conversely, a pilot could fly IFR in a manner requiring detailed heading and altitude or waypoint directions from air traffic control at every juncture. In essence the pilot would behave as a low-capability drone that is remote-controlled by air traffic control. The latter mode puts a substantial burden on air traffic controllers, but is sometimes required for separation assurance.

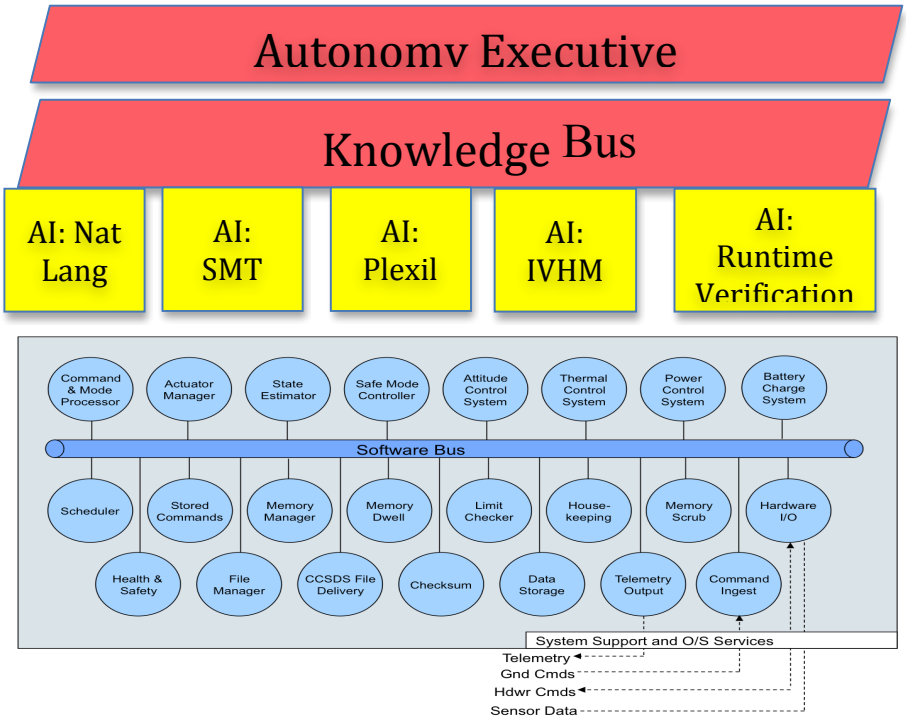
In most flights in the National Air Space, human pilots have a fluid and variable level of both judgmental and execution authority and autonomy with air traffic control. Air traffic control can switch from detailed directions for separation assurance (e.g., a series of heading vectors to the pilot for a portion of the flight plan labeled 'Radar Vectors') to continuing the flight under pilot judgment ("resume own navigation, no traffic between you and the destination airport"). In busy terminal air space in VFR conditions, air traffic control offloads their workload by requesting pilots to maintain self-separation for designated traffic, and provides variable degrees of constraint (e.g., "Follow company on final, maintain 2,000 until the outer marker") and freedom ("Turn base at your discretion") to pilots. For busy terminal air space, there is a factor of two between VFR and IFR operational capacity, due to the increased workload on terminal controllers for maintaining separation.

In order for an artificial intelligent pilot to fly a UAS in the National Air Space without remote human ground crew and without imposing an undue burden on air traffic control, it has to follow the same protocols of variable autonomy as human pilots. Moreover, while data-link is suitable for departure clearances and en-route directives, the rapid pace of terminal operations requires natural language interactions. The current architecture of AOS and its integrated artificial intelligence components is designed to provide the capabilities for an artificial pilot that is capable of flying not only in the relatively uncongested en-route air-space, but also busy terminal air space. This capability will also be needed in the longer-term future for dense autonomy, even with basic air traffic control functions being automated, due to robustness and latency requirements. Speech and natural language understanding will still be needed for any human supervisors in the loop.

III. AOS and Artificial Intelligence

A block diagram of AOS is shown below. The blue box is an example configuration of small spacecraft flight software built on cFS/cFE. The apps are blue circles hanging off the software bus. The core flight executive has scheduling table for the apps, and communication between apps is done on a publish/subscribe basis through typed messages denominated by IDs.

Developers for AOS can integrate directly on the software message bus. (Windhover Labs is developing an autopilot that interfaces directly through the software bus.) AOS has focused on the mission computer and integration of a critical mass of Artificial Intelligence capabilities in order to enable autonomy [4]. The artificial intelligence (AI) elements– in yellow- consist of reasoning components that themselves serve as integration points. The AI elements communicate through the knowledge bus (in red), which is a time-stamped database of assertions in predicate logic. The Autonomy Executive controls the level of autonomy, for example from self-navigation through adherence to low-level directives from ATC.



AOS has integrated with cFE a critical mass of Artificial Intelligence (AI) capabilities including diagnostic reasoning (DR and R2U2) for IVHM, plan and procedure execution (Plexil) [5] automated reasoning engines (Prolog and Z3 [6]), and natural language processing (GATE) [7]. The integration is

through the knowledge bus and software middleware services provided by cFE such as the software message bus and task scheduler. These are *model-based* AI capabilities: a general reasoning engine takes a model and a stream of data to generate output. An AOS AI app can use one or more of these general reasoning engines to accomplish a task such as converting ATC utterances into assertions about actions needed by navigation to comply with clearances and directives. AOS AI apps generally use the knowledge bus to exchange data in the form of assertions. Thus an AOS app developer can either develop a cFE app and interface directly to the software message bus and the task scheduler, or develop AI models that are then executed by one or more AI reasoning engine already interfaced to cFE.

A number of AOS apps are described in this paper providing variable autonomy and variable levels of capability. Variable autonomy is especially important for interfacing with present-day ATC. The levels of capability range from basic execution of nominal and off-nominal procedures defined by the FAA and Pilot Operating Handbook, to deliberative capabilities required for contingency management with trouble shooting and re-planning.

IV. ATC Communication, Natural Language and Variable Autonomy

AOS incorporates both a CPDLC (controller pilot data link communication) and natural language interface capability. The FAA has recently deployed CPDLC at many major airports and an increasing number of enroute sectors. The satcom-based CPLDC works well for oceanic ATC, departure clearance delivery, and enroute amendments – all areas where the high latency of CPDLC match the tolerable time delays for a pilot to acknowledge a clearance. For high density airspace with a requirement for fast pilot acknowledgement and response, natural language will likely persist as the preferred medium of human air traffic controllers. Terminal Airspace is a prominent example of this requirement. An informal survey of terminal controllers indicated that no more than a three second response is expected from pilots – otherwise the controllers need to replan their strategy for orderly sequencing and spacing for landing and departure. From a pilots' perspective, an audio interface to ATC allows them to focus outside the cockpit (VFR) or on their instruments (IFR) during fast-paced terminal operations, rather than "texting while driving". As long as humans are in the loop on either side, natural language will likely be a preferred mode of communication for highly interactive operations. As humans are eventually replaced on both sides, the communication path can be shortened to a digital encoding of the logical assertions that underlie the flow of information for AOS.

In AOS natural language processing is decomposed into an acoustic part (speech to text) and textual processing (text to logical assertions). A series of prototypes has demonstrated that advances in speech processing and methods for text-to-logic are capable of handling FAA phraseology for typical control tower environments or where there are strong radio signals. These prototypes were first built using beta-test platforms provided by Microsoft (LUIS) and Nuance (Nuance Mix) that are principally targeted towards development of webots that interact through speech with end-users over the internet, based on specialized grammars and a training set of utterances. However, for degraded acoustic environments sometimes encountered in aviation where there are weak AM radio signals, further development is needed. A specialized speech-to-text processing engine optimized through machine learning and trained on degraded aviation acoustic environments is being developed for deployment on-board. An alternative solution, which would also considerably help human controllers and pilots, is to move or augment the current aviation AM analog frequency band to digitized radio transmissions – as has already been done by numerous police, fireman, and public safety localities under encouragement of the FCC.

The current AOS implementation of speech-to-text processing is done through Dragon Professional on a stand-alone ground computer. During our drone experiments with simulated air traffic control, the text generated by Dragon Professional is transmitted to the mission computer on the drone. Text to assertion processing is done through the on-board mission computer.

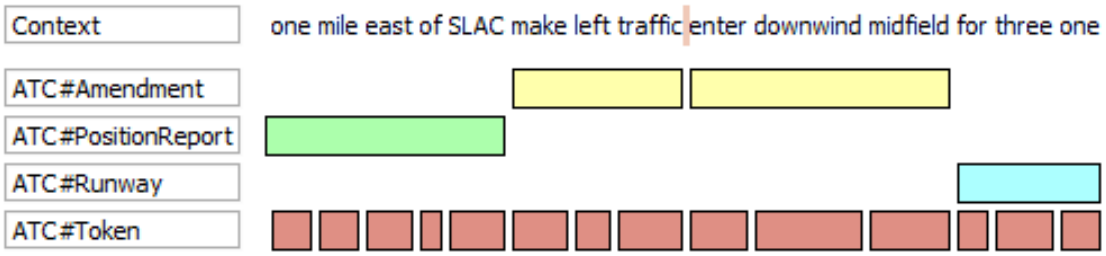
The text to assertion processing is implemented on the open framework GATE through rules specialized to the air traffic domain. In GATE, a directed graph is generated from a sequence of tokens (e.g., words or word parts). Rules then generate annotations on the graph, these annotations are encoded as new links on the directed graph. New rules are then triggered; the effect is to add layers to the graph. For our specialized domain of air traffic control, the layers are:

1) Words (tagged with parts of speech)

2) Semantic lookup identifying basic concepts (e.g., compass heading)
3) Identification of syntactic grouping (e.g., noun phrases, prepositional phrases)
4) Identification of basic ATC concepts (e.g., runway, waypoint, leg, aircraft)
5) Grouping concepts into an ATC message such as an amendment.

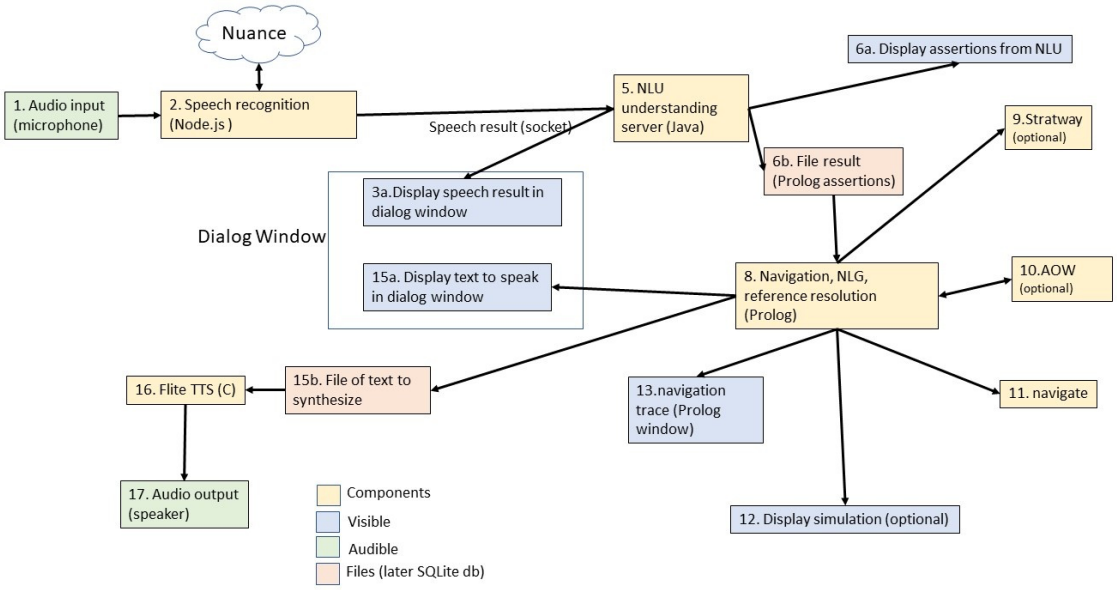
The amendment layer in GATE is then translated into logical assertions for subsequent reasoning by apps like Navigation. An example is shown in the figure below, with the original token string on the line labeled 'Context', and layers 4 and 5 shown below. The substring *'enter downwind midfield'* after processing is translated into the assertion:

amendment(id(amendment 131), aircraft_id(n007jb),
enter_at_landmark(leg(downwind),fix(midfield)))



This bottom-up approach to grammars and text processing has proven robust, and is tolerant of word error rates in the range of 2% to 4% in producing correct interpretations. It works well with standard FAA phraseology and minor deviations to FAA phraseology.

The figure below depicts the current AOS natural language processing chain and the interactive navigation app, in the configuration used for our flight tests with drones - the experimenters providing the simulated air traffic control dialogue. When an air traffic controller speaks to an AOS UAV, the natural language processing generates assertions that are then interpreted by the reasoning component within the navigation app. The result can be to update the flight path, engage routines to search for traffic, engage logic to follow other aircraft in sequence (adjusting speed as needed), and to provide information to air traffic control. AOS reads back clearances to air traffic control, reports back implicitly requested events (e.g., 'traffic in sight' or 'no joy on the traffic'), and makes natural language requests to air traffic control – for example, requesting a landing clearance. The result is a mixture of ATC originated and AOS originated interactions. The resulting flight paths range from flight paths that are tightly controlled by ATC (e.g., "turn heading 300, turn downwind, I will call your base, slow to 70 knots, turn base now") to flight paths whose broad outline is guided by ATC but involve onboard decision making and discretion (e.g., "turn crosswind at your discretion, you are following a Cessna midfield downwind, make room for two departures"). Stratway and AOW are capabilities for detecting and self-spacing from other traffic, built on DAIDALUS [8]. They originated as detect and avoid capabilities, our implementation is based on simulated traffic with ADS-B sensing.



A small portion of the logic (in Prolog) for updating flight paths is shown excerpted below. The logic spans the range from the assertions that are generated by text processing, to representations of the route that is flown by the autopilot. The leg segments of a route are represented by the FAA's path and terminator concept [9], which is also the basis for the ARINC 424 databases that encode instrument procedures for Flight Management Systems. The first part of the logic provides the facts for determining the headings for the legs of a typical VFR traffic pattern, based on the two-digit designator for a runway. For example, the downwind leg for runway 31 is deduced to have heading 130. The subsequent portion provides the basis for deducing an intercept angle from an ATC utterance. An intercept-heading to leg is of type **VI** (vector to intercept). As an example, if an air traffic controller stated an amendment "intercept the downwind on the right 45" the logic would deduce that the intercept angle was offset by 45 degrees from the runway heading, thus the intercept heading was 85 degrees magnetic. The route is amended with two legs – a VI intercept leg with heading 85, followed by **TF** leg completing the downwind from the intercept point.

```
runway_heading(runway(RunwayName), Heading) :- Heading is 10 * RunwayName.
leg_offset(downwind,_,180).
leg_offset(base,left,90).
angle_offset(HeadingIn, right, Angle, HeadingOut) :- !,
  X is HeadingIn - Angle, normalize_heading(X, HeadingOut).

tp_leg_heading(Runway, Leg_Name, Side, Heading) :-
     runway_heading(Runway, RunwayHeading),
   leg_offset(Leg_Name, Side, Offset),
   X is RunwayHeading + Offset,
   normalize_heading(X, Heading).
```

```
leg_entered_at_heading(leg(entry,df(_Wp)),leg(Leg,tf(Wp3,Wp4)),
                       ApproachHeading,
                       leg(entry,vi(ApproachHeading)),leg(Leg,tf(Wp3,Wp4)))
```

The reasoning is set up as forward chaining from assertions that are input to the navigation app from sources including natural language and CPDLC. To limit the search and forward chaining generation, a minor amount of added control is needed to be embedded in the logic– such as the cut operator (!) in the angle offset clause. Otherwise the assertions and the logic clauses are interpreted as standard horn clause logic.

V. Contingency Management: Decision Maker Autonomy

Human pilots are trained to close the loop on contingencies including vehicle system failures, weather, lost communication, control surface malfunctions, and many others. AOS has a basic capability described in sub-section A to flexibly execute both pre-defined normal and emergency procedures as defined by the FAA and Pilot Operating Handbooks through Plexil - a plan execution language and runtime system. AOS also has an advanced capability described in sub-section B to generate new plans and procedures through a deliberative process.

Below is a representative excerpt from the FAA Private Pilot Practical Test Standard for Emergency Operations for which a pilot candidate is expected to demonstrate both knowledge and capability, during the oral and flight portions of the exam. At a minimum, a private pilot candidate is expected to appropriately execute rote-memorized emergency procedures requiring immediate response, and then execute checklists for procedures from the Pilot Operating Handbook. Basic pre-defined procedure execution through Plexil is described in sub-section A. As a human pilot progresses from the private pilot through instrument and commercial pilot exams to the Airline Transport Pilot exam, contingency management is a primary area of increasing depth and breadth of capability that needs to be mastered. The pilot is expected to have increasing depth of knowledge of vehicle systems - their function and fault behaviors, and the ability and flexibility to deliberatively manage contingencies – even if not explicitly delineated in the Pilot Operating Handbook – and to replan the forward path and if necessary "land as soon as practical" or ditch immediately. A capable pilot is also able to actively trouble-shoot in the face of ambiguity, cross-checking a vehicle's response to find the cause of a problem and subsequently mitigate through re-configuration. This more advanced capability is described in sub-section B.

IX. Emergency Operations

Task	*C. Systems and Equipment Malfunction*
References	FAA-H-8083-2, FAA-H-8083-3; POH/AFM
Objective	To determine that the applicant exhibits satisfactory knowledge, risk management, and skills associated with system and equipment malfunctions appropriate to the airplane provided for the practical test and analyzing the situation and take appropriate action for simulated emergencies.
Knowledge	The applicant demonstrates understanding of:
PA.IX.C.K1	Partial or complete power loss related to the specific powerplant, including:
PA.IX.C.K1a	a. Engine roughness or overheat
PA.IX.C.K1b	b. Carburetor or induction icing
PA.IX.C.K1c	c. Loss of oil pressure
PA.IX.C.K1d	d. Fuel starvation
PA.IX.C.K2	System and equipment malfunctions specific to the airplane, including:
PA.IX.C.K2a	a. Electrical malfunction
PA.IX.C.K2b	b. Vacuum/pressure, and associated flight instruments malfunction
PA.IX.C.K2c	c. Pitot/static system malfunction
PA.IX.C.K2d	d. Electronic flight deck display malfunction
PA.IX.C.K2e	e. Landing gear or flap malfunction
PA.IX.C.K2f	f. Inoperative trim
PA.IX.C.K3	Smoke, fire, engine compartment fire.
PA.IX.C.K4	Any other system specific to the airplane (e.g., supplemental oxygen, deicing).
PA.IX.C.K5	Inadvertent door or window opening.
Risk Management	The applicant demonstrates the ability to identify, assess and mitigate risks, encompassing:
PA.IX.C.R1	Failure to use the proper checklist for a system or equipment malfunction.
PA.IX.C.R2	Distractions, loss of situational awareness, and/or improper task management.
Skills	The applicant demonstrates the ability to:
PA.IX.C.S1	Describe appropriate action for simulated emergencies specified by the evaluator from at least three of the elements or sub-elements listed in the K1 through K5 above.
PA.IX.C.S2	Complete the appropriate checklist.

A. Basic Procedure Execution by AOS

An example of a pre-defined emergency procedure is the IFR Lost Communication Procedure [10]. The inner loop of the procedure is excerpted below from the Aeronautical Information Manual, and directs a pilot as to the route segments and altitudes to fly subsequent to losing communication link with ATC.

Route.
(1) Last ATC clearance received;
(2) If being radar vectored, by the direct route to fix specified in vector clearance
(3) Route expected in a further clearance; or
(4) In the absence of an assigned route or a route that ATC has advised may be expected in a further clearance by the route filed in the flight plan.

Altitude. At the HIGHEST of the following altitudes or flight levels FOR THE ROUTE SEGMENT BEING FLOWN:
(1) The altitude in the last ATC clearance received;
(2) The minimum altitude for IFR operations; or
(3) The altitude or flight level ATC has advised may be expected in a further clearance.

Instrument pilot candidates need to demonstrate both the knowledge and capability to perform this procedure to pass the instrument practical test standards. The FAA IFR Lost Communication Procedure differs from lost link software on today's UAS. Current UAS have pre-programmed lost link procedures at takeoff, that do not take into account the history of interactions with ATC. The interactions with ATC are handled by human ground crew, outside the control loop of current UAS. A typical lost link procedure is to circle up to a predefined altitude, attempting to re-establish radio link, and if unable then to fly to a pre-designated location. This type of lost link procedure can cause considerable havoc if a UAS were to fly in congested airspace – requiring clearing out other traffic over a wide swath of airspace as the UAS circles up by itself without input from a human ground crew. In contrast, the FAA IFR Lost Communication Procedure designates how an aircraft should behave after losing communication with ATC – taking into account amendments and directives provided by ATC until the time of lost communication. ATC is careful and is required to provide pilots the information they need to safely continue a flight after loss of communication through directives, incorporated into departure clearances and subsequent communications, such as:

"Expect further climb to 5,000 ten minutes after departure"

"Expect further clearance to TRACY after clearance limit"

During normal operations these expectation directives help pilots plan forward, but would not be executed by a pilot until a normal clearance is subsequently provided, such as "Climb to 5,000 feet". But in the event of lost communications, the expectation directives in effect become clearances and are executed by the pilot at the time or triggering event they are told to expect. The IFR Lost Communication Procedure combines the flight plan, ATC directives including expectations, and map navigation information to provide a pilot with a sequence of waypoints and altitudes that will safely and predictably guide an aircraft to the destination airport causing minimum interference with other aircraft. The path is most consistent with ATC plans up until the time of lost communication, requiring minimal additional de-confliction with other aircraft.

A transcript of AOS executing a departure clearance out of Palo Alto airport, losing communication with ATC, and then subsequently executing the IFR Lost Communication Procedure is excerpted below. Note that the departure clearance includes instructions for a vector clearance (RV) to **SJC** and an expected climb to 5,000 feet. This scenario was tested both in the SIL and also in simulated airspace with a drone – the mission computer was operating the full flight software stack. The elements of the transcript are number by time, the black lines are internal assertions generated by AOS, and the red lines are utterances communicated over the radio.

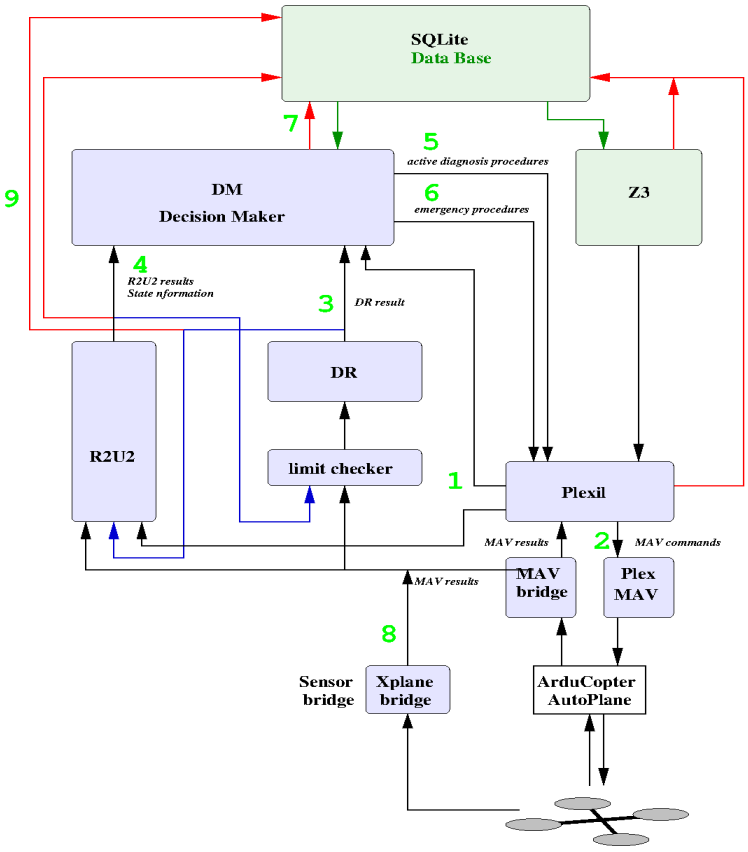
Departure Clearance: Right Turn 060 1 mile after departure, Radar Vectors SJC, climb 3000, expect 5000 10 minutes after departure.

```
[4] 007UAV : Received EFC for time 17 ; Alt =
5000
[4] 007UAV : RT 060
[5] 007UAV : Setting altitude: 3000
[8] ATC: Turn Heading 120
[9] 007UAV: Heading 120
[10] 007UAV: Radio check failed.
[11] 007UAV : Beginning lost comm procedure...
[12] ATC: Fly direct Sunol
[12] 007UAV : Squawking 7600
[13] ATC: NORDO 007UAV
[13] 007UAV: Attempting to hail ATC
[13] 007UAV: Attempting to visually
locate airports..
[14] 007UAV: Set waypoint SJC
[17] 007UAV: setting altitude to 5000, per EFC
[24] 007UAV: Reached waypoint: SJC
[24] 007UAV: Setting waypoint per flight plan:
Sunol
```

B. Advanced Contingency Management: Decision Maker

Advanced contingency management requires a deliberative process, which in AOS is provided through the Decision Maker app that generates plans which are then executed by Plexil. Current automation research has focused on providing human pilots increasingly better information for contingencies, such as vehicle sub-system diagnosis. But humans are still expected to close the loop. On-board human pilots are the fallback when an automated sub-system such as an autopilot is pushed beyond its nominal parameters – automation fails over to humans. For today's UAS, human ground crews are the fallback, but often lack the situational awareness to effectively catch automation failures.

This sub-section describes an implemented architecture and on-board reasoning capability that demonstrates the feasibility of graduating from automated systems to autonomous systems. It is currently focused on vehicle state and vehicle health mitigations, though similar reasoning capabilities are expected to be applicable for other contingencies. At the core is Decision Maker – an application that gathers input from diagnostic and prognostic modules (DR and R2U2), resolves ambiguities through active diagnosis, and then determines degradations in the vehicle state. Subsequently it projects forward the impact of the vehicle state going forward, considers and scores different alternative plans going forward, then chooses the best feasible plan and sends it to Plexil for execution. Plexil interacts directly with the autopilot, which for our current experiments with a drone is the ArduCopter software operating on a PixHawk. A sensor bridge provides sensor values during flight tests. During SIL and HIL testing simulated sensor values are provided through Xplane.



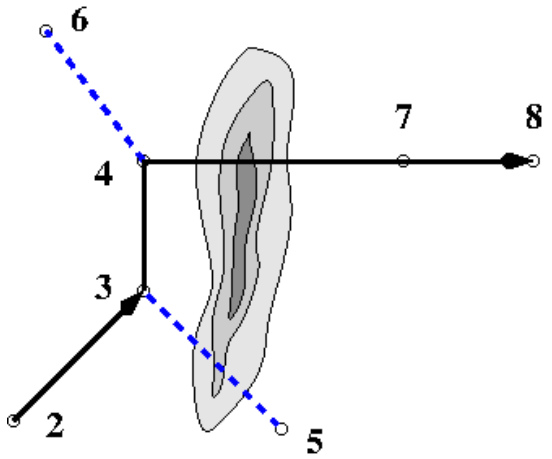
In the diagram above, DR provides diagnostic snapshot information at 1Hz for all sub-system fault states: normal, abnormal, or ambiguous. DR is configured through a *Diagnostic Matrix* for each vehicle, which maps discrete sensor values to candidate sub-system faults. The continuous sensor values are discretized by Limit Checker into nominal and off-nominal values that are sent to DR. The Diagnostic Matrix for a vehicle is compiled from a vehicle system model in a SysML-like format provided by design and system engineers. R2U2 provides a further level of diagnostic reasoning that feeds directly to Decision Maker or as a synthetic sensor for DR. R2U2 provides conventional signal processing capabilities (e.g, Kalman Filter), Bayesian reasoning (encoded as a Bayesian network that is compiled into an arithmetic circuit), and metric temporal logic (MTL) monitoring. The latter operates through a highly optimized interpreter that takes as input MTL formulae (expressing logical and bounded temporal predicates on sensor values and events) and a stream of sensor values and events. R2U2 outputs a stream of Boolean vectors corresponding to the MTL formulae valuations (through past-time or future-time temporal logic interpretation).

SQLLite provides a time-stamped knowledge base of logical assertions that can be asserted and retrieved by the various reasoning engines including Decision Maker. SQLLite has been configured to efficiently encode and retrieve logical formulae in an extended predicate calculus, built on the standard data base API for SQL. Z3 [6] is an open source Satisfiability Modulo Theory (SMT) reasoning engine from Microsoft that efficiently solves constraint satisfaction problems for other apps. SMT solvers combine propositional satisfiability procedures with automated solvers in various domains, such as linear arithmetic. SMT algorithms have advanced in problem solving capability by orders of magnitude since 2005, driven in part through an annual competition. SMT has become a standard part of formal verification of hardware and software systems. The advances in the speed of SMT constraint solving justifies its incorporation as a soft real-time component for AOS.

Decision Maker, implemented in Prolog, inherits its basic architecture from the AutoBayes program synthesis system [11]. AutoBayes takes as input a *specification* for a machine learning algorithm, and

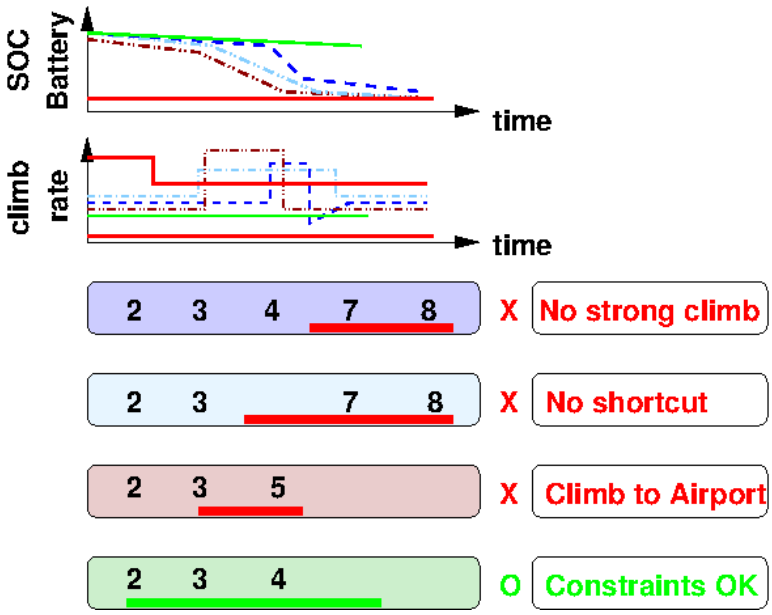
generates a combination of instantiated algorithm templates in a hierarchy and subsequently outputs code that implements the specification. Decision Maker takes as input a *description* of a contingency management problem, and then generates a plan in Plexil for resolving the contingency and charting a path forward for the aircraft to provide the best outcome, safely aborting the mission if necessary. Similar to AutoBayes, which recursively decomposes a specification into interacting parts, Decision Maker decomposes a contingency management problem into vehicle performance constraints based on disambiguated faults, and then a recursive branching path forward for alternative routing and commanded vehicle state changes.

The diagram below illustrates the overall contingency management process with Decision Maker (DM) through a scenario involving a small fixed wing UAS that encounters a large current sensed value for the left side of the aircraft. The flight originates at the airport designated 2, with an original flight path 3, 4, 7, and landing at airport 8. The contour lines represent mountainous terrain. The high current value is sensed halfway to waypoint 3 after takeoff from airport 2.

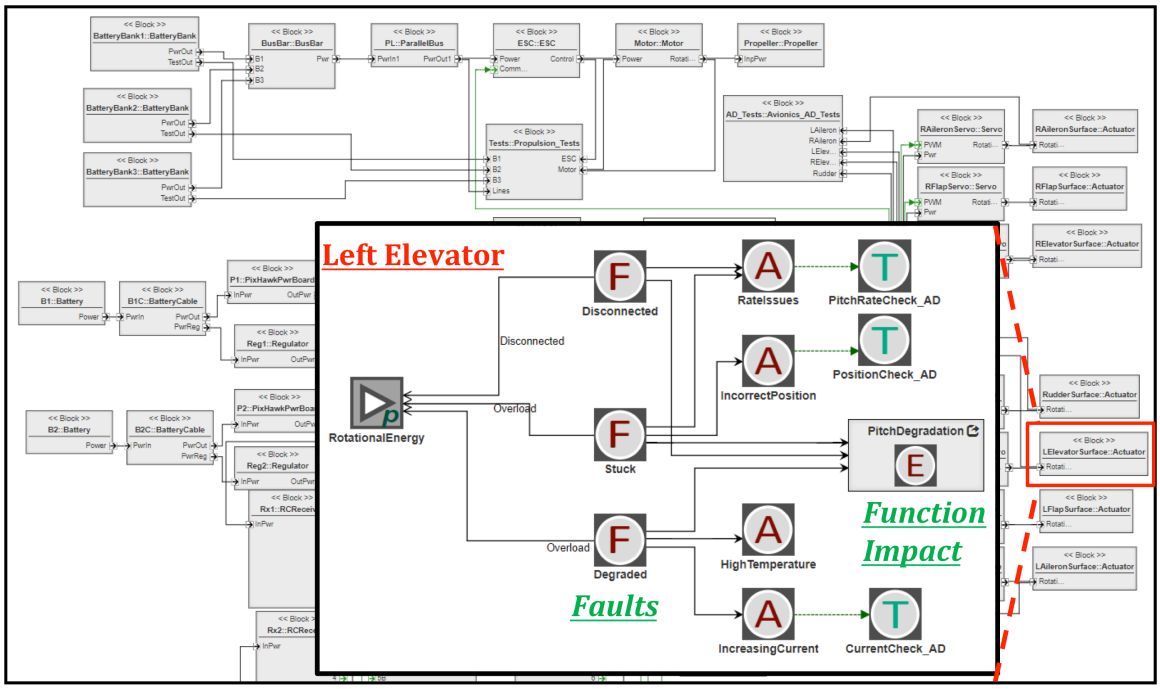


The high current sensor is flagged as an ambiguous fault by DR, which could be caused by various faulted actuators or faulted sensors on the left side of the aircraft (i.e., comprising an ambiguity group). For example, a stuck left aileron or stuck left elevator could cause this anomaly, as well as a faulty current sensor itself. To disambiguate, DM commands through Plexil a set of gentle maneuvers that isolate the problem – similar to a human pilot who moves the control stick around to check the response of an aircraft. Commanded rolls to the right then left yield the expected calibrated response, as sensed by gyro sensors. Subsequently commanded pitch up and down yields only 60% of the expected pitch rate values, isolating the fault to a stuck left elevator actuator (this particular UAS has separate left and right elevators, and hence separate left and right actuators). The stuck left actuator fault then is calculated to reduce the safe climb authority of the vehicle, and also the high current value will increase the battery drain.

The calculated vehicle degradation is then projected forward along the nominal path. The nominal path requires a steep climb over mountainous terrain, which is incompatible and unsafe with respect to the reduced climb authority. DM then deliberates over alternative paths. The reduced climb authority rules out any path over the mountainous terrain. The increased battery drain does not impact the possible direct routes, but would impact circumnavigation routes around the mountainous terrain. Finally Decision Maker decides that there is no feasible path to the original destination airport, and proceeds to waypoint 4. At this point it calculates an abort to alternative airport designated 6. The figure below shows the alternative routes Decision Maker considers, along with the forward projected constraints on the battery state of charge and the climb rate required.



This scenario has been simulated in the SIL with a full flight software stack using XPLANE for flight dynamics. An analogous scenario has been flight-tested on an Octopter with a lidar altimeter failure that degraded terrain-following capability, thus requiring re-planning. The vehicle systems knowledge used by the automated reasoning is developed in an extended Sys-ML graphical language in the open GME framework [12] by human systems engineers. The graphical descriptions are then automatically compiled into the representations used by the different AI components. Below is part of the hierarchical graphical representation for electrical and sensor vehicle sub-systems, with a blow-up of the elevator sensors and actuators. This particular description is for anomalies and faults, other descriptions support functional decomposition and reasoning, active diagnosis, and parametric reasoning.


Left Elevator
Disconnected
RateIssues
PitchRateCheck_AD
Disconnected
IncorrectPosition
PositionCheck_AD
RotationalEnergy
Overload
Stuck
PitchDegradation
Function Impact
Overload
Degraded
HighTemperature
Faults
IncreasingCurrent
CurrentCheck_AD

VI. Conclusion

This paper has described an implemented open architecture for smart UAS whose foundation is NASA's Core Flight Software (CFS). AOS provides a number of Artificial Intelligence capabilities for increasingly autonomous UAS. Apps are developed directly both on CFS, and using AI capabilities such as Plexil for basic pilot procedures, and automated reasoning for more deliberative autonomous operations. Natural communication with air traffic control (ATC) is provided both through datalink (CPDLC) and natural language processing interfaced to automated reasoning. Variable autonomy is achieved through the ability of AOS to fly under tight direction from ATC, to fly under broad guidance from ATC, and to autonomously manage contingencies. Representative scenarios have validated AOS capabilities in key areas of piloting, through both laboratory and drone flight-testing. AOS demonstrates the feasibility of autonomous pilots that could navigate the National Air Space responding like a human pilot to ATC, thus not causing congestion or imposing undue burden on ATC. AOS has also demonstrated the capability of handling contingencies such as vehicle sub-system failures on-board through deliberative automated reasoning, However, AOS currently only has a fraction of the knowledge needed by a pilot. The AOS architecture is designed so that new capabilities can be added through *apps,* similar to open architectures for smart phones. New capabilities and knowledge can leverage the extensive artificial intelligence capabilities that have already been integrated into AOS.

VII. Acknowledgments

The funding for Autonomy Operating System has been provided by NASA Convergent Aeronautic Solutions.

VIII. References

[1] McComas, D. "NASA/GSFC's Flight Software Core Flight System," 2012 Flight Software Workshop. Available: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20130013412.pdf

[2] NASA SBIR Abstract Windhover: https://aiaa-mst18.abstractcentral.com/login?URL_MASK=YNQplynQW6I

[3] Rios, Joeseph, NASA UAS Traffic Management National Campaign – Operations Across Six UAS Test Sites, IEEE DASC 2016

[4] Lowry, Michael, "Autonomy Operating System for UAVs: Pilot-in-a-Box", AIAA Aviation 2017

[5] Moura, Leonardo and Bjorner, Nikolaj, "Z3: An Efficient SMT Solver", in TACAS 2008: Tools and Algorithms for the Construction and Analysis of Systems. Springer Verlag 2008.

[6] Verma, V., Jonsson, A; Pasareanu, C.; and Iatauro, M.; "Universal executive and plexil: Engine and language for robust spacecraft control and operations," in Proc. of AIAA Space 2006. Plexil is available online at http://plexil

[7] Cunningham H., Maynard D., Bontcheva K. and Tablan V. "GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications", In proc. of the 40th Anniversary Meeting of the Association for Computational Linguistics, 2002. Download via https://gate.ac.uk/download/

[8] Munoz, C; Narkawicz, A; Hagen, G.; Upchuch, J; Dutle, A., Consiglio, M., Chamberlain, J. ; "DAIDALUS: Detect and avoid alerting logic for unmanned systems", IEEE Digital Avionics Systems Conference (DASC) September 2015

[9] FAA Instrument Procedures Handbook, FAA-H-8083-16B; Chapter 6 : Airborne Navigation Databases. Available at https://www.faa.gov

[10] In FAA Aeronautical Information Manual. Available at https://www.faa.gov

[11] Schuman, Johann; Cate, Karen, Lee, Alan, “Analysis of Air Traffic Data with the AutoBayes Synthesis System”, in Logic-Based Program Synthesis and Transformation: 20th International Symposium, Springer Verlag

[12] Lattman, Z.; Kecskes T.; Meijer, P.; Karsai G.; Volgyesi, P.; Ledeczi, A. “Abstractions for Modeling Complex Systems” in Leveraging Applications of Formal Methods, Verification and Validation, ISoLA 2016, Lecture Notes in Computer Science 9953, Springer Verlag