2017 International Conference on
Unmanned Aircraft Systems (ICUAS)
June 13-16, 2017, Miami, FL, USA

Mixed Reality Simulation for Incremental Development of Multi-UAV Systems

Martin Selecký, Jan Faigl, Milan Rollo

Abstract— **Development of complex multi-robot systems requires time-consuming and expensive testing and, especially in a case of unmanned aerial systems, it aggregates risk of hardware failures and legal issues when operating more than one unmanned aircraft simultaneously. It is highly favorable to deal with most of the eventual design flaws and system bugs before the final field tests in a simulation where the risks are significantly lower. On the other hand, the fidelity of the simulation needs to rise as the system development approaches the final stages and since some phenomena are difficult to be modeled precisely, a partial embodiment of the simulation in the physical world is necessary. In this paper, we present our results in the utilization of mixed reality simulation for incremental development of multi-UAV systems. We present three use cases where this method was used for development of various systems to show its versatility: (i) an unmanned system consisting of heterogeneous team of autonomous unmanned aircraft; (ii) a system for verification of collision avoidance methods among fixed wing UAVs; and (iii) a system for planning collision-free paths for light-sport aircraft.**

Fig. 1. Aerial vehicles used in practical deployment of the proposed method

I. Introduction

Development and verification of algorithms for command and control of heterogeneous robot teams is a complicated task since there is a lot of complex issues that need to be taken into account [1]. Some of them, like complex interactions among the team members, can be modeled with the help of multi-agent simulations and software simulations [2], [3]. However, since the goal of the development process is to deploy the algorithms in the real-world, software agent simulations by themselves are no longer appropriate. Effects of real world phenomena like those of the weather, communication issues, sensor and actuator errors or limited computational resources, which are difficult to model on high fidelity levels, need to be verified on real hardware assets.

Experiments were done in a virtual world and in the real world, both have their limits and justifications [4]. Simulations are much easier to set up and repeat; they can be easily used to model a basic functionality of the developed system and allow quick observation of the results. Simulations can be used to study effects of various environmental phenomena by modifying only specific environmental conditions and fixing others and thus removing possible sources of noise. Experiments in virtual environments can significantly save costs in cases of malfunctions and accidents.

On the other hand, it is challenging to incorporate all sources of inputs from the real world for realistic modeling of the behavior of the developed system. That is why the real world experiments should be an obligatory step for obtaining realistic results in later stages of development of a robotic system and for validating the robotic software's robustness before it is deployed in real missions.

Mixed Reality (MR) simulations [5] present the world where both the virtual and real objects and the virtual and real entities can co-exist and interact in real time. The MR concept allows the system developers to get more insight into the behavior of the entities (e.g., by visualization of their inner states) and to perform much cheaper and safer experiments with a part of the system being real and the other part virtualized. MR simulations also relieve simulators from recreating complete worlds since the simulation occurs partially in the real world where certain phenomena, such as noise and complex physics, do not need to be modeled.

A. Overview and Contribution

This paper reports on practical usage of the method for incremental development of HART (for human, agent, and robot teams) applications originally introduced in [1] and multilayer architecture for incremental development of complex unmanned aerial systems (UASs) [6]. The method and architecture have been employed in the development process of different aviation applications showing the versatility of the methods for UASs in research and development projects funded by Czech Ministry of Defence, U.S. Air Force Research Lab, and U.S. Army CERDEC, and also utilized for the development of an assistant system for light-sport aircraft pilots. An overview of aerial vehicles for which the method has been utilized is shown in Figure 1.

Authors are with the Czech Technical University, Faculty of Electrical Engineering, Technicka 2, 166 27, Prague, Czech Republic {selecmar|faiglj|rollom}@fel.cvut.cz

978-1-5090-4494-8/17/$31.00 ©2017 IEEE

The first presented use case is an unmanned system consisting of a heterogeneous team of autonomous unmanned aircraft, which is capable of performing complex tactical missions such as team area surveillance, target tracking or critical infrastructure protection and performing dynamic mission reconfiguration in case of change of the mission or number of available assets. The second application is a system for verification of cooperative, communication-based collision avoidance methods among fixed wing UAVs, where the system was incrementally deployed on two Lockheed Martin's Desert Hawk III aircraft. Finally, the third application is a system for increasing flight safety of light-sport aircraft for which a combination of negotiation among aircraft, cooperative, and non-cooperative methods of collision avoidance and trajectory planning algorithms were used to recommend the best collision-free trajectories for pilot or autopilot of light-sport aircraft.

After relating the presented approaches to the respective research fields in Section II, the remainder of this contribution is structured as follows. Section III describes the proposed multilayer architecture used for the incremental development of UAS together with options for virtualization of individual system layers, and various development stages that every such a system should go through during its development. Section IV presents the developed aviation systems and shows how they utilize the multilayer architecture. In Section V, the experience, and lessons learned from the system deployment are summarized. Section VI gives a conclusion of the achieved results.

II. Related Work

Simulations are commonly used to develop complex systems since they can speed up the development process and save costs on hardware since they allow to early discover, isolate and correct potential implementation issues. Mutter et al. [7] mentioned two approaches to a simulator development: (i) the software-in-the-loop (SIL), where all system components (sensors, actuators, aircraft, etc.) are simulated using mathematical models and (ii) hardware-in-the-loop (HIL), where only some (or none) system parts are simulated and other elements are replaced by real hardware. Examples of simulators using SIL approach can be found in [8], [2]. HIL approach is taken, e.g., by Goktogan et al. [9], Day et al. [10], and Pizetta et al. [11] to name few approaches.

This paper describes a utilization of the HIL approach extended to take benefits of MR simulations for incremental development of complex UAS and assistant system for pilots of light-sport aircraft. Therefore, in this section, we report on the related work in these research areas.

As for approaches to develop and deploy complex multi-UAV systems, Burkle et al. [12] introduced deployment of control mechanisms for teams of hardware Vertical Take-Off and Landing (VTOL) micro-UAVs, Scerri et al. [13] designed a system for deployment of multi-agent technology for UAV coordination to an industrially developed applications, and most recently, e.g., Sanchez-Lopez et al. [14] presented a multipurpose system architecture for autonomous multi-UAV platforms with basic high-level planning features.

As for systems for light-sports aircraft operation, there are existing off-the-shelve autopilots that can hold level flight and change aircraft heading to the next waypoint. These control systems are still subject of research and improvement [15]. The actual trend in this field is equipping the aircraft with Electronic Flight Instrument System (EFIS) with "glass cockpit" that integrates avionic sensors into one digital instrument [16] that allows for adding further functions [17], [18]. Haberkorn et al. [19] studied options and efficiency of methods for cooperative and non-cooperative detection (TCAS – Traffic Collision Avoidance System, PCAS – Portable Collision Avoidance System) of other aircraft for collision prevention and there have been projects dealing with increasing pilot's awareness of surrounding air traffic [20], [21].

One of the first notions of MR simulations used for development of mobile robots was presented by Chen et al. [5]. The authors used MR simulation tool based on the 3D robot simulator Gazebo in a single robot scenario to demonstrate advantages of combining real entity and simulated objects for robotic system development. Later, in [22], the same authors showed that MR simulations can help to provide efficient testing, identify improvements to the UAV controller, and provide valuable insights into the UAV system performance that would be otherwise difficult to obtain in real world tests. Honig et al. [4] used MR simulations together with robotic simulators, showed that these simulations can provide many advantages for research and development in robotics, and supported their findings by use-cases with nano quadcopters.

One of the most recent approaches to formal dealing with the issues of development of complex robotic systems has been proposed by Jakob et al. in [1]. They came up with a concept of incremental multi-level development of complex systems by a use of mixed-reality test-beds. The authors defined levels of virtualization as the extent to which parts of the target application setup are replaced with synthetic computational models in a given test-bed configuration. They proposed to begin development with a fully virtual system with all agents purely simulated and move to the configuration with no virtualization level by iterating through the space of test-bed configurations in a direction that depended on the development cost of the particular iteration step.

The architecture presented here builds on ideas of [1] that were generalized for incremental implementation of HW subsystems in MR simulation system. It allows incremental development of systems of multiple autonomous entities and based on its generality, it provides a universal method applicable for the development of systems of various types, as it is shown further in this paper.

III. Architecture for incremental system development

The architecture has been first introduced in [6] and already used for development of multi-UAV systems consisting of fixed wing aircraft [23], [24]. It is a seven-

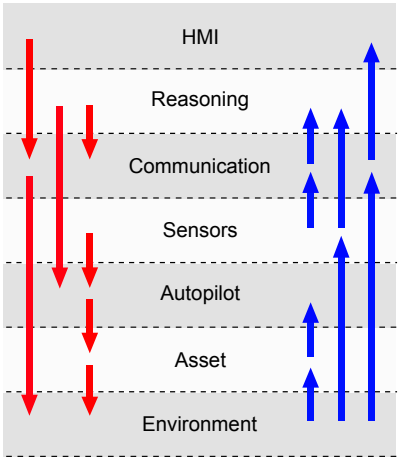

Fig. 2. Modular architecture for multi-robotics system development

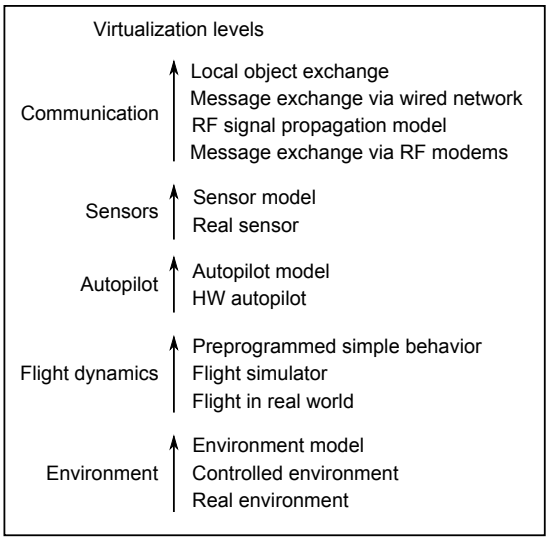

Fig. 3. Virtualization levels of the system layers

layer structure that modularizes a multi-robot system into individual subsystems. These modules can be represented by a real hardware or by a software model with various levels of virtualization. By a combination of these real and virtual modules in MR simulations, we can gradually decrease the virtualization level of the whole system by adding hardware parts and thus efficiently develop various complex multi-robot systems.

There are two main parts of the development method: (i) a multi-layer architecture of various virtualization levels; and (ii) various stages that determine the virtualization levels of the layers during development together with the process of verification of correct behavior on each level.

A. Design layers

The herein presented method uses a seven-layer architecture that is shown in Figure 2 together with schematic indication of interactions among individual subsystems. The architecture layers starting from the bottom are as follows.

1. Environment layer represents the environment in which all the assets or agents exist. It subsumes real environment if the virtualization level of at least one agent is such that it can sense or move in the real world, and it contains a simulation server that has a model of a virtual world (with the defined terrain, weather or entities obtained from third party systems). Simulation server can use the information from real assets to enhance the virtual world model with real world data.

2. Asset Layer defines the flight dynamics of the entities in the system and their physical behavior in the environment, i.e., how the environmental forces affect the state of the entity.

3. Autopilot Layer represents the behavior of the autopilots. Entities can be equipped either by hardware autopilots (in such case this layer implements the protocols to communicate with the autopilots) or by a set of algorithms and control loops that simulate the autopilot.

4. Sensory Layer specifies the payload that is deployed on the assets, such as camera, LiDAR, ADS-B, etc. Either real hardware devices or their models can be used. In the case of mixed-reality scenarios where both hardware and virtual sensors are used, the information from real sensors is passed through the simulation server to the simulated sensors and vice versa. This way, developers can increase both the scalability of the scenario and the simulation's fidelity.

5. Communication Layer is used for a message exchange and also defines a set of communication protocols, such as various interaction protocols (e.g., FIPA protocols [25]), routing protocols for Mobile Ad-hoc NETworks (MANETs) or protocols of the message transport (handshakes, etc.). Exchange of messages between entities with a different level of virtualization (i.e., between those with hardware and those with virtual communication infrastructure) is handled by the same means as other mixed-reality data exchanges – on the level of the simulation server.

6. Reasoning Layer defines algorithms for high-level control, coordination, decision making, mission planning, and execution monitoring. This is a pure software layer, and as such, it is the same for all agents independent of their virtualization level.

7. HMI Layer is the uppermost layer used by the system operators to command and control the assets and to visualize mission data.

The realization of the bottom five layers depends on the virtualization level of entities used in the testing scenario and can vary from simulated models to the utilization of hardware equipment. Possible variations of individual levels can be seen in Figure 3. More details about the individual layers and the subsystems their represent have been presented in [6].

B. Development Stages

During the development of an autonomous multi-UAV system, the very system must pass through several stages that differ in the utilized level of virtualization of their system

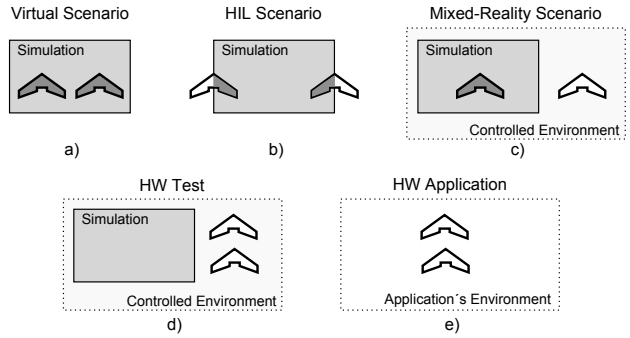

Fig. 4. Consecutive scenario stages of a development process

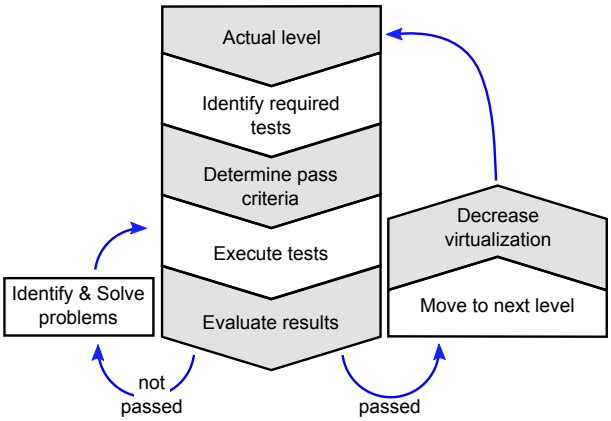

Fig. 5. Steps for verification of individual virtualization levels

layers. These development stages are represented by the following scenarios and schematically depicted in Figure 4.

Purely virtual scenarios (see Figure 4a), where the environment of agents is represented solely by the simulation server with no mixed reality feeds. The UAV assets are modeled by simple software behaviors, and they are equipped only with simulated sensors. This setting is most suitable for early stages of development where the system fidelity is not yet so important, and it is used for high-level system design and validation.

Mixed-reality scenarios, where at least one layer of the proposed architecture in one of the assets is virtual, and at least one layer consists of physical hardware. Depending on the virtualization level, the mixed reality scenarios can range from groups of agents with some simulated and some hardware layers (Hardware-in-the-loop (HIL) configuration), see Figure 4b, to groups where fully simulated agents interact with others that are fully hardware deployed[1] (Figure 4c)). In the case of an HIL configuration, the most frequently virtualized layers are the *Asset layer* and *Environment layer*, that are usually replaced by a flight simulator, since utilization of hardware assets is the most costly part, and outdoor flight tests are very time and resource consuming.

Fully deployed hardware field tests are such where all the agents are embodied by hardware assets of various types with real sensors, on-board computers and data modems under controlled conditions and operator and developers supervision (Figure 4d). This setting corresponds to the desired final stage of the UAS where all the assets operate in the real world in a distributed autonomous fashion with the minimal requirements on a human operator. Carrying out the flight test in this configuration is the most demanding in costs, time and risk taken. This type of scenarios does not scale well because of the price of the overall setting, and because of the limitations posed by communication medium bandwidth, legislation, and safety requirements.

Fully deployed hardware application (Figure 4e) is the final stage of the unmanned system development, where the system is deployed in the real application. System settings do not differ from the previous stage; however, the conditions are not controlled anymore. The system needs to be operational in all situations required by the application, such as various weather and light conditions, battery management needs to be solved (either by automatic exchange or recharge) and safety and legal issues need to be taken care of. Exhaustive field testing must precede this final system stage.

As the system has to pass through multiple virtualization levels during its development, it has to be verified on each such level before moving to the next one. Verification procedure ensures that eventual design or implementation errors are found and resolved and that the system is ready to proceed to further virtualization decrement. Individual verification steps for a given virtualization level are depicted in Figure 5.

At each virtualization level, we first need to identify required system behavior, fidelity features, and corresponding tests needed for a transition to the next level. Next, criteria for passing these tests based on the requirements should be set. If after the execution of the proposed tests and evaluation of their results the pass criteria are not met, it is necessary to identify and resolve the problems causing test failures and re-run the tests. Otherwise, if all the tests pass, it is possible to move to the next level and decrease the virtualization of the system.

IV. Case Studies

A. Heterogeneous Team of Autonomous UAVs

The method and architecture presented above have been used for development of unmanned system consisting of heterogeneous team of autonomous UAVs capable of performing complex tactical missions such as team area surveillance, target tracking or critical infrastructure protection and performing dynamic mission reconfiguration in case of change of the mission or number of available assets, the parts of which has been already described in [24], [6], [26]. Individual layers of the architecture and their inner content are shown in Figure 6. Note that the first five layers can be mixed up with both real and simulated parts to allow incremental development in MR simulation.

We have performed series of experiments with fixed wing and VTOL rotary UAVs to verify the architecture. These

[1]By fully hardware deployed entity it is meant an entity with none of its mission-critical parts virtualized that can be still equipped with virtual sensors for reception of simulation-based feeds.

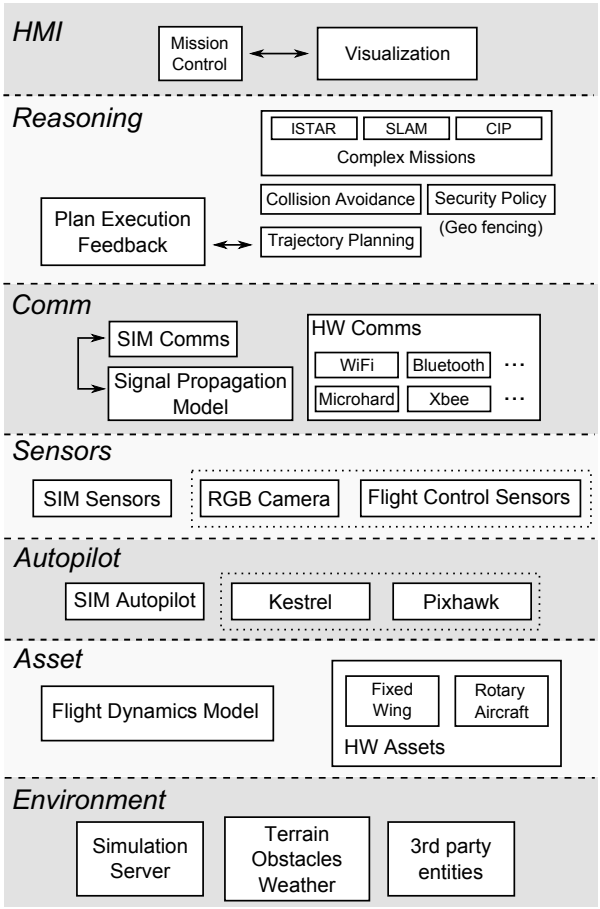

Fig. 6. Architecture of the system for command and control of autonomous UAVs

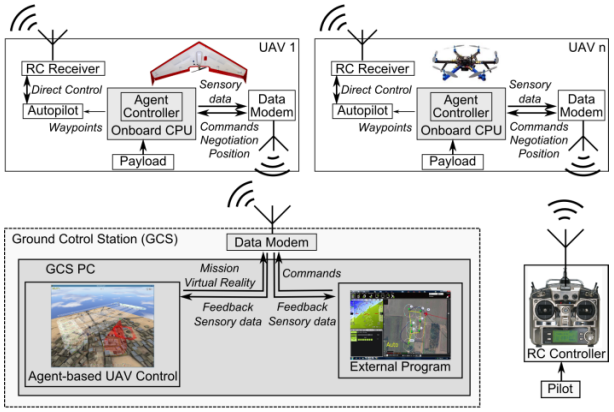

Fig. 7. Scheme of multi-UAV system deployment

Fig. 8. Flight tests of multi-UAV system

experiments started from the pure simulation of all assets and environment. Next, we run indoor hardware-in-the-loop simulations with real modems and autopilots with simulated sensory inputs and flight dynamics. We implemented interaction protocols for autopilots used (i.e., Kestrel, Pixhawk). Next, we performed field experiments in MR simulations with one hardware UAV and several virtual ones. Finally, we experimented with multiple fully deployed hardware assets, following the incremental development of HART application as specified in [1]. This incremental implementation of HW subsystems allowed less flight test hours for one and mainly two aircraft flying simultaneously. The proposed methodology has been found advantageous because of deployment cost because the tests were executed once they were necessary for the verification of the particular virtualization levels.

The system was deployed on real hardware UAVs according to the scheme shown in Figure 7. It is divided into two parts: (i) on-board agent control and reasoning; and (ii) ground control station (GCS) with mission control, simulation, and visualization. The first part runs on an on-board computer that is connected to all sensory payload, directly to an autopilot to allow waypoint upload, and to the data modem for team coordination and communication with the operator.

The GCS part runs all optional virtual UAVs together with the simulated environment if necessary, and is used for the mission assignment and visualization. Third party GCS software (Mission Planner, Virtual Cockpit, UGCS, etc.) can be connected for safety reasons and extended visualization options. For further improvement of the operation safety and for legal reasons the system allows to overtake control of each UAV by a pilot with the Remote Controller (RC).

The most significant difference among the experimental stages and one of the tests required for transfer between individual development stages were communication bandwidth and delay. The bandwidth has to be wide enough to transmit plans, commands, telemetry, and other data required for a proper system behavior and transmission delay had to be small enough to provide good response time and well-timed collision avoidance. The minimal bandwidth and maximal delays were estimated in simulated scenarios with a variable number of UAVs and tasks performed. The communication delay manifested itself as: (i) the delay between operator's command and the beginning of the mission execution; and (ii) the amount of time before possible collision when the collision avoidance planners start to re-plan the trajectories. Note that each UAV should be informed about any possible collision 50 seconds ahead because of periodic plan broadcast. Variable delays were caused by different data-link throughputs in every development stage. Together with

TABLE I

DATA-LINK QUALITY IN DIFFERENT DEVELOPMENT STAGES.

System	Data-link spec	Throughput [Mbps]	Mission execution delay [ms]	Collision Avoidance delay [s]
Sim	localhost TCP/IP sockets	650.0	10	50
HIL	RF modems in lab	30	300.0	48
HW	RF modems in the field	0.5	1500	40

the requirements on bandwidth, and available budget, these characteristics were used to select suitable communication hardware. Since cheap XBee modules shown unsatisfactory for our purposes because of their range and available bandwidth, Microhard nVIP2400 RF modems were used, and their effects on throughput and delays are shown in Table I.

B. System for Verification of Collision Avoidance Methods

The presented development method has also been deployed in a project with U.S. Airforce Research Lab (AFRL), Quanterion Solutions Inc., and NUAIR alliance, where it was used to enable high-level command and control of multiple fixed wing aircraft Desert Hawk III. The emphasis here was placed on training the system operators, flight safety in shared airspace and verification of cooperative collision avoidance among aircraft where the testing was crucial for achieving desired reliability.

This system is similar to the first one, and also the individual layers of the architecture and their inner content are similar to those in Figure 6. Increased safety requirements lead to usage of extra RC as well as extra GCS with proprietary third party software (Virtual Cockpit) in the HMI layer for every single UAV and a common GCS for multi-agent command and control. Also, a new autopilot protocol (AFRL's Common Communications Client for Payload Operations (C3PO)) had to be implemented and tested, see Figure 9.

It should be noted, that the system was not deployed and operated by us, but by members of Quanterion Solutions with our remote assistance only. This refers to simplicity and portability of the resulting system. Figure 10 shows Desert Hawk III launch and field test setting.

During this project, there was an extensive hardware-in-the-loop testing phase, not only for the development reasons but also for system operators to get familiar with it safely. These operators provided feedback that resulted in changes in the HMI layer and integration of third party software for monitoring and visualization (Virtual Cockpit[2] software). Also, since communication based cooperative collision avoidance techniques were to be tested, verification of

[2]http://www.lockheedmartin.com/us/products/procerus/kestrel-autopilot.html

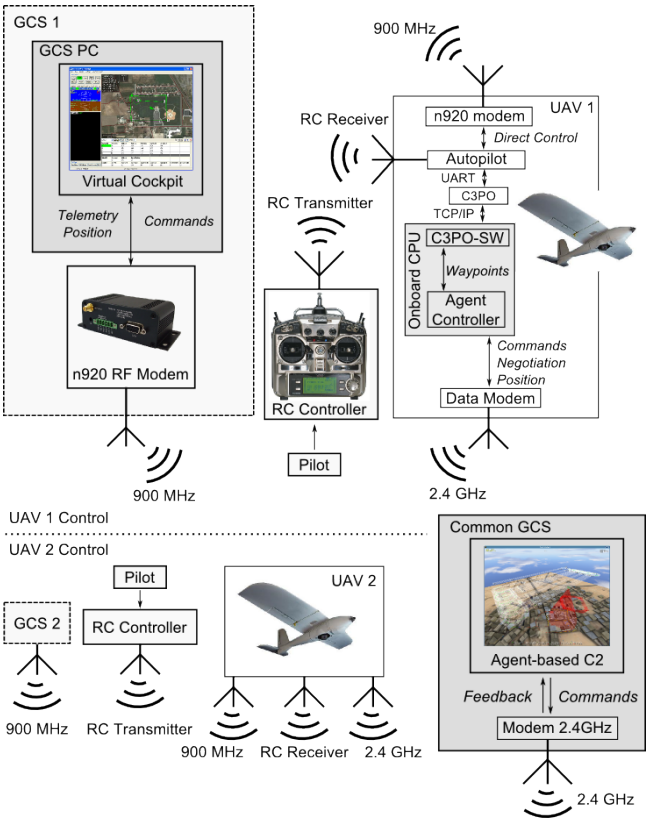

Fig. 9. Scheme of system deployment on Desert Hawks

Fig. 10. Flight tests with Desert Hawks III

communication devices, particularly the sufficiency of their range and reliability was performed.

C. Safefly - System for Flight Safety of Light-Sport Aircraft

The last herein reported use case where the architecture for incremental development has been used is the Safefly project that aimed to build a system for increasing flight safety of light-sport aircraft. A combination of negotiation among aircraft, cooperative, and non-cooperative methods of collision avoidance and trajectory planning algorithms were employed to prepare and recommend the best collision-free trajectories for pilot or eventually upload them to an autopilot of light-sport aircraft if possible.

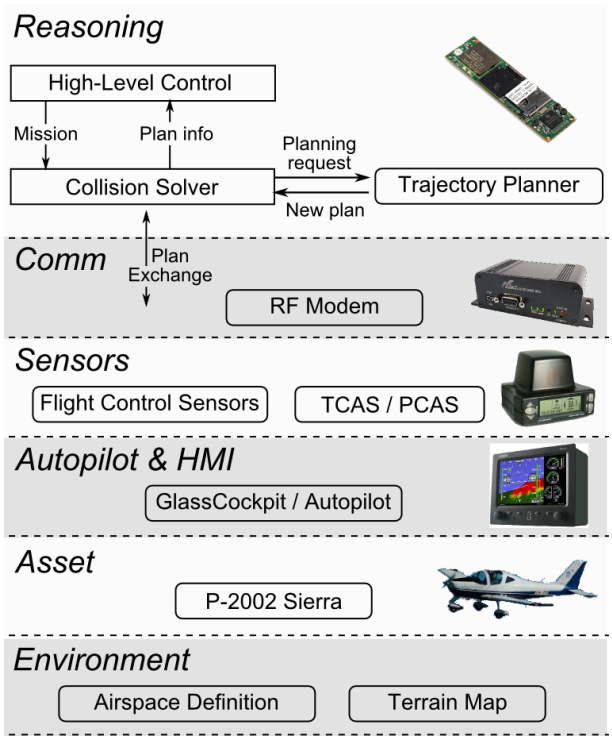

Fig. 11. Architecture of the system for light sport aircraft control with hardware components used for its realization

The system consists of multiple hardware parts that were selected to satisfy needs of the final system on computational power, communication bandwidth, size, and weight.

- ***TL-1000 Integra*** - an electronic flight instrument system (EFIS) that integrates all primary flight instruments, navigation, and 3-D terrain, together with glass cockpit for various visualization options and autopilot options for pitch and roll control. Integra systems were integrated into two light-sport planes Tecnam P-2002 Sierra and connected with the other system parts.
- ***Zaon PCAS XRX*** - a passive collision avoidance system that uses amplitude and phase shift of other aircraft's transponder responses to estimate their presence and position. Since a great majority of civil aircraft is equipped with a transponder, this system provides non-cooperative detection of surrounding aircraft.
- ***Microhard n920F*** - an RF module for cooperative plan exchange and negotiation of evasion maneuvers among aircraft equipped with the Safefly system. It is also used to transmit mixed reality data and information to/from GCS.
- ***Gumstix Overo FireStorm*** - an onboard ARM computational unit that is connected to all the other system parts and runs the reasoning algorithms for planning, collision avoidance, and negotiation.

The individual layers of the architecture and their inner content are shown in Figure 11 where the simulated parts of the layers are omitted because they are similar to those in Figure 6.

Fig. 12. Model of P-2002 Sierra utilizing BADA performance characteristics used for MR simulations

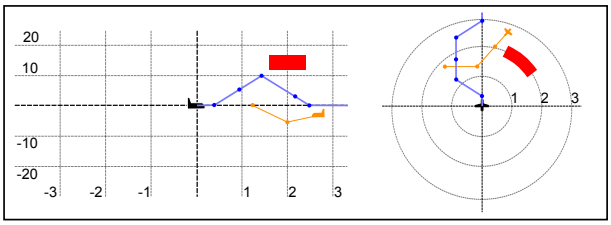

Fig. 13. Plan visualization in Integra Glass Cockpit used for situation awareness (in red are depicted Zaon observations and orange is a trajectory of near cooperating aircraft) and flight path recommendations (blue line)

Accordingly to the concept of the incremental development, we firstly prepared a flight dynamics model of Tecnam aircraft based on their BADA performance characteristics [27] and tested planning and collision avoidance maneuvers in pure simulation, see Figure 12. Later, a model of Zaon PCAS, based on its theoretical characteristics stated by the manufacturer was integrated into the system to provide for non-cooperative collision avoidance. After that, we obtained a simulation of the Integra EFIS from its manufacturer and prepared interaction protocols to upload plan waypoints to glass cockpit / autopilot.

In the next stage, we tested individual physical modules for reliability, bandwidth and range of RF modems, real behavior of the Zaon PCAS, and interaction with the pilots in various situations to verify the suitability of the modules and to increase the precision of the respective software models. After a discussion with pilots, we have designed a plan visualization widget (Figure 13) to be shown in the glass cockpit together with other flight data that would help them in following the provided best collision-free path.

Next, we deployed the Safefly system on one of the experimental Tecnam planes (see Figure 14), fed it with data of a virtual plane in MR simulation and observed pilot reactions on plan recommendations. After integration of pilots' feedback into the system and solving the raised hardware issues, we finally fully deployed the system on two experimental planes.

V. Lessons Learned

The incremental development strategy with the use of MR simulation shown itself useful in all the presented projects. MR simulation provided valuable insights into the systems that would be otherwise difficult to obtain in regular real

Fig. 14. Aircraft pilot panel with integrated Safefly system – build-in EFIS Integra, Zaon PCAS with GPS on the flight panel and rest of the system integrated under the panel

world tests. The most valuable lessons learned are summarized in the following paragraphs.

A. Re-use of system modules

The modular architecture enables re-use of many system modules in different projects even if they are of different type (e.g., communication protocols, planning and collision avoidance methods, etc.) and can save time and effort when searching for hardware requirements since they were already estimated before. For example, the communication device and onboard computers in the Safefly system were selected based on our measurements of system requirements from previous projects.

B. Interchangeable modules

Using MR simulation with incremental decreasing of virtualization level makes replacement of individual system parts a straightforward task. This was the case in the Safefly project because the manufacturer of Zaon XRX PCAS device stopped its production and support and some of its alternatives have to be used (e.g., PowerFLARM). Integration to the system requires just implementation of the device interface and its step by step verification in particular development stages.

C. Incremental training of system operators

MR simulations can significantly help final system operators in getting familiar with the system without legal issues and risk of damage to hardware or third party property. They can utilize incremental decreasing of virtualization level to get familiar with the system step by step. This is especially useful for aerial systems where the risks and legal issues are of the high importance, as was found in the AFRL project in which the operators requested extensive hardware-in-the-loop testing.

D. Localization of problems

Incremental development helps in the localization of problems that would not appear in purely SW simulation, and that would be difficult to isolate in the fully deployed complex system, e.g., autopilot behavior during path following, or effect of delays in the communication.

E. Bridging development delays

In a case of a cooperation of multiple parties on a project, such as the Safefly project, the MR simulation shown itself useful for substituting delayed hardware parts. During the project, there were significant periods when we were waiting for software or hardware delivery from our project partners. In these cases, virtualization and modeling of individual system layers shown to be beneficial since they were used as a substitute for the hardware without the need to delay further system development.

F. Problematic wireless communication modeling

MR simulations are difficult to be used for study wireless communication related issues. The problem here is that even if the virtual entities use hardware RF modems to communicate with the physical entities, and thus contribute to the common bandwidth use, the fact that locations of the modems do not correspond to the assumed locations of the virtual entities cause incorrect assumptions on the RF signal propagation. We would like to address this issue in our future work.

VI. Conclusion

In this paper, a methodology for incremental development and verification of complex multi-robot systems used in a number of UAS projects is presented. Universality of this method is demonstrated by its utilization for building three different aviation applications: (i) a system for command and control of heterogeneous team of autonomous unmanned aircraft; (ii) a system for verification of collision avoidance methods of third party's unmanned aircraft; and (iii) an assistant system for pilots of light-sport aircraft that should increase their situation awareness and safety through recommendations of collision-free flight paths.

Acknowledgment

The presented work has been supported by the Czech Science Foundation (GAČR) under research project No. 16-24206S, Ministry of Agriculture of the Czech Republic under contract No. QJ1520187, and by the Technology Agency of the Czech Republic under project No. TA01030847.

References

[1] M. Jakob, M. Pěchouček, M. Čáp, P. Novák, and O. Vaněk, "Mixed-reality testbeds for incremental development of HART applications," *IEEE Intelligent Systems*, vol. 27, no. 2, pp. 19–25, 2012.

[2] R. Garcia and L. Barnes, "Multi-UAV simulator utilizing x-plane," in *Selected papers from the 2nd International Symposium on UAVs, Reno, Nevada, USA June 8-10*, 2009, pp. 393–406.

[3] A. Komenda, J. Vokřínek, M. Čáp, and M. Pěchouček, "Developing multiagent algorithms for tactical missions using simulation," *IEEE Intelligent Systems*, vol. 28, no. 1, pp. 42–49, 2013.

[4] W. Honig, C. Milanes, L. Scaria, T. Phan, M. Bolas, and N. Ayanian, "Mixed reality for robotics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 5382–5387.

[5] I. Chen, B. MacDonald, and B. Wunsche, "Mixed reality simulation for mobile robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 232–237.

[6] M. Selecký, M. Rollo, P. Losiewicz, J. Reade, and N. Maida, "Framework for incremental development of complex unmanned aircraft systems," in *Integrated Communication, Navigation, and Surveillance Conference (ICNS)*. IEEE, 2015, pp. J3–1.

[7] F. Mutter, S. Gareis, B. Schatz, A. Bayha, F. Gruneis, M. Kanis, and D. Koss, "Model-driven in-the-loop validation: Simulation-based testing of UAV software using virtual environments," in *18th IEEE International Conference and Workshops on Engineering of Computer Based Systems (ECBS)*, 2011, pp. 269–275.

[8] S. Demers, P. Gopalakrishnan, and L. Kant, "A generic solution to software-in-the-loop," in *Military Communications Conference (MILCOM)*. IEEE, 2007, pp. 1–6.

[9] A. Goktogan and S. Sukkarieh, "Distributed simulation and middleware for networked UAS," in *Unmanned Aircraft Systems*, 2008, pp. 331–357.

[10] M. Day, M. Clement, J. Russo, D. Davis, and T. Chung, "Multi-uav software systems and simulation architecture," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2015, pp. 426–435.

[11] I. Pizetta, A. Brandao, and M. Sarcinelli-Filho, "A hardware-in-the-loop platform for rotary-wing unmanned aerial vehicles," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 725, pp. 725–743, 2016.

[12] A. Burkle, F. Segor, and M. Kollman, "Towards autonomous micro UAV swarms," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1, pp. 339–353, 2011.

[13] P. Scerri, T. Von Gonten, G. Fudge, S. Owens, and K. Sycara, "Transitioning multiagent technology to UAV applications," in *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS): Industrial track*, 2008, pp. 89–96.

[14] J. Sanchez-Lopez, J. Pestana, P. de la Puente, and P. Campoy, "A reliable open-source system architecture for the fast designing and prototyping of autonomous multi-uav systems: Simulation and experimentation," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1-4, pp. 779–797, 2016.

[15] P. Chudy, P. Dittrich, and P. Rzucidlo, "HIL simulation of a light aircraft flight control system," *IEEE/AIAA 31st Digital Avionics Systems Conference (DASC)*, pp. 6D1–1, 2012.

[16] M. Aydemir, "Design and implementation of a compact avionics instrument for light aviation," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 24, no. 5, 2016.

[17] P. Pačes, T. Levora, O. Bruna, J. Popelka, and J. Mlejnek, "Integrated modular avionics onboard of small airplanes: Fiction or reality?" in *IEEE/AIAA 30th Digital Avionics Systems Conference (DASC)*, 2011, pp. 7A1–1.

[18] K. Rydlo, P. Rzucidlo, and P. Chudy, "Simulation and prototyping of FCS for sport aircraft," *Aircraft Engineering and Aerospace Technology*, vol. 85, no. 6, pp. 475–486, 2013.

[19] T. Haberkorn, I. Koglbauer, R. Braunstingl, and B. Prehofer, "Requirements for future collision avoidance systems in visual flight: a human-centered approach," *IEEE Transactions on Human-Machine Systems*, vol. 43, no. 6, pp. 583–594, 2013.

[20] J. Pellebergs and Aeronautics, "The MIDCAS project," *Saab Aeronautics*, 2012.

[21] C. Munoz, A. Narkawicz, G. Hagen, J. Upchurch, A. Dutle, M. Consiglio, and J. Chamberlain, "DAIDALUS: detect and avoid alerting logic for unmanned systems," in *IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, 2015, pp. 5A1–1.

[22] I. Chen, B. MacDonald, and B. Wunsche, "Evaluating the effectiveness of mixed reality simulations for developing uav systems," in *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, 2012, pp. 388–399.

[23] M. Selecký and T. Meiser, "Integration of autonomous UAVs into multi-agent simulation," *Acta Polytechnica*, vol. 52, no. 5, pp. 93–99, 2012.

[24] M. Selecký, M. Štolba, T. Meiser, M. Čáp, A. Komenda, M. Rollo, J. Vokřínek, and M. Pěchouček, "Deployment of multi-agent algorithms for tactical operations on UAV hardware," in *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2013, pp. 1407–1408.

[25] L. Chiariglione. (2001) FIPA: Foundation for intelligent physical agents. [cited 5 May 2017]. [Online]. Available: http://www.fipa.org

[26] M. Selecký and M. Rollo, "Distributed control of heterogeneous team of autonomous uavs," in *Proceedings of EXPONENTIAL 2016: Association for Unmanned Vehicle Systems (AUVSI)*, 2016, pp. 707–717.

[27] A. Nuic, D. Poles, and V. Mouillet, "Bada: An advanced aircraft performance model for present and future atm systems," *International Journal of Adaptive Control and Signal Processing*, vol. 24, no. 10, pp. 850–866, 2010.