Why Controllers Seldom Stick to the Book and How Their Commands are Predictable Nevertheless

Michael Hössl
Institute of Aerospace Technology
University of Applied Sciences Bremen, Germany
and Institute of Flight Guidance (DLR)
michael.hoessl@ymail.com

Hartmut Helmke, Jens Gottstein
Institute of Flight Guidance
German Aerospace Center (DLR)
Braunschweig, Germany
hartmut.helmke@dlr.de, jens.gottstein@yahoo.de

***Abstract*— Modern air traffic approach control often uses electronic assistance systems. Routing and sequencing suggestions are issued to support the controller. To provide these systems with data about future possible system states, the Mapped Exclusion Method (MEM) has been developed. MEM enables the prediction of a set of possible controller commands from radar data. The algorithm is rule based and uses operational knowledge of the airspace to predict future ATC commands. Using the OpenStreetMap geo-information system and the interpreted programming language Python, a software tool was developed which even enables ATC experts without programming knowledge to model their airport's airspace.**

In combination with the author's experience as commercial airline pilot, flight tracks and controller pilot communication have been analyzed and relevant findings were implemented into functional software. For an inbound flight approximately 130 different command-value combinations are possible, from which 75% can be excluded on average, while the prediction error rate can be kept below 1%.

Keywords- Arrival Management (AMAN), Automatic Speech Recognition (ASR), Intent Prediction; Command Prediction

I. INTRODUCTION

A significant part of human collaboration is coordinated via voice, especially if complex contexts or meta-concepts are involved. Controlling aircraft in the vicinity of an airport is an example of such a working environment in which two working groups – i.e. pilots and controllers – implement a smooth, efficient and safe traffic flow via radio communication. All pilots in the same sector are supported by a dedicated controller (team). They use a unique frequency for communication within the sector. This enables the party line effect, i.e. all participants can create a common mental model of the current situation and of future actions. Unfortunately this does not include today's controller assistance tools yet, since these cannot recognize voice commands. However, voice communication will definitely remain a pillar of air traffic control. The Strategic Research & Innovation Agenda (SRIA) of ACARE [1] or Flightpath 2050 [10] do not expect a fully automated ATM-System in the next decades.

Air traffic controller support systems like DLR's Four Dimensional Cooperative Arrival Manager (4D-CARMA) are based on published airport instrument procedures [16]. These procedures offer a guideline to both pilots and air traffic control (ATC) on how a departure or arrival has to be executed. This paper focuses on the highly dynamical approach sector, where it is very common for air traffic controllers to deviate from published procedures. These changes are communicated via radio link to the pilot. Controllers do so in order to implement shortcuts or dynamic solutions based on experience, aircraft separation requirements and traffic load. There are no printed rules on how controller deviations are issued – it is the result of years of training and experience.

The fact that the automation does not follow human communication splits the interaction into two different worlds: one, in which humans communicate via radio links, and another one, in which machines communicate via computer networks. These worlds are connected by a human machine interface through which humans inform the machines and vice versa. As controller deviations especially occur in situations with high workload, the controllers do not have time to inform the assistant system about their intents. In this case, the automation may suggest advisories contrary to the intent of the controller. This situation may persist until the assistant system realizes the deviation through analysis of radar data. Hence, the system needs support from the controllers exactly when the controllers would urgently need the support of the system, due to a high workload. Automatic Speech Recognition (ASR) can cut this Gordian knot without additional effort for the operators. It enables common situation awareness without discrepancies between automation and humans.

ASR performance in ATM applications can significantly be improved if dynamic context information is available. Shore et al. [25] report on a word error rate (WER) reduction by a factor of 5. In our approach the dynamic context needed by the speech recognition is the set of possible expected controller commands. This set has to be defined precisely, because on the one hand ASR performance increases with the number of controller commands we can exclude from the set of possible controller commands (reduction rate, RR), but on the other hand ASR performance decreases if actually issued commands are excluded (context error rate, CtxER).

To enable an automated system to generate dynamic context about expected controller deviations, the system needs to have access to operational data and be able to compute possible changes according to observed practice rather than printed

The work was conducted in the AcListant® project, which is supported by DLR Technology Marketing and Helmholtz Validation Fund.
Michael Hössl now works as a first officer on a Boeing 737 aircraft, and Jens Gottstein is now with Siemens AG.

rules. Therefore, implementing the controller's 'gut feeling' into an algorithm is the main challenge.

In the next chapter we give a short overview of related work focusing on ASR and arrival management. Chapter III describes the knowledge sources (e.g. controller interviews, radar data) we use for command prediction. Chapter IV concentrates on the algorithm to model this knowledge, whereas the following chapter describes the implementation with the help of Python and the Java OpenStreetMap Editor (JOSM). Chapter VI presents our test results and chapter VII describes next steps, i.e. the integration of probabilities for which we currently have not performed tests.

II. Related Work

With rising demand more sophisticated assistant systems are introduced to support ATM operations, e.g. Arrival Managers (AMAN), Surface Managers (SMAN), and Departure Managers (DMAN). First commercial implementations of an AMAN have been operational at hubs (Frankfurt, Paris) since the early 90s. Today their application is still limited to the coordination of traffic streams between different working positions (e.g. sector and approach controllers) [14]. An extension of their application to support the controllers with advisories in order to implement fuel and noise efficient approaches currently fails when controllers start to deviate from AMAN advisories. At this point automatic speech recognition becomes necessary.

The word error rate (WER) is the most commonly-used metric for evaluating ASR performance [19], [20]: Current systems have a WER between 15% and 35% for conversations via telephone lines. Digit string recognition via headset can achieve a word error rate of less than 1% (under optimal conditions). If there is background noise, however, the WER can easily rise to 30% or 50% at a signal to noise-ratio of 0 dB. Participants of the second Pascal-Speech-Separation-Challenge even had to accept a WER of 50%. At the moment speech recognition is far from being perfect. This is also valid in the context of ATC with its very limited grammar and standardized phraseology. Cordero et al. recently reported of word error rates of 30% when applied in real traffic situations [5].

In ATM, it is not important that every word is recognized. It is important that the detected concept is correct. It is not important that ASR correctly recognizes “Good morning Lufthansa one two tree descend level one two zero”, but that the command “DLH123 DESCEND FL 120” is extracted. The command error rate quantifies this metric.

ASR is not new in the context of ATM. Hamel [13] and Weinstein [26] have described the application of speech technology in ATC training simulators in the early 90s, however with limited success. Dunkelberger et al. [7] described an intent monitoring system which combines ASR and reasoning techniques to increase recognition performance: In a first step, a speech recognizer analyses the speech signal and transforms it into an N-best list of sentence matches. The second step uses context information to reduce the N-best list. Schäfer [24] applied ASR in the context of an ATC simulation environment to replace simulation pilots (so called *pseudo pilots*) in validation trials with controllers. He used simulation data to predict what a given aircraft's future status will be (e.g. at which altitude and airspeed it will be flying), and which possible conflicts may occur that need to be resolved by the controller.

Currently, different successful applications of ASR in ATC training simulators are available. The FAA reports the successful usage of advanced training technologies in the terminal environment [11]. The Terminal Trainers prototype, developed by CAASD (Center for Advanced Aviation System Development, USA), includes voice synthesis, speech recognition, multimedia lessons, game-based training techniques, simulation, and interactive training tools. The German air navigation service provider DFS uses the system *Voice Recognition and Response* (VRR) of UFA (Burlington, MA) for controller trainings since August 2011. Less pseudo pilots are needed in its flight service academy. DFS, however, also reports that these systems are currently only usable for training purposes, because they only accept standard phraseology [4].

DLR’s application aims at a completely new application area: that of an overhearing speech recognition system which is embedded in an operational context. In contrast to existing applications, there is a human-human communication and the system is listening in to derive knowledge about the plans of the human participants [15]. This aspect is somewhat similar to the goals of the AMIDA project in which meetings were overheard and summaries were generated [2]. The new aspect of the proposed approach is that the speech recognition system is part of an agent (assistant system) that has knowledge about the physical world and at the same time makes plans about actions, which are then proposed to the human participants. Thus, one novelty is that the overheard human-human communication is directly used to improve the assistant system. The other novelty is more of a by-product: the physical context of the planning system (e.g. the aircraft in airspace) is used to improve speech recognition. This is based on the fact that not all interpretations of a speech utterance are equally likely in a given physical context.

III. Knowledge Engineering for Controller Intent Recognition

Speech recognition for aviation applications benefits from the high level of standardization and procedural structure that is used for all commercial airline operations [8], [18].It is sufficient to detect the used concept rather than every single word. Especially the standardized grammar limits the number of possible word combinations. However, what if it was possible to reduce this set of potential word combinations? A smaller number of combinations to choose from could mean a better chance of recognizing the correct terms. This reduction can be achieved by anticipating air traffic controller commands i.e. by excluding commands which are not feasible due to operational constraints from the context.

Real life air traffic control operations include a great number of variables that can cause the air traffic controller to deviate from published procedures. Reasons are e.g. different airplane types and their aerodynamic characteristics, payload, airline standard operating procedures, environmental

considerations, weather and above all, the omnipresent necessity to keep approach throughput high. Even though these deviations may seem random at first, it was shown that they do follow certain rules, ultimately allowing a reduction in possible controller commands [17].

Dusseldorf Int. Airport, located in the busy airspace of the Rhine-Ruhr metropolitan area in Western Germany has been selected as example airport for this project. The basic principle of our considerations, however, is universally applicable. This is achieved by strict separation of generic and airport specific data and discussed in detail in [17]. To meet the requirement of reliable command context generation, operational research had to be conducted. This deemed itself necessary due to the lack of available procedural information on daily operations and the dependence of everyday deviations on the local airport setup. The means of acquiring such operational information are threefold.

A. *Air Traffic Controller Interviews*

Consequently interviews were held with four air traffic controllers of Dusseldorf Approach Control. Additionally their work has been observed, in order to recognize patterns and reason as to why these deviations are performed. In combination with Mr. Hössl's experience as commercial airline pilot, flight tracks and possible instructions were analyzed and the resulting deviations divided into three categories.

1) *Descent Profile and Airspace Limitations*

The descent angle of any given aircraft is limited by basic physics and it is understood that excess drag cannot be created limitless due to aerodynamic boundaries. Therefore, a standard descent angle of 3°, shallow enough for all types of aircraft, has been established worldwide (where possible) for instrument approaches. The 3° angle is not only limited to the final approach phase, but in reality used, when possible, during all descent phases all the way back to the Top Of Descent Point (TOD) in the en-route part of flight. Modern jet aircraft operate most efficiently at high altitudes whereas prolonged flight at other than the optimum altitude increases fuel consumption. The controller will try to allow a continuous descent for all aircraft under his supervision in order to keep the aircraft at high altitude as long as practicable. In Dusseldorf over 70% of approaches are so called continuous descent approaches (CDA) [17] and the altitude is incorporated into the approach sequence in a substantial manner.

However, due to the compact airspace structure of the Rhine-Ruhr metropolitan region various deviations from the above stated standard have been observed. Especially aircraft arriving from south east direction are facing very late descent due to the underlying airspace of Dortmund Wickede Airport, causing increased descent rates, delayed sequencing or longer transition vectors.

2) *Feeder Operations on Downwind*

The time period from when an aircraft enters downwind to when it is fully established on final approach are characterized by a sequence of commands, affecting heading, altitude and airspeed almost simultaneously. It is the phase of flight, pilots reconfigure their aircraft to landing configuration and the feeder controller builds the final approach sequence according to operational standards. Typical commands include the turn on localizer intercept heading (usually 30° - 45° offset from the runway heading), the descent to the glideslope intercept altitude at the final approach point and a speed reduction below minimum clean speed, which is usually 210-220 kt.

3) *Shortcuts to Increase Efficiency*

With high operating cost and low profit margins the airlines struggle to generate revenue, which is of increasing concern to both ATC and cockpit crew. Delay or detour on any given flight has financial consequences for the airline and should be avoided. To achieve this goal, ATC uses shortcuts from the published RNAV transition, to reduce track mileage of arriving traffic. This of course leads to further deviations from the published track. In this phase of the approach, mainly heading and direct commands are issued. For example, arrivals via the GPS waypoint BIKMU encounter the possibility of reducing their distance to land, by turning into the south downwind via DL406 rather than following the (longer) published procedure via DL426 (see Fig. 1). ATC will usually support this, unless there is too much other arriving traffic in the area.

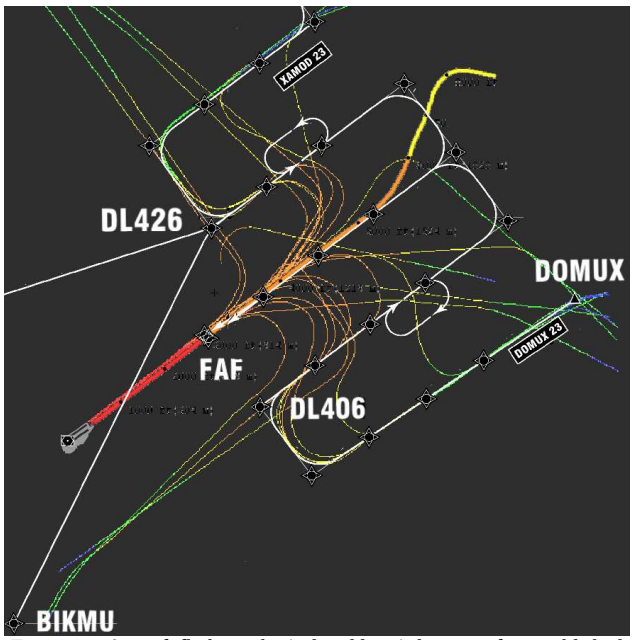

Figure 1. Aircraft flight tracks (colored lines) deviating from published procedures (white lines) at the approach into Dusseldorf Int.

B. *Stanly Radar Tracking*

In order to gain additional data the Stanly Track tool of German Air Traffic Control (Deutsche Flugsicherung – DFS) has been used. Stanly Track allows the user to monitor traffic on various airports across Germany nearly in real time [6]. The tool has been used as a fundamental source for understanding, verifying and testing operational ATC procedures. Its advantages are the ability to monitor live traffic as well as visualizing past traffic in user defined time intervals. Usage of the Stanly Track application has increased the understanding of operational deviations within a wider time interval. Additionally it was possible to visually confirm interview data.

C. Recorded Controller Commands

Within the scope of DLR's 4D-CARMA project, a variety of test series have been conducted in cooperation with the German air navigation service provider DFS. This includes approach traffic simulations with real air traffic controllers. The data derived from these tests (speech recordings, radar data and command transcriptions) were then used as a method to validate the results of the developed algorithm.

IV. Mapped Exclusion Method

The conducted research has shown a way of converting operational observations into sets of rules in order to ultimately limit the number of possible commands for each approaching aircraft. Based on the aircraft's parameters, such as position, altitude, airspeed and heading, the amount of possible commands can significantly be reduced. The implementation of this concept is called Mapped Exclusion Method (MEM). The MEM is based on the idea that future ATC commands are primarily influenced by the aircraft's position and configuration. For example, an aircraft at high altitude far away from the field, will most likely not receive instructions to reduce to minimum clean speed or to contact the airport's Tower, as this is only applicable to aircraft closer to the airport.

Vice versa aircraft on final approach are not expected to receive last minute heading changes or altitude clearances, as they are already following the published horizontal and vertical approach profile.

A. Controller Command Prediction – a Function of the Aircraft State and the Traffic Situation

The prediction of controller commands can be viewed as a function that maps the state of the aircraft to a list of possible controller commands. From the controller interviews the following list of essential attributes of the aircraft state was compiled:

1. Position: latitude *lat*, longitude *lon* and altitude *alt* and
2. Movement vector: heading (track angle) *hdg*, ground speed *spd* and rate of climb/descent *rocd*

Obviously the prediction function MEM differs from airport to airport, it will change as the airport and its surrounding airspace evolve, and will even have to be adapted when ATC changes internal procedures. Therefore, an implementation of MEM needs to be split into a generic **algorithm** and airport specific **model** data. The airport Model *AM* would be a further parameter to MEM. Therefore, the MEM-function has the following signature:

MEM(AM, lat, lon, alt, hdg, spd, rocd,)→ list of commands

With this abstraction the prediction of controller commands is as simple (or as hard) as programming the MEM-function: A computable representation of all the commands a controller might give to an aircraft at a specific point in airspace in a specific situation.

B. Simple Approach: Sampled Map

The first idea for getting a model was to sample the input-space of MEM and build a database of all possible controller commands for every possible input data set. If just the three-dimensional airspace around the airport was to be sampled, this would be achievable by drawing pixel graphics for all altitude bands and all commands. Each graphic file would be a cartographic map showing where a command is expected. On one hand the algorithm that interprets the database would be trivial, since it just would have to look up the right pixel in the right pixel map. On the other hand, drawing such a huge pile of pixel maps would be a big effort, and maintenance would be nearly impossible.

So a more complex algorithm was chosen to make modeling of approach procedures simpler or even possible at all.

C. Intuitive Approach: Mapped Exclusion Method (MEM)

During interviews the controller pointed to his radar screen, explaining: “At this point we only issue command *x*. At that point we only issue command *y* if the aircraft is higher than *z* feet.” From this intuitive description a MEM-function as the following can be sketched:

The MEM-function starts with the assumption that all commands are possible all over the approach sector, i.e. the set of possible commands for an aircraft contains all thinkable commands. The regions the controller points to, can be modeled as polygonal areas on a map. Areas have attributes such as **precondition**s e.g. “alt > 50” and **restrictions** e.g. “Direct('TEBRO')”. If an aircraft is located within an area and all preconditions are met, the set of possible controller commands is limited by the restriction-attributes of that area, i.e. the previous set of possible commands is intersected with all restrictions of the area. The aircraft is checked against all areas and the set of possible controller commands is reduced by the restrictions of applying areas. As a result the list of possible commands is cut down to just the few commands that can be expected in real operations. With this approach the data model of the MEM function is a representation of the controller working style.

D. Expansions to Controller Command Sets

So far MEM only excluded commands via the restriction attributes of areas. During the interviews the controllers often stated that in an area they explicitly give a certain command, while everywhere else this command is prohibited. This is useful when predicting special purpose advisories such as the ILS clearance. The ILS clearance command should be prohibited all over the approach sector using a huge area with a comprehensive restriction. Just close to the final approach there needs to be areas with the expansion-attribute stating “Special('ILS Clearance')”.

Therefore, the algorithm has to be extended by the rule: After all restrictions of all met areas have been applied, the resulting command set has to be extended by all extension attributes of met areas.

Finally, the implementation of restriction and expansion sets differs a bit from common mathematical set algebra:

restriction and expansion sets only interfere with classes of commands they specify commands for. The restriction set "Speed 140, Speed 150" only makes a statement on speed commands. If for example heading commands are also possible for the aircraft, than this restriction set does not interfere with them. This way restriction sets can be stated in the compact way as used in this paper.

E. Sum up the MEM-Algorithm

The MEM-algorithm to calculate the possible commands of a class (e.g. descend, reduce, heading) can be summarized as:

1. Start with all possible commands of all classes
2. **intersect** with all **restrictions** of the respective class of all met areas
3. **unite** with all **expansions** of the respective class of all met areas

V. MODELLING AREAS FOR MEM

As introduced before, the Mapped Exclusion Method's model consists of a set of areas. An area is a georeferenced polygon with a set of attributes. Manually coding the areas, especially the coordinates of the polygons would be a cumbersome task. The best way would be to draw the areas directly on a georeferenced air-navigation map. This is the general domain of Geographic Information Systems (GIS) [23]. While general purpose GIS have a wide range of applications and features, here only the "polygon drawing and annotation" feature is needed.

A. Java OpenStreetMap Editor (JOSM)

As GIS the mapping application of the OpenStreetMap (OSM) [3] project was used. The collaborative OSM-Project is drawing a free open source world vector map. The data is collected and annotated by volunteers using GPS devices. Any data that volunteers find worth mapping should be and is mapped. The data model of OSM is very simple and flexible; it features georeferenced nodes and paths of nodes. If the first and the last node of a path are identical, the path is a polygon. Every node and path (or polygon) can have arbitrary free-text key-value-pairs as attributes. These polygons with the key-value-pairs are a perfect data-model for MEM.

To edit such data the OSM-Project developed a simple but specialized GIS-Application: the Java OpenStreetMap Editor (JOSM) [22]. With this editor the annotated polygon-areas can easily be edited, individually or in combination. As background for drawing the OpenStreetMap or any georeferenced bitmap can be used. For controller command prediction the approach charts from Eurocontrol Aeronautical Information Database EAD [9] were converted and used as background (see Fig. 2).

With JOSM the annotated areas were created according to the controller interviews. Instead of streets and land-use-areas, polygons of controller intentions were modeled. Instead of standard key-value-pairs from OSM for highways or forests, sets of controller commands were modeled with preconditions and restrictions. Instead of uploading the data to the public OpenStreetMap servers the areas were saved as osm-files. The osm-files are simple xml-files containing the georeferenced nodes, polygons and the key-value-pairs.

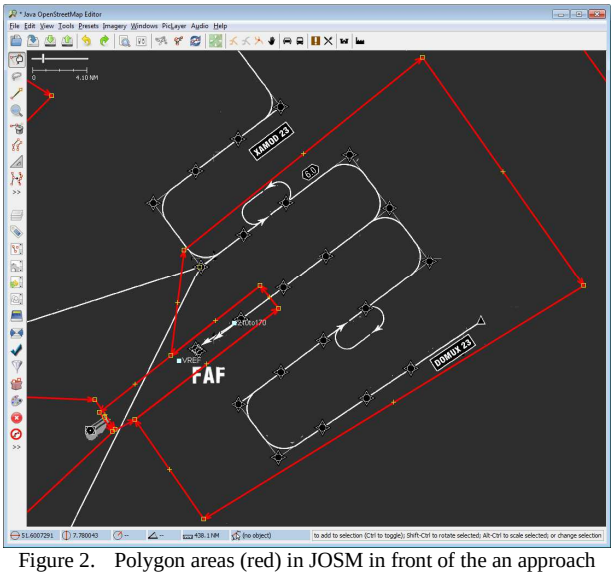

Figure 2. Polygon areas (red) in JOSM in front of the an approach chart of Düsseldorf (gray)

B. The Implementation of the Mapped Exclusion Method Algorithm

The osm-files containing annotated areas with controller command sets have to be read in. Then the areas have to be inferred with incoming radar plots of aircraft to perform the exclusion of impossible controller commands. Within this process the preconditions of areas have to be evaluated and restrictions and expansions have to be applied to command sets.

1) Preconditions are Embedded Python Programming Language Code

A precondition is a conditional statement on the input variables of the MEM-function. The input variables have to be combined into terms as in usual programming languages; e.g. "hdg > 35 and hdg < 180". To keep the algorithm generic and to keep all model information in the maps, these precondition terms have to be stated within the .osm-file as an annotation to the areas (see Fig. 3). The simplest way to evaluate the preconditions is to write them in an interpreted language and call the interpreter with the condition term.

The interpreted dynamically typed language Python [21] was chosen for this purpose. To keep the project simple, the whole MEM function was implemented in Python. For the MEM-function Python has the following useful features: Python is easy to learn, emphasizes on complex data structures, is embeddable into C++ code (such as the AMAN 4D-CARMA), has libraries for geographic computations as well as XML-parsing and ships with a unit-testing framework.

When interpreting preconditions, the MEM-function sets up a Python context with the aircraft's state variables (*lat*, *lon*, *alt*, *hdg*, etc.) and evaluates the precondition within the context. If the evaluation returns true the precondition is met and the restriction is applied to the controller command set.

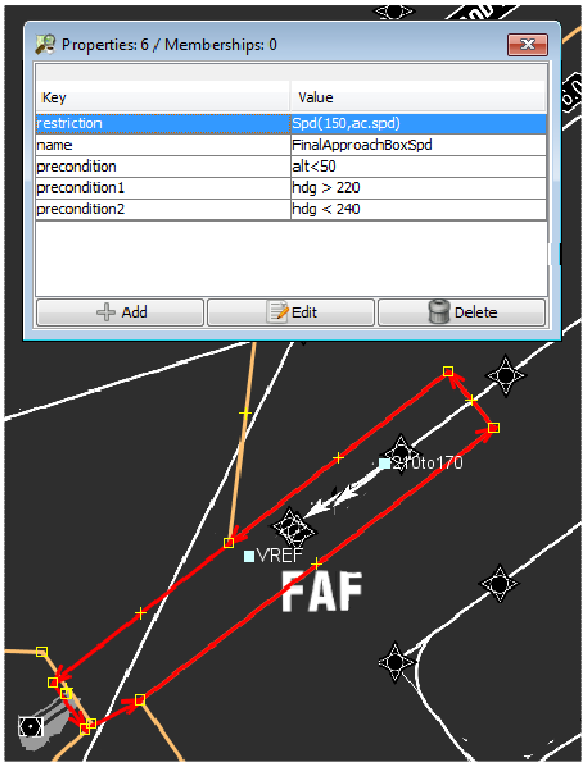

Figure 3. Polygon area property editor with examplatory preconditions and an expansion.

2) Restrictions are Embedded Python Programming Code

The restrictions of an area are a set of controller commands. If an area (and its preconditions) is met, the restrictions of this area are intersected with the possible controller commands for that aircraft. Writing down the restriction set by enumerating every single command is error prone. In many cases the restriction set will also depend on the aircraft state. So restrictions are implemented as Python expressions that return sets of controller commands. Examples of valid restrictions are: "Direct('TEBRO XAMOD')", "Hdg(350, 10)", "Hdg(HdgTo('REGNO'), HdgTo('DL450'))" and "Spd(150, ac.spd)".

Direct(), Hdg(), Spd() as well as Alt() and Special() are predefined helper functions that return sets of controller commands:

- "Direct('TEBRO XAMOD')" returns direct commands to the specified waypoints TEBRO and XAMOD.
- "Hdg(350, 10)" returns all heading commands from 350° to 10°, which are "Heading 350", "Heading 0" and "Heading 10".
- "HdgTo('REGNO')" returns no command set itself but is a helper for other functions. It returns the heading form the current aircraft position to the waypoint REGNO.
- So "Hdg(HdgTo('REGNO'), HdgTo('DL450'))" returns all heading commands that sent the aircraft somewhere between the waypoints REGNO and DL450.
- "Spd(150, ac.spd)" returns a set of speed commands in the range of 150kts up to the current aircraft speed.

Using Python code in the restriction-attributes of areas introduces a huge flexibility in modeling possible controller commands. While predefined helper functions as stated above, keep models readable and adoptable to further changes.

Of course, the expansion-attributes are embedded Python code as well.

VI. Experiments and Results

When developing experimental software based entirely on recent research, testing the actual result obviously becomes an important issue. In terms of the practical aspect of the MEM, it would have been of huge value to record real world ATC radio transmissions and compare them to the software output in order to note any discrepancy. There are, however, certain legal limitations to this matter since German telecommunication laws do not allow the recording, publishing or analysis of ATC radio transmissions for the general public [12].

A. Test Methods

Therefore, two different test methods have been established. The first method is based on Stanly Track observations and uses real rather than simulated input data. Since each test scenario had to be individually designed by hand, collecting sufficient test data would have been a painstaking way of validation. Therefore, the second test method was based on existing test runs of DLR. This data were derived from simulations with real controllers guiding simulated traffic. Together both tests form a satisfying way of assessment and present interesting results. In order to assess the quality of the final result, three values are introduced.

1) Context Error Rate CtxER

The context error rate *CtxER* is the relation of missed instructions n_{MISSED} to given instructions n_{GIVEN} expressed as a percentage ($CtxER = n_{MISSED}/n_{GIVEN}$).

It equals 0% if all instructions have been previously predicted and equals 100% if no given instruction has been predicted.

2) Context Reduction Rate RR

The context reduction rate *RR* is the average quotient of momentarily excluded commands $n_{EXCLUDED}$ and the total number of commands n_{TOTAL}. ($RR = n_{EXCLUDED}/n_{TOTAL}$).

The context reduction rate shows by which factor the amount of possible commands is reduced. It equals 0% for all commands possible at all times and equals 100% if no commands being possible at all. The initial unreduced number of commands possible for each aircraft is 131.

B. Description of Test Environment

The test input data consists of radar data from previous test runs. These simulations have been conducted featuring real air traffic controllers and their corresponding given commands, all within Dusseldorf airspace.

The simulated radar data of incoming traffic were then sent to the prediction module which created the command context. The created context was then compared to the actual given commands which allowed calculating the relevant context reduction rate and context error rate.

C. *Test Results*

The average context error rate is 0.9%, meaning 99.1% of commands issued by the controller in the test cases, have been included in the prediction set by the algorithm (982 out of 991). Analysis of the test context showed that in most cases the missed commands include unusual procedures that are only rarely used, e.g. a handover to tower as far as 25NM from runway threshold instead of the usually observed 8-14NM. Although it is possible to increase the percentage of recognized commands even further, it is questionable if this is truly desired, as such practice comes with a degradation of the context reduction rate. Ultimately it is a question of application which of the two values is considered more important for the overall result.

The average context reduction rate in this case is 77.3% , meaning that only 22.7% of the total context remains within the algorithm's output (204,308 out of 899,577 commands). It is important to consider that the total context has already been initially reduced. This includes the reduction to only relevant (and expected) callsigns, as well as an initial set of commands containing about 131 commands per aircraft.

VII. NEXT STEPS

On the one hand the previous chapters have shown how to generate a limited set of possible controller commands to improve the performance of ASR. On the other hand an assistant system like an AMAN can benefit from the additional sensor of the speech recognizer. The next steps will be aimed at a further increase of the capabilities of the reduction algorithm.

In an effort to achieve said increase, a statistical analysis of the given commands, their amount and their location has been conducted. Figure 4 shows the respective descent commands and the number of times they have been observed in the test data (FL 30 -- 50 means altitude 3000 -- 5000 feet).

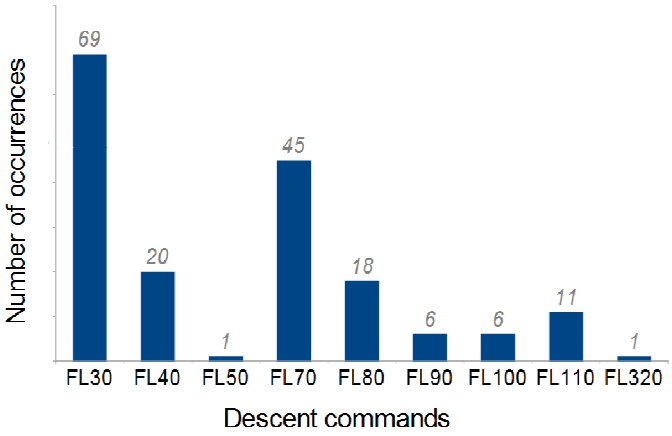

Figure 4. Frequency of descent commands

It becomes clear that some commands are issued very frequently, whereas other are rarely or even never issued. Additionally the location of the respective airplane, at the time it was issued a certain command, was recorded and the result analyzed. Certain commands are typically issued at certain geographical locations, forming command clusters of varying size and shape. Figure 5 shows such clusters on the final approach to Dusseldorf.

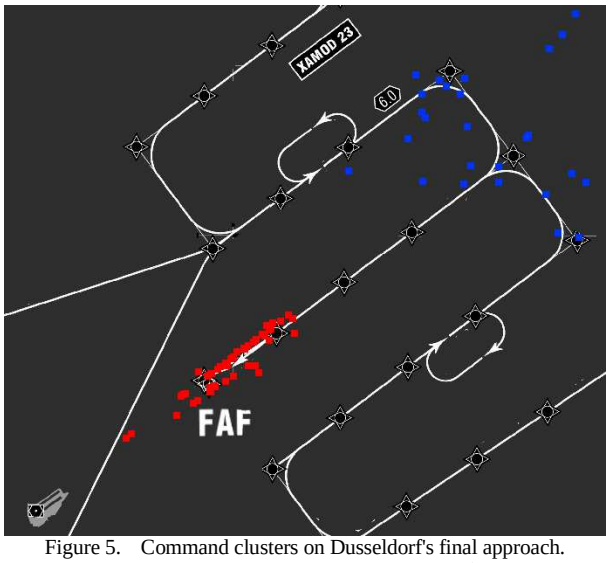

Figure 5. Command clusters on Dusseldorf's final approach. Predictable handover commands (red) are located on final approach. Accordingly ILS clearances (blue) are found further out.

As a next step the algorithm shall be able to issue a probability value along with each possible command. This value should indicate the overall probability for each command along with its respective callsign. The application of probabilities to the context output should further increase the context reduction rate with minor to none effect on the context error rate.

Further increase of the context reduction rate can also be achieved by using AMAN data as an input. These data can e.g. be used to remove already issued ATC commands from the context, as commands are usually only issued once.

Ultimately the improvement in connection with the speech recognition software shall be systematically evaluated and the factor of such improvement determined.

VIII. CONCLUSIONS

One reason for the huge difficulties of introducing higher levels of automation in ATM refers to the intensive use of spoken language. Controllers and pilots communicate their intentions via voice communications, whereas the assistant systems rely on (radar) sensor information without any knowledge of operator’s intentions. This fact may create misinterpretations and may lead to failures and further on to the rejection of automation.

One solution for these problems is the introduction of speech recognition following the "motto": Only systems that can listen may be in the loop! If the recognition rate is good enough speech recognition could be even trigger the introduction of higher levels of automation. This results in the

need to use the current situational knowledge of the assistant system to create dynamic speech hypotheses, which enables high recognition rates. We showed that we can reduce the number of theoretically possible commands by a factor of 4 if we use the current traffic situation for deducing the command hypotheses. Our error rate is below one percent, i.e. less than one of 100 commands given by the controller is not in our prediction set. This deduction is, however, not a trivial task because approach controllers do not just stick to a few standard arrival procedures, but choose from a wide variety of strategies. These strategies arise from published charts, written internal work orders, informal knowledge and personal experience.

Our formalization to predict the set of possible controller commands is the Mapped Exclusion Method (MEM), which is based on geo-referenced areas with attributes. If an aircraft is within an area and meets the areas' precondition attribute, the set of expected controller commands is modified according to the restriction and expansion attributes. The areas are modeled with the open source map editor JOSM of the OpenStreetMap-Project. Precondition, restriction and expansion attributes of areas are embedded within the GIS-data as Python programming language code. This offers the necessary flexibility in modeling controller strategies and even ATC experts without knowledge in programming languages are able to model the approach area of their airport.

REFERENCES

[1] ACARE, "Realizing Europe's vision for aviation – Strategic Research & Innovation Agenda (SRIA)," Vol. 2, Sep. 2012.

[2] AMIDA, "Augmented Multi-party Interaction with Distance Access," http://www.ercim.eu/activity/projects/amida.html, 2009.

[3] J. Bennett, *OpenStreetMap: Be your own Cartographer*. Packt Publishing 2010 ISBN 978-1847197504

[4] S. Ciupka, "Siris große Schwester erobert die DFS," in German, Engl. title „Siris big sister captures DFS," transmission, Vol. 1, 2012.

[5] J. M. Cordero, N. Rodríguez , J. M. de Pablo, M. Dorado, "Automated Speech Recognition in Controller Communications applied to Workload Measurement," Third SESAR Innovation Days, 26. – 28. November 2013, Stockholm, 2013.

[6] DFS: Stanley Track http://stanlytrack2.dfs.de/

[7] K. Dunkelberger, and R. Eckert, "Magnavox Intent Monitoring System for ATC Applications," Magnavox, 1995.

[8] Eurocontrol, "All Clear? The path to clear communication. ICAO Standard Phraseology A Quick Reference Guide for Commercial Air Transport Pilots," http://www.skybrary.aero/bookshelf/books/115.pdf, 2011.

[9] *European Aeronautical Information Service Database – EAD* Eurocontrol, Brussels, http://www.ead.eurocontrol.int

[10] European Commision, "Flightpath 2050, Europe's Vision for Aviation Maintaining Global Leadership & Serving Society's Needs -- Report of the High Level Group on Aviation Research," 2011.

[11] FAA, "2012 National Aviation Research Plan (NARP),", March 2012.

[12] Federal Republic of Germany: "Telekommunikationsgesetz,§88, §89", http://www.gesetze-im-internet.de/tkg_2004/index.html\

[13] C. Hamel, D. Kotick, and M. Layton, "Microcomputer System Integration for Air Control Training," Special Report SR89-01, Naval Training Systems Center, Orlando, FL., USA, 1989.

[14] N. Hasevoets, and P. Conroy, "AMAN Status Review 2010," Eurocontrol, Edition number 0.1, 17 December, 2010.

[15] H. Helmke, H. Ehr, M. Kleinert, Fr. Faubel, D. Klakow, „Increased Acceptance of Controller Assistance by Automatic Speech Recognition," ," 10th USA/Europe ATM R&D Seminar, 9. - 13. June 2013, Chicago, Illinois (USA), 2013

[16] H. Helmke, R. Hann, M. Uebbing-Rumke, D. Müller, and D. Wittkowski, "Time-based arrival management for dual threshold operation and continous descent approaches," 8th USA/Europe ATM R&D Seminar, 29. Jun. - 2. Jul. 2009, Napa, California (USA), 2009.

[17] M. Hössl, "Predicting Air Traffic Control – Anticipation of Approach Controller Intentions at the Example of Düsseldorf Airport," University of Applied Sciences Bremen, September 2013

[18] "Air Traffic Management – Procedures for Air Navigation Services", ICAO Document 444, Fifteenth Edition, 2007

[19] D. Jurafsky and J. H. Martin, "Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition," 2nd edition. Englewood Cliffs, NJ, USA: Prentice-Hall, 9th Feb. 2008.

[20] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in: Soviet Physics -- Doklady 10.8, Feb. 1966.

[21] M. Pilgrim, "Dive Into Python" Apress 2004 ISBN 978-1590593561, http://www.diveintopython.net

[22] F. Ramm, J. Topf, S. Chilton. "OpenStreetMap: Using and Enhancing the Free Map of the World". UIT Cambridge. 2010, ISBN 978-1906860110

[23] M. J. de Smith, M. F. Goodchild, P. A. Longley, "Geospatial analysis: A comprehensive guide to principles, techniques and software tools (2nd ed.)". Troubador, UK 2007 ISBN 978-1848761582, http://www.spatialanalysisonline.com

[24] D. Schäfer, "Context-sensitive speech recognition in the air traffic control simulation," Eurocontrol EEC Note No. 02/2001 and PhD Thesis of the University of Armed Forces, Munich, 2001.

[25] T. Shore, Fr. Faubel, H. Helmke, D. Klakow, "Knowledge-Based Word Lattice Rescoring in a Dynamic Context," Interspeech 2012, Sep. 2012, Portland, Oregon.

[26] C. Weinstein, "Opportunities for Advanced Speech Processing in Military Computer-Based Systems," Proceedings of the IEEE, Vol. 79, No. 11, 1991.