Autonomy Operating System for UAVs: Pilot-in-a-Box

Contributors: Michael Lowry, Anupa Bajwa, Michael Dalal, Patrick Quach, Patrick Castle, Fritz Renema, Jonas Johnson, Lawrence Markowsian, Charles Fry, Lilly Spinovich, Gabor Karsai, Johann Schumann, Kristen Rozier, Eric Rozier, Sanjay Rajan

Introduction

The Autonomy Operating System (AOS) is an open flight software platform with Artificial Intelligence for smart UAVs. It is built to be extendable with new *apps,* similar to smartphones, to enable an expanding set of missions and capabilities. AOS has as its foundations NASA's core flight executive and core flight software (cFE/cFS). Pilot-in-a-Box (PIB) is an expanding collection of interacting AOS apps that provide the knowledge and intelligence onboard a UAV to safely and autonomously fly in the National Air Space, eventually without a remote human ground crew. Longer-term, the goal of PIB is to provide the capability for pilotless air vehicles such as air taxis that will be key for new transportation concepts such as mobility-on-demand. PIB provides the procedural knowledge, situational awareness, and anticipatory planning ('thinking ahead of the plane') that comprises pilot competencies. These competencies together with a natural language interface will enable Pilot-in-a-Box to dialogue directly with Air Traffic Management from takeoff through landing. This paper describes the overall AOS architecture, Artificial Intelligence reasoning engines, Pilot-in-a-box competencies, and selected experimental flight tests to date.

AOS Heritage: NASA Core Flight Software

Software reuse is hard. Even harder is to provide an open platform that empowers a community of developers to contribute an expanding set of reusable software modules. The *iOS* and *Android* smartphone platforms have successfully fostered communities of developers, for deployment of collections of loosely interacting *apps*. These have provided billions of smartphone users with highly versatile and configurable devices. In aerospace, the challenges for an open platform are more difficult: the interactions between applications are often tightly coupled, and both platform and apps need to be certifiable.

NASA's core Flight Executive (cFE) has met the challenges of an open, certifiable software platform for small spacecraft. An expanding library - Core Flight Software (CFS) - of reusable software applications, ranging from thermal management to attitude control to telemetry, has been developed and is provided through an open repository. NASA Goddard originated cFE/CFS, and it has now been adopted by all of NASA and many outside participants. Over 22 small spacecraft missions have been successfully certified through NASA and flown using cFE/CFS. In addition, NASA Johnson has developed and certified to level A (safety critical human rating) a version of the Orion backup flight software based on CFS. This provides evidence that the high level of software assurance needed for eventual FAA certification is achievable through a foundation based on cFE.

AOS Architecture : An Open Platform

Both NASA's AOS project and outside companies have demonstrated through flight tests that cFE/CFS can transition from small satellites to UAVs. UAVs have a different pattern of data throughput, command and control throughput, and interrupt events than small satellites, so this was not a forgone conclusion. The AOS project has successfully flown the UAV software architecture whose foundation is cFE/CFS on flight experiments since June 2016. This architecture is illustrated as a block diagram in Figure 1.

AOS is focused on intelligent UAV applications, such as the pilot procedural knowledge specified in FAA documents, e.g. FAR/AIM. In addition to AOS, a number of small companies have developed and flown remote controlled UAVs using cFE/CFS. The apps include autopilots, sensor/actuator device interface, and battery management – with the goal of near term FAA certification. An open-source cFE UAV software for the Parrot Drone suitable for teaching college students the fundamentals of embedded systems is available. In contrast to other platforms suitable for UAV software prototyping such as Robot Operating System, cFE/CFS is built from the ground up to provide real-time guarantees and the software assurance that will be required for certification.

AOS inherits the NASA Ames LADEE (Lunar Atmosphere Dust and Environment Explorer) version of cFE/CFS, which incorporates many Matlab-compiled applications. A toolchain that simplifies generating an AOS application from Matlab is provided as part of the AOS open platform. The LADEE architecture is in blue below. LADEE apps that were reused from the CFS library are circles below the blue line for the software bus. Apps that were generated specifically for LADEE through Matlab are above the blue line.

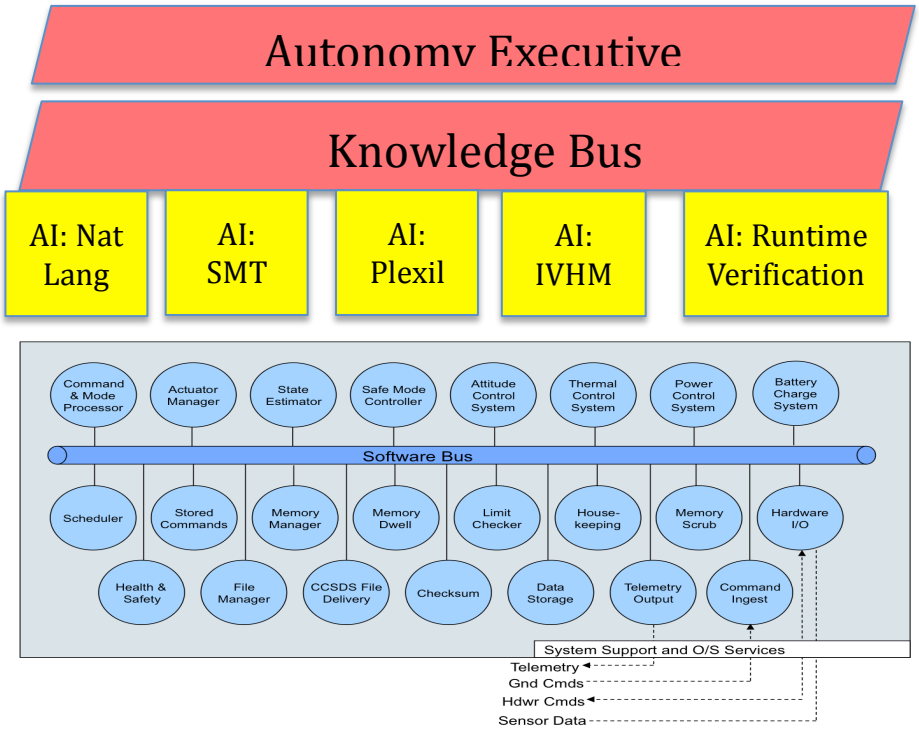

Figure 1: Block Diagram of AOS Architecture

AOS has integrated with cFE a critical mass of Artificial Intelligence (AI) capabilities including diagnostic reasoning(DR) for IVHM, plan and procedure execution (Plexil), an automated reasoning engine, and natural language processing. The integration is through the 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 mote AI reasoning engine already interfaced to cFE.

The natural language engine takes as a model an ontology and a grammar, and the stream of data is utterances from Air Traffic Control and pilots on other aircraft. The output is assertions that are put on the Knowledge Bus. The SMT engine is for automated logical reasoning: it takes as a model a logical theory formulated for SMT-LIB, the data is a set of assertions, and the output is deduced assertions. The diagnostic reasoner for IVHM takes a vehicle system model, such as the wiring diagram for the electrical subsystem; and a stream of input data from current, voltage, and temperature probes; and produces a stream of time-stamped diagnoses as to which components are working normally and which are faulted.

The runtime verification serves as a hybrid between a standard watchdog process and an intelligent co-pilot. The runtime verification capability is provided by R2U2, which incorporates

capabilities for Bayesian (probabilistic) reasoning, signal processing, and fast monitoring of temporal logic formulae over sensor and event input streams. It serves as a model-based cross-check on the internal actions of other components. The reasoning capabilities provided by R2U2 are more limited than the other engines, but they are guaranteed to run within small time bounds

Plexil takes a plan formulated as a hierarchy of activity nodes and then flexibly executes the plan with data that is provided from the software message bus and the knowledge bus, taking into account when execution failures of children nodes require alternative procedure executions. In addition to executing plans generated from a planning engine, Plexil can also take as input plans developed by human designers. The Plexil language resembles software procedural languages; the Plexil compiler takes a Plexil program and translates it into a hierarchy of activity nodes. Plexil has proven to be a versatile language for encoding the procedural knowledge required of pilots.

The first Plexil plan flown on an AOS flight experiment was the FAA IFR lost communications procedure (14 CFR 91.185), in June 2016. In contrast to the fixed (before takeoff) lost link procedures currently implemented on remote controlled UAVs, the FAA IFR lost communication procedure requires pilots to consider **both** the flight plan and the history of interactions with ATC to determine the appropriate flight path to follow. As a result of this more encompassing input that includes the ATC interactions, an IFR lost communication airplane flies a predictable path that minimizes disruption to the airspace and allows air traffic controllers to minimize the impact to other aircraft. In order for air traffic operations to be robust, the potential loss of communications is interwoven into Air Traffic clearances. For example, departure and enroute clearances typically incorporate “Expect Further Climb” and “Expect Further Clearance”, not just as a heads up to pilots, but because these expectations are specifically called out in the FAA IFR lost communication procedure. The AOS IFR lost communications Plexil plan correctly implements 14 CFR 91.185 and handles these ‘expectations’ embedded in ATC clearances.

The AOS project has already successfully encoded many of the backbone FAA procedures into Plexil, including departure, enroute, and the VFR traffic pattern. But there are numerous procedures pilots must follow documented by the FAA in FAR/AIM, guidebooks such as Instrument Procedures Handbook, Aircraft Flying Handbook, and pilot practical test standards. The AOS open platform *apps* strategy addresses this large number of FAA procedures by encouraging outside developers to contribute. In that regard, student interns developed the first version of the VFR traffic pattern AOS app, and flew it August 2016. Student AOS projects at two universities are planned for 2017. The lessons learned from these student projects, together with the heritage from cFE/CFS, is expected to enable AOS to become a low-barrier open platform with an expanding library for smart UAV development.

In order to provide UAVs autonomous access to the National Air Space, many if not all of the FAA specified human pilot competencies will need to be transferred to onboard machine intelligence. The AOS project is developing the core of these competencies as a collection of interacting apps called Pilot-in-a-Box.

Pilot-in-a-Box

The objective of PIB is to migrate UAV command and control from remote human ground crews to onboard autonomy. The certification of these competencies is expected to be more challenging than the certification of fully autonomous self-driving cars, both because the FAA is a more rigorous certifying authority than NHTSA and state DOTs, and also because piloting skills are more cognitively demanding than driving skills.

The military requires both an instrument-rated pilot and sensor/systems operator to fly large predator-class drones on DoD missions. In order to fly in the civilian US National Air Space, these large UAVs require an additional instrument-rated co-pilot and a dedicated person to co-ordinate with civilian Air Traffic Control. The additional pilot provides the cross-checking between crew members that leads to the safety of part 121 operations in civilian air space. The ATC co-ordinator handles the dialogue with air-traffic control.

The AOS project is developing all four of these capabilities:

• Co-pilot: part 121 operations that require a co-pilot have accident rates that are several orders of magnitude less than general aviation single pilot part 91 operations. The co-pilot cross-checks the PIC both through co-ordinated execution of checklists and independently observing the flight progression. (The co-pilot can also relieve workload saturation for the PIC).

Embedded computer systems also use cross-checking of a primary processor by a second – watch dog processor - to catch runtime execution problems. However, at present this watch-dog cross-checking is at a low system level such as monitoring heartbeats from the primary processor and restarting the primary processor as necessary. As described above, AOS incorporates a second processor – R2U2- that is running checks at a much higher functional level. (Currently R2U2 is run on the same processor as the rest of AOS, but will be migrated to its own special-purpose FPGA processor to provide independence from processor and software faults.) The temporal logic checking has proven computational time bounds, guaranteeing that checks are run in fractions of a second. A checklist is encoded as a sequential series of logical formula. While R2U2 does not have the capability to relieve the workload on Pilot-in-a-Box, as is done by a human co-pilot, it does provide the independent cross-check that is the machine intelligence counterpart to human crew resource management.

• Air Traffic Control dialogue: air traffic control ensures the separation and orderly flow of air traffic. The FAA has begun migrating portions of pilot-ATC interactions to data-link, using the CPLDC protocol. AOS is incorporating a CPLDC capability as a front-end to Pilot-in-a-Box. However, for the foreseeable future, more complex and timely ATC-pilot interactions will still be done through natural language. For example, in a busy general aviation airport operating a VFR traffic pattern, the tower controller often provides complex sequencing instructions to pilots that involve modifications to downwind, base, and final legs such as landmarks (e.g., “enter downwind midfield”), timing (e.g., “I’ll call your base”), and other aircraft (e.g., “provide spacing for two departures”). In addition, the tower controller provides situational awareness to pilots ranging from position reports of other aircraft to runway conditions to birds in the vicinity.

AOS has incorporated a natural language text interface to Pilot-in-a-Box, that has already been prototyped with Nuance Mix cloud server for the VFR traffic pattern. Utterances from Palo Alto airport (KPAO) during high-density operations have been used to train the semantic parser for VFR traffic pattern interactions through machine learning. We are investigating whether recent improvements to machine speech recognition technology can meet the needs of rapid-fire voice loop communications. The pilot read-back protocol required by ATC provides a positive

handshake that would make small levels of inaccurate speech recognition acceptable. The VFR traffic pattern was chosen because it exemplifies the more complex and rapid-fire interactions between pilots and ATC that would be difficult through CPLDC datalink. The natural language text capability and speech recognition are in the process of being migrated from the cloud to onboard.

• Pilot-in-command: Pilot-in-a-box

Pilot-in-a-Box provides the procedural knowledge, situational awareness, and anticipatory planning ('thinking ahead of the plane') that comprises pilot competencies. The procedural knowledge is encapsulated in Plexil procedures such as lost communication, departure, enroute, and VFR traffic pattern. To date, the situational awareness is threefold: awareness of the vehicle system status, awareness of the ownship flight envelope and operating parameters, and awareness of other aircraft and their intent. The vehicle system status awareness is implemented by the Diagnostic Reasoner, which tracks not only vehicle sensor data but also flight modes (climb, cruise, descent) and their transitions. The awareness of other aircraft and their intent is provided by matching descriptions from Air Traffic Control to GPS tracks in ADS-B format.

Talking with ATC

Speech communication is integral to Air Traffic Control. Pilot-in-a-box incorporates recent advances in machine natural language processing in order to provide a natural interface to ATC. The AOS project developed specialized concept hierarchies for the Air Traffic Control domain, and then generated semantic frame parsing grammars through machine learning as part of the Nuance cloud SDK. During real or simulated flight, ATC utterances are parsed into a parse tree through the semantic frame grammar, and then transformed into logical formula. These logical formula are then transmitted on the knowledge bus to the Navigation manager and the Dialog manager. These managers then invoke automated reasoning in order to process clearances, to verbally respond to ATC, and to develop airspace situational awareness from ATC descriptions of traffic and other entities.

A portion of the concept hierarchy used in parsing traffic information is shown in figure 2.

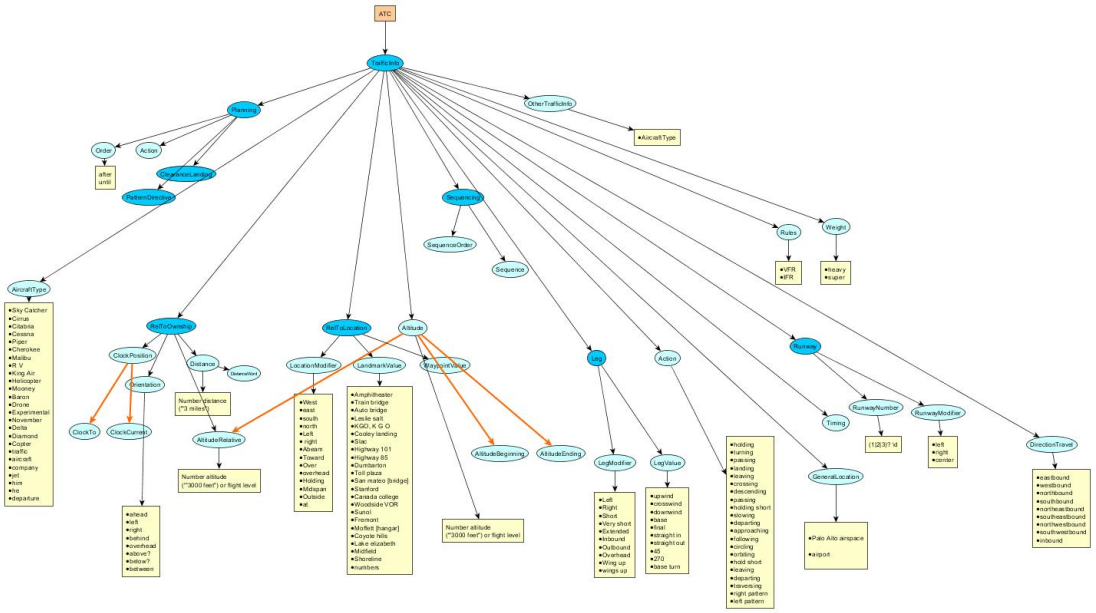

Figure 2: Concept hierarchy for Traffic Information

AOS has tested the natural language processing with speech recognition in desktop and headset environments. The acoustic signal from a microphone is digitized and then processed to text through speech recognition systems. The semantic frame grammar is robust to the typical 4-5% word error rate. Our acoustic testing environment is similar to ATC terminal tower acoustic environment with noise-cancelling headsets. For speech originating in the aircraft headset environment, there have been several certified products for pilots to command navigation systems through speech. This provides evidence that speech transmissions from human-piloted aircraft to AOS-piloted aircraft could also be decoded if the transmitted speech is digitized.

Digitized speech transmission is now prevalent in public safety and business radio frequency bands where the FCC has mandated migration from 25Khz bandwidths to 12.5 KHz bandwidths (i.e., FCC narrow-banding mandate). The FCC is contemplating 6.25KHz bandwidths on congested frequency bands, which will accelerate the migration from analog to digital transmission. As part of NextGen, the FAA is transitioning its ground voice communication system from analog to digital, in order to enable capabilities such as controllers at distant facilities to be switched into a sector when airspace become congested. EUROCAE (European Organisation for Civil Aviation Equipment) has already developed operational and technical requirements for Voice over Internet Protocol Air Traffic Management Systems. Thus in the intermediate term, targeting digitized speech as opposed to analog radio transmitted speech is expected to be only a minor impediment to AOS deployment. For the immediate future, the poor acoustic quality of analog transmitted speech in the current Aviation frequency band will be a challenge for AOS deployment. Research is ongoing to address this challenge.

PIB includes a datalink capability based on the CPDLC standard for those portions of flight that the FAA is transitioning to text exchange. The FAA started rolling out CPDLC for departure clearances at major airports in 2016, with plans for enroute communications to be shifted to datalink.

ATC communications in the terminal area will likely remain as speech, because pilots need to respond to ATC with low latency (i.e., a few seconds) while maintaining visual focus on instruments and outside the cockpit. ATC directives for sequencing, spacing, landing clearances, and terminal pattern clearances need to be followed and acknowledged quickly for ATC to manage aircraft in close proximity. Communication between aircraft will also likely remain as speech, especially in un-towered environments.

For these reasons the AOS project has focused developing natural language and automated reasoning capabilities for the terminal environment with digitized speech. Terminal operations are also the key to unlocking the civilian National Air Space to regular access by autonomous fixed wing air vehicles. Remotely piloted aircraft have accessed military airports such as Creech Air Force Base for a number of years. However the terminal controllers require special training, and the limited situational awareness of the remote pilots and the latency involved in remote piloting imposes considerable workload on the controllers. In essence the controllers need to verbally joy stick the UAV through the terminal environment by communicating low-level directives (e.g., slow down 10 kts, turn left ten degrees, turn to base in ten seconds) to the remote pilot. This workload reduces the number of aircraft the terminal controllers can handle at one time by 50%. In contrast, aircraft with in-situ human pilots can follow higher-level spacing, sequencing, and traffic directives that relieve workload on air traffic controllers. Capable autonomous aircraft could bridge the gap between current remotely piloted UAVs and directly piloted aircraft.

Palo Alto airport (KPAO) was chosen as an exemplar for the VFR terminal environment. Palo Alto is amongst the busiest general aviation single runway airports. The AOS project also cross-checked with a database of 20,000 transcribed utterances recorded from the busiest commercial FAA facilities, including control towers such as Reagan, Atlanta, and Dallas-Fort Worth. This database resulted from a research agreement between NASA Ames and the FAA.

The majority of communications between ATC and flying aircraft in the terminal area are short, repetitive, and routine. However, as density increases, the communications become more complex as ATC offloads airspace situational awareness to the pilot, who becomes responsible for following other aircraft in sequence, self-spacing, and traffic avoidance.

Knowledge Bus

From an anthropomorphic viewpoint, the knowledge bus provides the machine equivalent of stream of consciousness and short-term memory as a pilot performs aviate, navigate, and communicate tasks. In AOS, the stream of consciousness consists of time-stamped logical formulae. The current AOS implementation is to store and retrieve these logical formulae encoded in a relational database. SQL queries provide an efficient means of selectively filtering and retrieving the formulae for purposes such as automated reasoning, described in the next section.

The time-stamped formulae can originate from both internal and external sources. For example, when an ATC utterance is received, it is first parsed into a tree, and each node of the tree becomes a relational formulae. These formulae are time stamped, stored in the SQL database, and then retrieved for *intra-utterance resolution* by automated reasoning. The intra-utterance resolution performs several functions, including unifying references within a single utterance, and generating internal actions for communication and navigation. The result of *intra-utterance resolution* is the generation of further sets of formula including those denoting actions to be done by the dialog manager, the navigation manager, and situational awareness modules such as acquire-traffic. For example, given the following Palo Alto airport terminal utterance (utterance 1):

“N007JB, Palo Alto terminal, make right traffic, follow the Bonanza mid-field right downwind, additional traffic is a departing Mooney traversing from your eleven o’clock to your two o’clock. Report traffic in sight.”

After parsing, intra-utterance resolution produces the following formulae (simplified with less verbosity compared to the actual machine encoding):

(= Traffic1 Aircraft1)

(Model Aircraft1 Bonanza)

(Location Traffic1 [Lat1, Long1])

(Intent Aircraft1 (Legs [PAO-right-downwind, PAO-right-base, PAO-final]))

(= Traffic2 Aircraft2)

(Model Aircraft2 Mooney)

(Direction Traffic2 330 60)

(Intent Aircraft2 Departure)

(Ownship Navigate-Action [PAO-right-downwind, PAO-right-base, PAO-final])

(Ownship Navigate-Action Follow Traffic1)

(Ownship Acquire-Traffic-Action Traffic1)

(Ownship Acquire-Traffic-Action Traffic2)

(Ownship Communicate-Action Readback Utterance1)

(Ownship Communicate-Action (Acquire-Traffic Traffic1))

(Ownship Communicate-Action (Acquire-Traffic Traffic2))

In order to generate these formulae, intra-utterance automated reasoning performs deductions such as the following, as well as transforming the formula from the parse tree to action descriptions:

1. The ‘traffic’ to be reported are the aircraft described by ATC earlier in the utterance.

2. If ATC directs ownship to follow another aircraft, then the aircraft being followed has the same intent as that assigned to ownship.

3. If an ATC utterance conveys clearances or directives, readback is required.

Automated Reasoning

Machine automated reasoning according to rules of logical deduction was once thought to be too slow for real-time embedded systems. However, orders of magnitude improvements in the algorithms for logical deduction since 2000, and the continued advancement of Moore’s law since the 1970 introduction of a processor on a chip, has made logical deduction suitable for real-time embedded aviation systems performing pilot cognitive tasks. The advantage of logical deduction as the principle means of automated reasoning for Pilot-in-a-box is that it is sound, it is traceable, and it is general purpose. In fact, PIB currently implements its automated reasoning through a program principally developed for formal software verification: Microsoft’s open source Z3 SMT solver.

Machine automated reasoning for logical formulae originated in the 1950s with Newell and Simon's Logic Theorist [Newell 1956]. The Logic Theorist proved 38 of the first 52 theorems in chapter 2 of Whitehead's Principia Mathematica. Subsequently in 1965 Alan Robinson developed a simple and complete rule for logical reasoning, called resolution. Cordell Green demonstrated how resolution could be used as an algorithm for a wide variety of problems in Artificial Intelligence, including question answering, planning, software verification, and automatic program synthesis. However subsequent advances in resolution were not sufficient for the speed required for real-time embedded systems, although ground-based Aerospace applications were successful (e.g., Amphion).

In 2000 two Princeton undergrads (Matthew Mskewicz and Conor Madigan) engineered the *Chaff* algorithm that solved propositional satisfiability problems with 10x to 100x speed improvements compared to other existing algorithms. The undergrad's success was the result of careful engineering and elegant combination of existing techniques, rather than a new mathematical or algorithmic advance. Their success inspired the automated reasoning research community to develop benchmarks and hold competitions for a number of classes of automated reasoning algorithms to spur both mathematical and engineering advances. Satisfiability Modulo Theory (SMT) is a generalization of propositional satisfiability where efficient decision procedures for non-propositional domains such as linear arithmetic are combined with propositional satisfiability. In practice, SMT algorithms are a general constraint solver based on logical deduction. Since the SMT competitions began in 2005, there has been over five orders of magnitude improvement in the speed of SMT algorithms. In addition, the complexity of problems that can be solved in a set amount of time (e.g., a fraction of a second) has greatly increased.

The responsiveness of the Z3 SMT solver has been demonstrated both for intra-utterance resolution and situational awareness tasks such as determining landing order sequencing from Air Traffic Control utterances. For the latter, the tests have included synthetic situations with upto 50 planes in the traffic pattern, with completion of landing order sequence within seconds. For more realistic scenarios with no more than 16 planes in the traffic pattern, the determination is under a second.