DOI: 10.13009/EUCASS2019-302

An Automation Language for Increasing Repeatability in Scaled Flight Testing of New Aircraft Configurations

R. Rocchio F. Corraro** G. Di Capua*** L. Garbarino**** N. Genito******
CIRA – Italian Aerospace Research Centre
** r.rocchio@cira.it*
*** f.corraro@cira.it*
**** g.dicapua@cira.it*
***** l.garbarino@cira.it*
****** n.genito@cira.it*

Abstract

Current commercial autopilots allow full turnkey solutions for remote piloting of unmanned aircraft (including scaled ones), optimized for the most common flight operations and with limited possibility of being adapted for flight testing of new aircraft configurations.
This paper describes an autopilot specifically designed for supporting flight testing on unconventional aircraft configurations, allowing integration of custom advanced control laws architectures and strategies, logics, automated flight modes and execution of special flight test procedures. The paper also includes results of validation tests performed on an HW-in-the-Loop test-rig with a vehicle real time simulator.

1. Introduction

Essentially built upon the positive experience of the Clean Sky SFWA (Smart Fixed Wing Aircraft) project, the Clean Sky 2 LPA (Large Passenger Aircraft) operational activities started in July 2014 in all three major work packages also called "Platforms". Platform 1 (Advanced Engine and Aircraft Configurations) will provide the development environment for the integration of the most fuel efficient propulsion concepts into the airframe targeting next generation short and medium range aircraft (A/C). In Platform 1, WP1.3 (Integrated Aeronautics Demonstration Platform) aims at validating scaled flight testing as a viable means to de-risk disruptive A/C technologies and A/C configurations to high TRL.
In this framework, CIRA (Italian Aerospace Research Centre) is the leader of the WP1.3.5 (A/C Guidance and Control) aiming at developing an A/C Guidance, Navigation and Control (GNC) system, and a dedicated testing framework for supporting the dynamically scaled vehicle flight demonstrations.
As already shown in [1], scaled flight testing has been a way of demonstrating new aeronautic technologies since many years. Because of the typical small dimensions of the A/C used for this kind of testing, the guidance and control equipment used for support scaled flight tests are the same of remotely piloted A/C or unmanned aerial systems. In these fields, only quite recently some commercial products have been introduced on the market. These products offer a "full turnkey" solution for remote piloting of an unmanned A/C from a small ground station [2] [3] [4]. The key advantages of such products rely on their compactness, affordability, configurability and easy-to-use HW/SW but they have many limitations when used for unconventional A/C configurations and/or when special flight test procedures shall be executed.
Instead of using standard available autopilots (AP), in this project an HW/SW framework for GNC of A/C in the class of 150Kg Take-Off weight is proposed. Such framework not only includes basic features of the above mentioned products, but also introduces new characteristics specifically designed for supporting scaled flight testing on unconventional A/C configurations, allowing implementation of advanced control laws architectures and strategies. Moreover, to facilitate flight tests, the framework integrates a dedicated SW function for automated flight tests, so avoiding, as much as possible, manual piloting and increasing test condition repeatability.
An additional relevant characteristic of such framework is a functional and SW architecture that, even if the specific flight tests planned in this project do not need a high level of autonomy [5] [6], is already designed to be compatible with the highest autonomy levels. This goal is pursued by developing a modular and scalar architecture and by adding the capability to execute complex automatic functions based on a simple flight instruction list defined by the remote pilot or the ground operator. To this end, have been developed a programming meta-language to allow performing mission tasks with better efficiency and effectiveness and, for the purpose of flight testing, with increased repeatability.

DOI: 10.13009/EUCASS2019-302

Some words have been defined corresponding to all the possible states of the abstract state machine and to the parameters associated to the functions performed in those states. Moreover, a syntax has been defined to allow interpretation and conversion of sentences into proper GNC actions for executing the required operation.
One of the parameters that is included in the sentences of the defined language (i.e. the instructions) indicates which control module combination, among the available ones, should be used in performing that operation. This possibility maximizes the benefits deriving from a scalable, flexible and modular architecture and allows automatic testing, in few steps, of dedicated advanced control laws.
To this end, the proposed AP comes with a software framework that, following a few, simple steps and using a specifically developed library, allows the users integrating new flight test functions and/or control laws to the basic SW and let include them in the instruction set.
Since the early phase of this work, the automation language has been designed considering the future implementation of a voice engine software, a step beyond that allows interpreting and translating high level vocal commands by an operator in a set of instructions to the AP system.
Eventually, the proposed automation language, integrated with the development framework, eases automated flight tests of new scaled A/C configuration and allows increasing test condition repeatability avoiding manual piloting as much as possible. This can also be considered a step towards an increased autonomy in UAS operations as it improves and sets the grounding for a better a thorough Human Machine Interaction (HMI) and, consequently, for developing a social interaction in a socially assistive robotics environment.
The first part of the paper includes a system SW architecture overview with a particular attention on the GNC functions. Subsequently, the proposed automation language will be described and the firsts words created in the presented programming meta-language will be shown illustrating the functionality specifically developed for flight testing. Moreover, the HW-In-the-Loop (HIL) test-rig used for verification and validation will be presented with some detail of the simulation test rig and the monitoring cockpit display developed for the ground pilot/operator. Finally, some Real Time (RT) simulation test results will be discussed.

2. Autopilot SW architecture

As above mentioned, one of the aim of the subject research project was to support flight testing for unconventional A/C configurations and to facilitate the execution of special flight test procedures with accurate and repeatable conditions. To this aim, a specific HW/SW architecture has been designed that has the required flexibility and automation features.

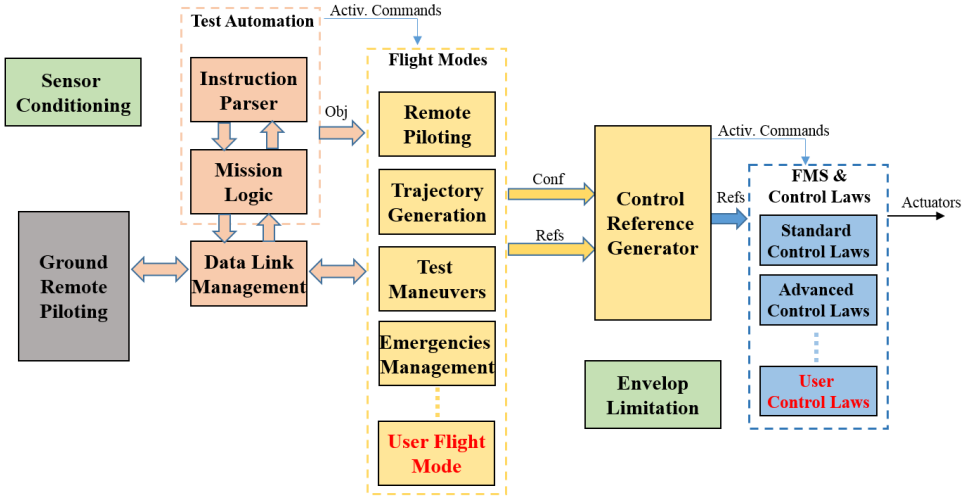

Figure 1: AP's GNC functional architecture

For achieving the above goals, it is crucial in the AP having an effective GNC function, whose architecture is shown in Figure 1. Its main modules are:

- *Sensor Conditioning*: Analyses all the sensors' measures and diagnostics, apply appropriate signal conditioning and corrections, and eventually computes the validated vehicle state, integrity and accuracy information and advanced diagnostics;
- *Envelop Limitation*: Generates the limitations on vehicle key parameters (velocity, accelerations, etc.) taking into account the health state of A/C and the current flight conditions;

- *Test Automation*: Implements the mission management function. It is composed by a *Mission Logic* state machine, which designates the flight mode in command, and an *Instruction Parser*, which manages the instruction list to be executed. In addition, it contains an "*Event Generator*" module that creates the trigger for the state machine state changes depending on the remote pilot commands, the A/C state and the current instruction in execution and a "*Selector Manager*" that actually activates the required flight mode;
- *Flight Modes and Control Reference Generator*: This container module includes several flight modes and module for selecting the right references and configuration data as commanded by the Test Automation;
- *FMS* & *Control Laws*: implements all the functions and algorithms needed to generate and track a trajectory, defined through waypoints, and to control the flight of A/C. This module is modular and on-line configurable thanks to special selector modules. They allow a completely customizable routing of the control references generated by current active flight mode.

To achieve adaptability to unconventional A/C configurations, the proposed architecture allows adding custom flight control lows and/or custom 'flight modes'. The flexibility, obtained through a fully customizable reference selection approach, makes the integration of such custom modules very easy. Moreover, to facilitate their implementation, the development framework also includes a library of basic guidance and control modules (e.g. customizable PID controller, filtered bumpless, etc.) and a procedure for efficiently and reliably generating code deployable on the on-board avionic HW.
To facilitate special flight test procedures, the on-board AP was provided with an on-line configurable standard control laws, a Flight Management System (FMS) and a "Test Maneuver" flight mode to generate predefined maneuvers (e.g. step, singlet etc.) on any available single variables and using any available control lows.

3. Flight Test Automation Language

Set in a wider plan to create a High Autonomous Mission Management (HAMM) system, this work describes its initial phase in the creation of a mission automation language, to achieve the above stated goal of enabling accurate and repeatable flight tests.
An HAMM system is responsible of all the aspect concerning an autonomous mission planning in terms of defining an action sequence to accomplish a given set of objectives and constrains in an optimal way. In a system like this, the operator role is to define objectives and (part of) constraints and to manage and supervise the mission.
Due to the lack of involvement by the pilot in command and control tasks, being those executed by the vehicle in autonomy, the interactions between the Ground Control Station and the A/C are limited to high level instructions.
A key function of a HAMM is to extrapolate and formalize the objective of the mission (e.g. position and characteristics of the target to reach). To this aim, the mission instruction interpretation have to be done through an algorithm based on a formal language.
Different standards already exists for the representation of the information that define Unmanned Aerial Vehicle (UAV) mission's operations. Two of them that include command and control messages are STANAG 4586 [9], a NATO standard only used for unmanned A/C, and JAUS [10], used, instead, for all kind of vehicles.
The STANAG 4586 protocol is a standard that defines architectures, interfaces, communication protocol and messages format to manage a complex operative scenario composed by a multinational group of UAVs. It defines standard interfaces, substantially composed by a set of messages, on which the communication between an UAV and its control station could be based. Its second version has been developed so to promote interoperability among one or different control stations, UAVs and information from the ground network in complex mission scenarios.
The JAUS (Joint Architecture for Unmanned Systems) is a SAE (Society of Automotive Engineers) standard that defines a communication protocol for unmanned vehicles systems in order to achieve uniformity and consistency and to allow interoperability, interchangeability and modularity. JAUS uses a SOA (Service Oriented Architecture) approach to achieve the distributed command and control of unmanned systems in a net; the systems based on JAUS uses JSIDL (JAUS Service Interface Definition Language) [11] to formalize those services. In JSIDL any service is defined as a specific set of input and output, together with a stochastic automaton to manage how to receive, treat and send back the messages.
Inspired by those protocols, and considering the aim of the project, an HAMM system specifically aimed to increase repeatability in flight-testing has been developed. Indeed, this paper presents only the first words of such language, specifically designed to formalize instructions for performing flight testing, but also thought as a base for increasing the vehicle's system autonomy level.
An HAMM shall include, above other functions, a formal language to formalize the objective of the mission, an instruction parser to extrapolate, instruction by instruction, the objective of the entire mission, and a mission logic, to enable the proper flight module to achieve it.

DOI: 10.13009/EUCASS2019-302

There are several possible approaches for defining a language, as generative, recognition or denotational [7]. As in the majority of programming languages, the recognition approach was chosen. It assumes that all the sentences understandable by a predefined automaton belong to the language [8]. In the presented framework, the words composing the language corresponds to both the possible states of the mission logic state machine and to the parameters associated to the flight modes enabled in those states. Moreover, a syntax has been defined to allow interpretation and conversion of single instructions into proper GNC actions for executing the required operation.
Taking examples on FMS early patent [12] and on the already existing mission automation languages, presented in the previous chapter, an xml representation of the instructions was selected.
Due to the simplicity of the possible instructions, so far a single sentence approach has been chosen. All the currently available instructions have a target WP and a maneuver to be performed (between the possible maneuvers is also available 'none' that disable this second part of the instruction) meanwhile the vehicle is tracking the leg towards it. Figure 2 shows an example of how an instruction is represented.

```
<FlightPlan>
  <Name>A</Name>
  <Scope>0</Scope>
  <Waypoint>
      <Latitude>40.5</Latitude>
      <Longitude>17.42</Longitude>
      <Altitude>1000</Altitude>
      <IAS>33</IAS>
      <TrackAngle>-120</TrackAngle>
      <HorizontalSteeringMode>0</HorizontalSteeringMode>
      <hStep>0</hStep>
      <VerticalSteeringMode>0</VerticalSteeringMode>
      <vStep>0</vStep>
      <SpeedSteeringMode>1</SpeedSteeringMode>
      <sStep>0</sStep>
      <nMode>0</nMode>
      <lastWP>0</lastWP>
      <priorityWP>0</priorityWP>
      <MANOUVER_THRUST_DEF_MODE>1</MANOUVER_THRUST_DEF_MODE>
      <MANOUVER_LATERAL_DEF_MODE>1</MANOUVER_LATERAL_DEF_MODE>
      <MANOUVER_DIR_DEF_MODE>0</MANOUVER_DIR_DEF_MODE>
      <MANOUVER_LONGITUDINAL_DEF_MODE>1</MANOUVER_LONGITUDINAL_DEF_MODE>
      <TEST_MODE>0</TEST_MODE>
      <TEST_WAVE>5</TEST_WAVE>
      <delayTime>0</delayTime>
      <waveTime>0.1</waveTime>
      <waveAmplitude>4</waveAmplitude>
      <testTimeLimit>1</testTimeLimit>
  </Waypoint>
```

Figure 2 : Instruction xml definition

In the above figure, the first part of the instruction set contains information regarding the location of a WP and related capture properties (e.g. fly-by or fly-to both in horizontal and vertical channel in an independent way, etc.), while the second part, instead, contains instructions on the maneuver to execute (e.g. type of maneuver, variable, amplitude, timing etc.).
Through the maneuver's parameters of the instruction, it is also possible to define which control module combination, among the ones available, should be used in performing that operation. This possibility effectively uses the benefits of the proposed architecture in terms of scalability, flexibility and modularity and allows automatic testing, in few steps, of custom/advanced control laws.
Finally, to verify instruction lists and validate all the AP functions, a RT simulation test rig has been developed for performing simulated flight tests directly from the ground remote pilot station, as will be described below.

4. HW-in-the-Loop Simulation Test Rig

The reference vehicle architecture for integration of the proposed AP is shown in Figure 3. The proposed AP is depicted in cyan colours and is composed by two segments: the On-board Guidance, Navigation and Control (OGNC) System and the Ground Remote Pilot Station (GRPS).
Regarding the OGNC, a RT computer executing the SW described in the previous chapter, when in command generate the references for the A/C actuator so to execute the instruction sent by the pilot through the GRPS.
The GRPS allows the on ground pilot/operator to interact with the automated A/C. Moreover, the graphical HMI shows, using features available by the AP, information on the current instruction that the HAMM system is executing in a clear way and give an extensive overview of the A/C state (as shown in Figure 5). In addition, the GRPS also provide a cockpit OTW (Out Of Window) live streaming video coming from the on-board forward camera. Figure 4 shows the layout of the whole GPRS.
As already said, directly in the GRPS is also integrated a RT simulation facility composed by a vehicle RT simulator model and a copy of the OGNC allowing remote pilot training and SW-In-The-Loop (SIL) simulation to verify automation sequence, flight modes, control laws and mission execution.

DOI: 10.13009/EUCASS2019-302

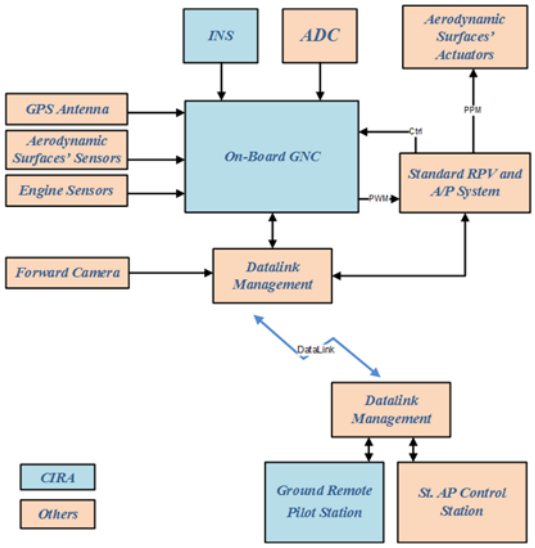

Figure 3 : HW system architecture

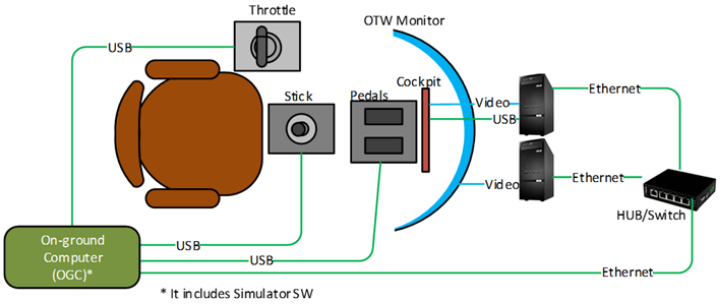

Figure 4 : Ground remote pilot station architecture

All above equipment and SW have been verified and validated in a dedicated test rig for performing RT HIL and Pilot-in-the-Loop testing.
Such test rig is substantially composed by a RT simulator that emulate all the equipment that interface the OGNC of the proposed AP and the GPRS. All the systems connected to the On-Board GNC in Figure 3 are simulated together with the datalink and the vehicle allowing a complete, HW and SW, validation of the developed AP.

Figure 5 : Human machine interface

DOI: 10.13009/EUCASS2019-302

5. Test Results

Using above described test rig, a verification and validation campaign has been performed, aimed not only to testing the automation language but also to validating the overall AP system and assessing effectiveness of its main features. Both offline and RT HIL simulation have been performed to verify logics, selection mechanism and automation in transition between different flight modes. After being successfully completed this phase, the proposed autopilot will be integrated in the final flight demonstrator for executing the flight campaign. Actually, two campaigns of flight tests are expected to be performed both in Holland and in Italy over the next two years on two different configurations of the flight demonstrator.
Just for example, in this section will be presented a specifically designed test to show the capability of the system to follow automatically a list of instructions. The executed instruction list (red triangles in the top image of Figure 6) is:

- WP1, horizontal and vertical fly-to, no maneuver;
- WP2, horizontal fly-by, vertical fly-to, altitude step, AP altitude controller on longitudinal, AP track controller on lateral, direct rudder on directional and AT IAS controller on velocity;
- WP3, horizontal fly-by, vertical fly-to, track step, AP altitude controller on longitudinal, AP track controller on lateral, direct rudder on directional and AT IAS controller on velocity.

The test results are shown in Figure 6. The top figure shows a top view of the test, on that the waypoints are represented as red triangles, green and red crosses indicate initial and ending points for maneuvers, the magenta circle is the starting point of the simulation and the green one the point in which the system take control on the A/C. Moreover, the bottom figures show reference and controlled variables (on altitude, track and IAS) during the maneuvers executed reaching WP2 (left) and WP3 (right).

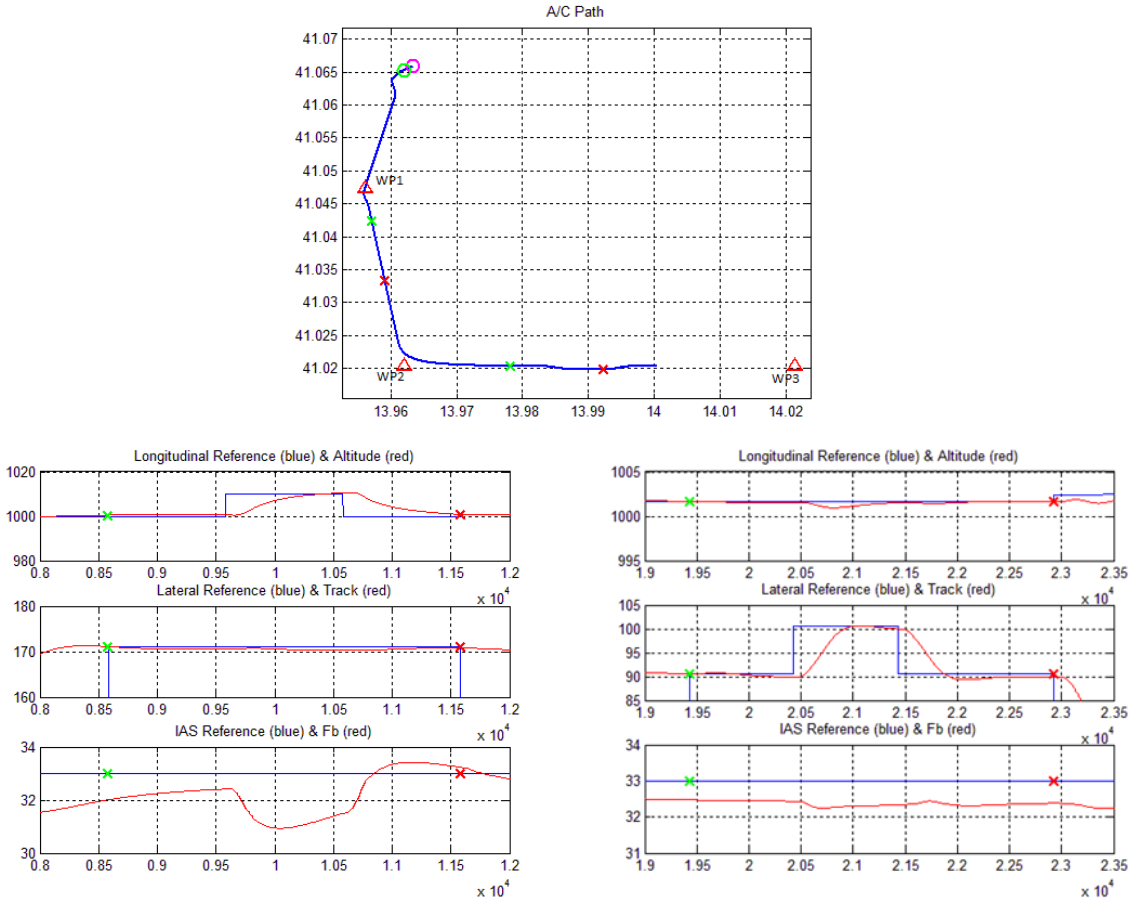

Figure 6 : Test results

6. Conclusions

In this paper, an automatic flight testing oriented AP has been presented. Its framework has been developed as an ad hoc solution, alternative to the commercial "full turnkey" AP already existing, for supporting the Clean Sky 2 dynamically scaled vehicle flight demonstrations requiring scaled flight testing on unconventional A/C configurations and of advanced control laws architectures and strategies. This goal has been pursued by developing a modular and scalar architecture enabling the integration of custom flight control lows and/or custom 'flight modes'. Moreover, to increase repeatability, the capability to execute automatic missions based on a simple flight instructions list was introduced. To formalize instructions, but also to increase efficiency and effectiveness, a programming meta-language has been developed together with a test automation interpret function. Finally, a complete simulation environment for SIL, HIL and Pilot-In-the-Loop simulations both in RT or off-line has been presented aimed to verification, validation and rapid prototyping.
The presented architecture has been developed to be compatible with higher levels of autonomy considering numerous future possible implementation, as the integration of a voice engine software aimed to interpreting and translating high level vocal commands by an operator. In addition, some additional advanced control law functionalities will be integrated for managing future versions of the scaled vehicle demonstrator.

Acknowledgments

The authors thank CSJU (Clean Sky Joint Undertaking) for funding the project Clean Sky 2 Large Passenger Aircraft Platform 1 WP1.3 under which the activities described in this paper have been carried out.

References

[1] E. De Lellis, et. al., "Design of a SW Framework for an Autopilot with Automated Test Capabilities on an Experimental Mid-Sized UAS", SciTech 2019
[2] Micropilot Products - http://www.micropilot.com/products.htm
[3] ArduPilot Products - http://www.ardupilot.co.uk/
[4] Open Pilot Products - http://www.openpilot.org/
[5] Beer, J. M., et al., "Toward a Framework for Levels of Robot Autonomy in Human-Robot Interaction", Journal of Human-Robot Interaction, Vol. 3, No. 2, Pages 74-99, DOI 10.5898/JHRI.3.2.Beer, 2014
[6] Proud, R. W., et al., "Methods for Determining the Level of Autonomy to Design into a Human Spaceflight Vehicle: A Function Specific Approach", NASA, 2003
[7] S. Greibach. Formal languages: origins and directions. Ann. Hist. Comput., 3, 1981
[8] Hopcroft, Motwani, Ullman. Introduction to Automata Theory, Languages, and Computation. Addison Wesley, 2007
[9] "STANAG 4586 NAVY (EDITION 2) - Standard Interfaces of UAV Control System (UCS) for NATO UAV Interoperability", 2007 NSA 1022(2007) NAVY/4586
[10] http://www.openjaus.com/understanding-sae-jaus
[11] "JAUS Service Interface Definition Language", SAE Aerospace Standard AS5684A
[12] Bodin, W.K., Redman, J.J.W., Thorson, D. C.: US20046813559B1 (2004).