Noname manuscript No.
(will be inserted by the editor)

Analysis of Using Mixed Reality Simulations for Incremental Development of Multi-UAV Systems

Martin Selecký · Jan Faigl · Milan Rollo

Received: date / Accepted: date

Abstract Developing complex robotic systems requires expensive and time-consuming verification and testing which, especially in a case of multi-robot unmanned aerial systems (UASs), aggregates risk of hardware failures and may pose legal issues in experiments where operating more than one unmanned aircraft simultaneously is required. Thus, it is highly favorable to find and resolve most of the eventual design flaws and system bugs in a simulation, where their impacts are significantly lower. On the other hand, as the system development process approaches the final stages, the fidelity of the simulation needs to rise. However, since some phenomena that can significantly influence the system behavior are difficult to be modeled precisely, a partial embodiment of the simulation in the physical world is necessary. In this paper, we present a method for incremental development of complex unmanned aerial systems with the help of mixed reality simulations. The presented methodology is accompanied with a cost analysis to further show its benefits. The generality and versatility of the method is demonstrated in three practical use cases of various aviation systems development: (i) an unmanned system consisting of heterogeneous team of autonomous unmanned aircraft; (ii) a system for verification of collision avoidance methods among fixed wing unmanned aerial vehicles; and (iii) a system for planning collision-free paths for light-sport aircraft.

Keywords UAS development · Mixed-reality simulations · UAS applications · multi-UAV systems

1 Introduction

With increasing numbers of Unmanned Aerial Vehicles (UAVs) simultaneously carrying out their missions increases the cognitive load on the UAV operators. To deal with this problem, the autonomy of UAV assets needs to be improved and the research focus is being shifted from development of solitary intelligent vehicles to teams of

Authors are with the Czech Technical University, Faculty of Electrical Engineering, Technicka 2, 166 27, Prague, Czech Republic
{selecmar|faiglj|rollom}@fel.cvut.cz
Tel.: +420-22435-7677

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

cooperating intelligent UAVs. Developing and verifying algorithms to control teams of UAVs is a complicated task with a lot of complex issues that need to be taken into account [1]. Some of them, like complex interactions among the team members or basic functionality of the planning algorithms can be modeled with the help of multi-agent software simulations [2], [3]. However, pure software simulations are not enough to deploy the algorithms in the real-world. Effects of real-world phenomena like those of the communication issues, sensor and actuator errors, weather or limited computational resources on the function and efficiency of the algorithms have to be studied. Since in many cases these phenomena are difficult to be modeled on high fidelity levels, the whole system needs to be verified using real hardware assets, which can be very costly if it is made for all components at every stage of the system development and integration.

Recently, experiments have been conducted both in a virtual world and in the real world as each of them has its limits and justifications [4], e.g., specifically for the real experiments that are costly and has to respect physical constraints such as available space. Virtual simulation allows for quick observations of results of interactions among simulated entities and the environment fast modeling of a basic functionality of the developed system. Its main advantage is an ease of set up and execution and repeatability. Simulations can be used to study effects of various environmental phenomena by modifying only specific conditions and fixing others, and thus removing possible sources of noise and ambiguity. Moreover, experiments conducted in virtual environments can significantly save costs both during regular testing and also in cases of malfunctions and accidents.

On the other hand, software simulations cannot incorporate all sources of the real world inputs that are needed for realistic modeling of the behavior of the developed system. Thus, 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 robustness before it is deployed in real missions.

Mixed Reality (MR) simulations [5] provide a world where the virtual and real objects and entities can co-exist and interact in real time. The MR concept allows 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 parts of the system being real and other parts 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 and are observed instead. In contrast to hardware-in-the-loop (HIL) simulations [6,7], the concept of MR simulations allow for easily executable co-existence and interaction of purely virtualized and fully hardware deployed entities.

1.1 Overview and Contribution

This paper provides analysis of the method for incremental development of HART (human, agent, and robot teams) applications originally introduced in [1], and reports on practical usage of modular multilayer architecture for incremental development of complex unmanned aerial systems (UASs) [8]. Both the method and the architecture have been successfully employed during development of different aviation applications within projects funded by Czech Ministry of Defence, U.S. Air Force Research Laboratory, and U.S. Army CERDEC, and also utilized for the development of an assistant system for light-sport aircraft pilots. This multitude of applications shows the versatility of the presented method for UASs in research and development. An overview of the real aerial vehicles for which the method has been utilized is shown in Figure 1 and three use cases that are reported in this paper are as follows.

The first presented use-case application is an unmanned system that provides command and control capabilities over a heterogeneous team of autonomous unmanned aircrafts [9,8]. The aircrafts are capable of performing complex tactical missions such as critical infrastructure protection, team area surveillance or target tracking, and they can perform distributed dynamic mission reconfiguration in a case of change of the mission or the number of available assets [10]. The second use-case application is a system for verification of communication-based cooperative collision avoidance methods among multiple fixed wing UAVs. In this case, the objective has been to develop a complete testing environment by incrementally deploying the system on two Lockheed Martin's Desert Hawk III aircraft and training the system operators. Finally, the third use-case application is a system for providing increased flight safety in a domain of light-sport aircraft where a combination of existing sensors, negotiation among aircraft, cooperative, and non-cooperative methods of collision avoidance and trajectory planning algorithms has been used to recommend the best collision-free trajectories for pilot or autopilot of the real light-sport aircraft.

A short version of this paper has been presented at the ICUAS 2017 [11]. The herein presented results originate on the previously published achievements and have been substantially updated in terms of the impact to further developments and deployments of complex UASs. In particular, we present an analysis of the risks and costs that may occur during the incremental development process. Further, a detailed description of the development process in the presented use-cases together with the important measurements are presented to support deeper insights into the benefits of the proposed methodology and architecture.

The remainder of this paper is structured as follows. After relating the presented approaches to the respective research fields in Section 2, the proposed multi-layer architecture used for the incremental development of UAS together with risk and cost analysis of individual stages of the development process are described in Section 3. Next, the three aviation use-case applications are presented in Section 4 showing how they utilize the proposed multi-layer architecture and methodology, followed by a summary of the experience and lessons learned from the system deployment in Section 5. Finally, Section 6 gives a conclusion of the achieved results.

2 Related Work

Software simulations are commonly used to develop complex systems because they can significantly speed up the development process and save costs on hardware as they allow to early discover, isolate, and correct potential implementation issues. According to Mutter et al. [12] there are two approaches to a simulator development: (i) the software-in-the-loop (SIL), where all system components (sensors, actuators, airframes, communication, etc.) are simulated using mathematical models; and (ii) the hardware-in-the-loop (HIL), where only some (or none) system parts are simulated, and other elements are the particular physical hardware. Examples of simulators using SIL approach can be found in [13] and [2]. The HIL approach is taken, e.g., by Goktogan et al. [14], Day et al. [6], and Pizetta et al. [7] to name few.

This paper describes utilization of the HIL approach extended to integrate benefits of MR simulations for incremental development of complex multi-UAV systems and an assistant system for pilots of light-sport aircraft. Therefore, we report on the related work in each of these research areas.

2.1 Deployment of Algorithms on Multi-UAV Systems

Various approaches have been proposed to develop and deploy complex multi-UAV systems. For example Šišlák et al. [15] developed AgentFly system for simulating UAVs and civilian air traffic. It allows complex coordination and cooperation of the agents, and provides collision avoidance mechanisms. However, it could not control hardware assets. Bürkle et al. [16] presented a deployment of control mechanisms for teams of hardware VTOL micro-UAVs capable of on-board planning and cooperation on mutual tasks, but they do not coordinate their flight in terms of collision avoidance. The system also does not allow co-existence of hardware UAVs together with simulated ones. Scerri et al. [17] designed a system for deployment of multi-agent technology for UAV coordination to an industrially developed applications, but they used a straightforward approach that switches from a purely virtual configuration directly to the full hardware configuration, which is not suitable for more complex systems. Most recently, Sanchez-Lopez et al. [18] present a multipurpose system architecture for autonomous multi-UAV platforms with basic high-level planning features which, however, does not allow for complex distributed multi-UAV planning.

2.2 Advisory Systems for Light-Sports Aircraft

As for the control and advisory systems for light-sport aircraft operation, there are already available off-the-shelve autopilots that can hold the flight level and change the aircraft heading to the next waypoint according to the given plan. These autopilot systems are still subject to research and improvement [19]. The actual trend in this field is equipping the light-sport aircraft with the Electronic Flight Instrument System (EFIS) with a "glass cockpit" that integrates all avionic sensors into one digital instrument [20] that allows for adding further functions [21,22]. In the field of surveillance and collision avoidance systems, Haberkorn et al. [23] studied options and efficiency of methods for cooperative and non-cooperative detection (TCAS – Traffic Collision Avoidance System, PCAS – Portable Collision Avoidance System) of other aircrafts for a collision prevention and there have been projects dealing with increasing pilot's awareness of the surrounding air traffic [24,25].

2.3 Mixed Reality Simulations

One of the first notions of MR simulations used for the development of mobile robots has been presented by Chen et al. [5]. The authors used an MR simulation tool based on the 3D robot simulator Gazebo in a single robot scenario and demonstrated advantages of combining real entity and simulated objects for development of a practical robotic system. Later, in [26], the same authors showed that MR simulations can help to provide efficient testing tools, 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. Recently, Honig et al. [4] used MR simulations together with robotic simulators and showed that these simulations can provide many advantages for research and development in robotics. The authors supported their findings by several use-cases with Crazyflie nano-quadcopters.

One of the most recent approaches to formal dealing with the process of the development of complex robotic systems has been proposed by Jakob et al. in [1]. The authors suggested a process of incremental development of complex systems by utilization of mixed-reality test-beds. They defined *levels of virtualization* as a degree to which parts of the target system setup are replaced with synthetic computational models in a given test-bed configuration. The authors suggested to start the system development with a fully virtual system with all parts purely virtualized and get to the final configuration with zero virtualization level by iterating through the space of test-bed configurations in directions which are given by the development costs of all the necessary iteration steps. Figure 2 shows the test-bed configuration space for n UAV agents. However, the work is aimed rather at explaining the plain concept of utilization of mixed-reality simulations for system development, and it does not describe its practical application, architecture specifics nor the process of estimating the iteration step costs.

The architecture presented here builds on the ideas of [1] that are further generalized for the use of the incremental implementation of hardware subsystems in MR simulation system. Based on the practical development and deployment of the real UAV applications, the found insights to the development process and its risk and cost analysis are provided in this paper. The proposed architecture allows for incremental development of systems of multiple autonomous robotic entities and thanks to its gen-

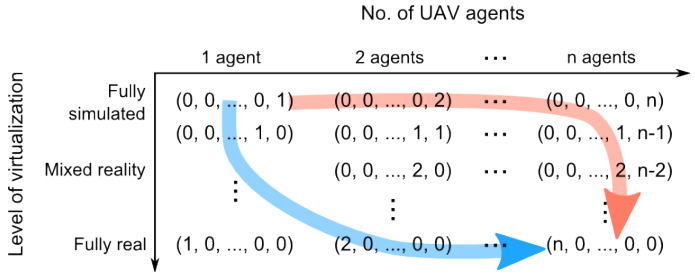


Fig. 2: Space of test-bed configurations and example of two strategies of iterative development

erality, it provides a universal method applicable for the development of systems of various types, as it is shown further in the rest of the paper.

3 Architecture for Incremental System Development

The herein presented architecture leverages on the work [8] and it has been already successfully used for development of multi-UAV systems of cooperating fixed-wing aircraft [27] and [9]. Within the architecture, individual subsystems of a multi-robot system are modularized and situated inside a seven-layer structure. Hereby defined modules can be represented by a software model with various levels of fidelity or by a physical hardware modules. By a combination of these virtual and physical modules inside MR simulations, the virtualization level of the whole system can be gradually decreased by the iterative addition of hardware parts. This process then leads to the efficient development of various complex multi-robot systems.

There are three main parts of the proposed iterative development method: (i) a modularized multi-layer architecture that enables simulations at various virtualization levels; (ii) a process of verification of the correct behavior on each of these levels; and (iii) an estimation method of the development cost of each iteration step.

3.1 Design Layers

The seven-layer architecture for the classification of system sub-modules is shown in Figure 3 together with the schematic indication of possible interactions among the 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 a real environment if the virtualization level of at least one agent is such that it can sense or move in the real world. It also contains a simulation server with a model of the virtual world (with the defined terrain, weather or entities obtained from third-party systems). The simulation server can use the information from the real assets to enhance the virtual world model with the 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. In a case of the zero virtualization level, this layer is represented by the physical airframe itself.

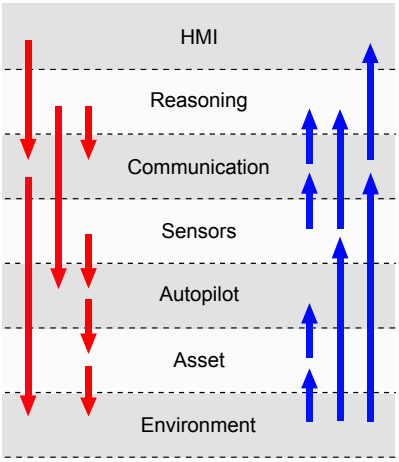


Fig. 3: Modular architecture for multi-robotics system development

3. Autopilot layer specifies the behavior of the autopilots. Entities can be equipped either by hardware autopilots (in such a case this layer implements the interfacing protocols to communicate with the autopilot) or by a set of algorithms and control loops (stabilization, trajectory following, navigation, etc.) that simulate the autopilot behavior.

4. Sensory layer specifies the payload that is deployed on the assets, such as LiDAR, camera, ADS-B, etc. Again, either real hardware devices or their computational models can be used. In the case of mixed-reality scenarios with a combination of the hardware and virtual sensors, the information from the real sensors is sent first to the simulation server and from there passed to the virtual entities that should be able to sense this type of information. A similar process applies to virtual sensors and physical entities. This way, developers can increase both the scalability of the scenario and the simulation fidelity by allowing both more entities and real sensory feeds into the simulation.

5. Communication layer is used for implementation of message exchange and also defines a set of communication protocols, such as routing protocols for Mobile Ad-hoc Networks (MANETs), various interaction protocols (e.g., FIPA protocols [28]), or protocols for message transport (handshakes, acknowledgments, etc.). Exchange of messages among entities with a different level of virtualization (i.e., between those with hardware and those with a virtual communication infrastructure) is handled by the same means as the mixed-reality sensory data exchanges – with the help of the simulation server.

6. Reasoning layer defines algorithms for high-level control, decision making, mission planning, team coordination, and plan 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 as a command and control tool to assign tasks to the assets and to visualize mission data.

The realization of the bottom five layers can vary from simulated computational models to the utilization of the hardware equipment. It depends on the virtualiza-

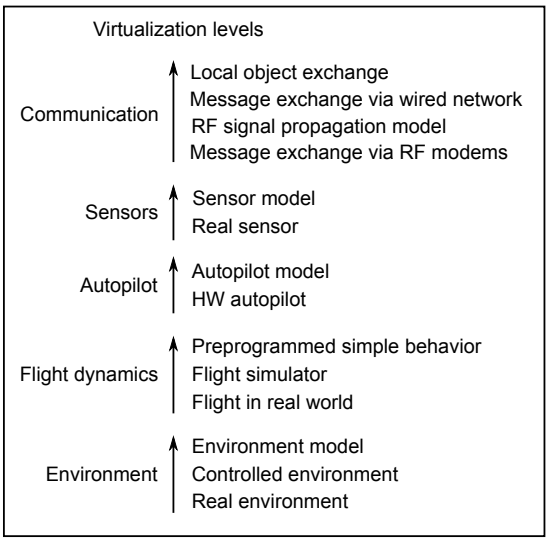


Fig. 4: Virtualization levels of the system layers

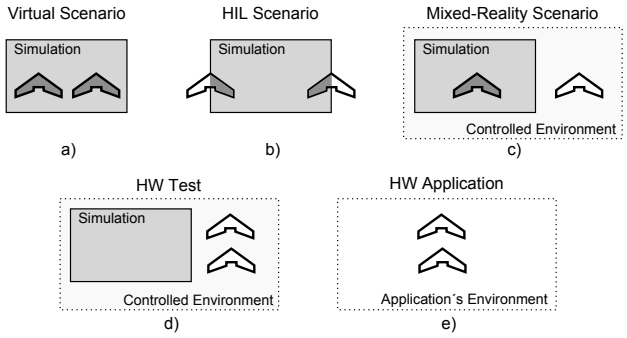


Fig. 5: Consecutive scenario stages of the development process

tion level of entities used in the testing scenario configuration. Possible variants of individual-level realizations are depicted in Figure 4 and more details about the individual layers and the subsystems can be found in [8].

3.2 Verification Process

During the incremental development of an autonomous multi-UAV system, the very system must pass through several stages that differ in the utilized level of virtualization of its system layers. Basic categorization of these development stages is presented in our previous work [11]. From the simplest to the most advanced stages are depicted in Figure 5 and they can be characterized as follows.

Purely virtual scenarios – the environment is represented solely by the simulation server, and the UAV assets are equipped only with simulated sensors and their

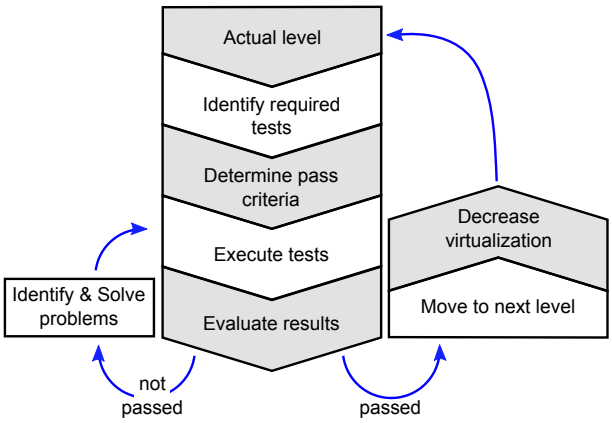


Fig. 6: Steps for verification of individual virtualization levels

movement is modeled by simple computational models. This configuration is suitable for early stages of development where the system fidelity is not yet important, and it is used for high-level system design.

Mixed-reality scenarios – at least one of the assets utilizes a computational model for at least one layer of its architecture, and at least one asset uses some sort of physical hardware. Mixed reality scenarios can range from groups of agents with mixed virtual and hardware layers (Figure 5b), to groups of interacting purely virtual and full hardware deployed[1] agents (Figure 5c)). Usually, the most frequently virtualized layers are the *Asset layer* and *Environment layer* that are replaced by a flight simulator because the outdoor flight tests are very time and resource consuming.

Fully deployed hardware field tests – correspond to the desired final stage of the UAS where all the assets operate in the real world with minimal requirements on a human operator. The only difference is that the tests are performed under controlled conditions and operator and developers supervision. Hardware assets of various types with real sensors are used, equipped with on-board computers and data modems. Scalability of these scenarios is limited by the price of the overall setting, communication medium bandwidth, safety requirements, and legislation.

Fully deployed hardware application – the final stage of the unmanned system development, where the system is deployed in the real application and the conditions are not controlled anymore. The system must remain operational in all situations required by the application, such as various light and weather conditions, safety and legal issues need to be taken care of, and battery management needs to be solved.

As the system moves through various virtualization levels during the development, it has to be verified on each such level before moving to the next level. The verification procedure presented further in this paper ensures that eventual design or implementation errors are discovered and resolved as soon as possible. Thus, the system is then ready to proceed to further virtualization decrement. Respective steps for the verification of a given virtualization level are depicted in Figure 6.

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

At each level, it is necessary to determine the required behavior of the system first and requested fidelity features, but also identify the corresponding tests needed for a transition to the next level. Then, the criteria for successful passing of the tests are set, and the test should be executed either in software simulation or field experiments. If the pass criteria are not met, it is necessary to identify and resolve the problems that caused the test failures and re-run the tests. Once all the tests pass, it is possible to move to the next development level by decreasing the virtualization of the system.

3.3 Step Cost

During the incremental development of the system, the most convenient way to lower the system virtualization level and define a new testbed configuration has to be chosen at each development step. This selection should be based on the *iteration cost* for a given pair of the testbed configurations. As defined in [1], the iteration cost of a step $cost(\sigma_1, \sigma_2)$ that transforms the system from an application that works correctly on the testbed configuration σ_1 to an application that works correctly on the configuration σ_2 consists of three partial costs:

- the *testbed cost* of providing a testbed with the configuration σ_2;
- the *development cost* of modifying the application logic of the testbed configuration σ_1 to work correctly on the testbed σ_2;
- the *test cost* of verifying that the modified application works correctly on the testbed σ_2.

Authors of [1] do not specify how to estimate these costs. Based on years of experience of building UASs and many experiments with different testbed configurations, a more detailed specification of the costs of an iteration step is provided here together with the analysis of the risks and costs during the development process.

The first two partial costs can be considered as self explanatory – *testbed cost* ($C_E(\sigma_1, \sigma_2)$) is a cost of the hardware equipment needed for moving from the testbed configuration σ_1 to σ_2 and *development cost* ($C_D(\sigma_1, \sigma_2)$) is a price of the time and resources needed to modify the application logic to work in the testbed configuration σ_2. The estimation of the *test cost* ($C_T(\sigma_2)$) is a bit more complicated since it should reflect two issues: (i) the cost of the experiments ($C_{ex}(\sigma_2)$) that consists of the preparation costs, cost of work-time of each person presented at the experiment, and cost of processing the results, and (ii) costs caused by accidental failures ($C_{fail}(\sigma_2)$) that consists of the cost of repairs and cost of repair time when no other experiments can be conducted with the broken equipment. The resulting iteration step cost can be expressed as

$$cost(\sigma_1, \sigma_2) = C_E(\sigma_1, \sigma_2) + C_D(\sigma_1, \sigma_2) + C_{ex}(\sigma_2) + r(t, \sigma_2)N(\sigma_1, \sigma_2)C_{fail}(\sigma_2), \quad (1)$$

where $r(t, \sigma_2)$ is the time-dependent failure rate of the testbed with the configuration σ_2 and $N(\sigma_1, \sigma_2)$ is the number of experiment required for verification of all new features of the configuration σ_2.

Tools of the reliability theory can be used to estimate the probability of a failure or en error in design and implementation of a complex system. According to the reliability theory, the *failure rate* (frequency with which an engineered system or component fails, expressed in failures per unit of time) in the initial stages of a system development can

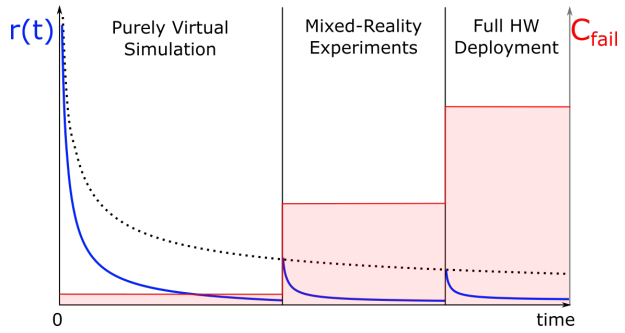


Fig. 7: Development of the failure rate $r(t)$ and costs caused by incidental failures C_{fail} in three different development stages

be modeled by Weibull distribution [29], [30]. This distribution models a decreasing failure rate caused by design faults and initial problems and their fixing in the burn-in phase. The failure rate function according to Weibull distribution is given by

$$r(t) = (\beta/\alpha)(t/\alpha)^{\beta-1}, \tag{2}$$

where α is called the scale parameter and $\beta \in (0, 1)$ is the shape parameter.

The development of the costs of failures $C_{fail}(\sigma_2)$ and the failure rate $r(t, \sigma_2)$ during the development process are depicted in Figure 7. When an error occurs, and is fixed, the system should be verified in all the previous development stages again, and the next stage should be started only after thorough testing to ensure that the error rate is low enough. This way, in stage s_x, most of the problems that would otherwise occur also in the stage s_{x+1} are discovered and resolved. Thus, a lower probability of error can be expected in the stage s_{x+1} compared to the stage s_x. Notice the sharp step changes in the error probability at the beginning of each stage in Figure 7. These step changes are caused by an occurrence of new errors that could not appear in the previous stages because of their lower fidelity. Over the time, across all the stages, the overall error probability should be decreasing, since it follows Weibull distribution in a burn-in stage.

4 Case Studies

4.1 Heterogeneous Team of Autonomous UAVs

The first herein presented use-case of the presented method and architecture is a development of the UAS with a heterogeneous team of autonomous UAVs. The system allows for the execution of complex tactical missions such as critical infrastructure protection, team area surveillance, or target tracking, as well as performing dynamic mission reconfiguration in a case of any change in the mission tasks or the number of available assets. Individual parts of this system are described in our previous works [9], [8], [10], therefore only a brief overview is presented here to make the paper self-contained.

The layers of the system architecture and their inner content, specific for this application, are depicted in Figure 8. The first five layers contain both real and virtual

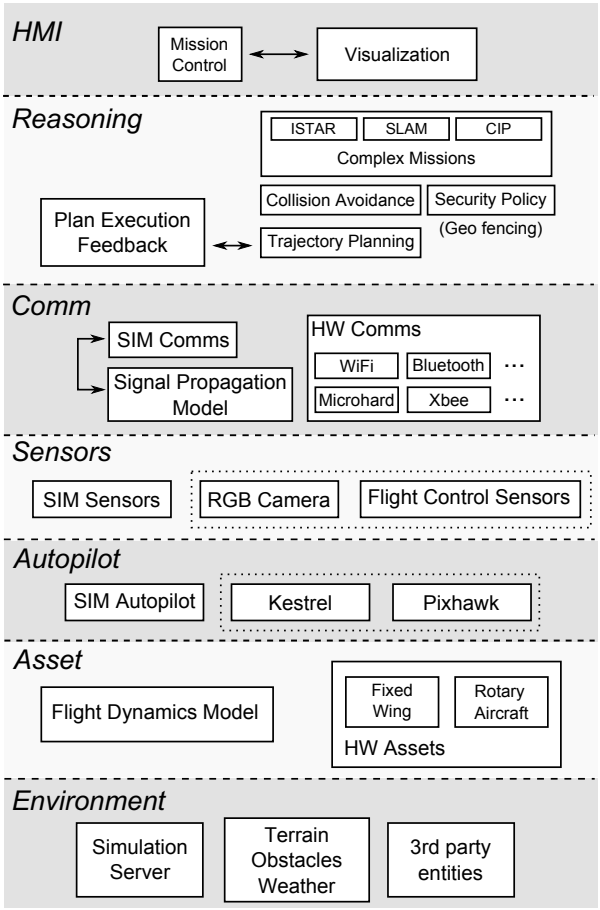


Fig. 8: Architecture of the system for command and control of autonomous UAVs

parts that can be mixed in the MR simulation during the incremental development process. A series of experiments have been performed with both the fixed wing and VTOL rotary UAVs with the aim to verify the architecture. Following the methodology, the experiments started with the pure virtual simulation of all unmanned assets and the environment using multi-agent simulations to verify basic functionality of the team interaction, coordination, and planning. Next, the interaction protocols for the intended autopilots (i.e., Kestrel, Pixhawk) have been implemented and a set of indoor hardware-in-the-loop simulations with the real modems and autopilots (with simulated sensory inputs and flight dynamics) have been conducted to verify the basic communication properties, autopilot APIs, and basic plan execution with more complex flight dynamics. After these tests, field experiments in MR simulations with one hardware UAV and several virtual ones have been performed to test the correct behavior of the planning algorithms in the wind and acquire characteristics of the communication devices on large distances. Finally, the experiments with multiple fully deployed hardware assets have been performed, e.g., see Figure 9.

Fig. 9: Flight tests of multi-UAV system

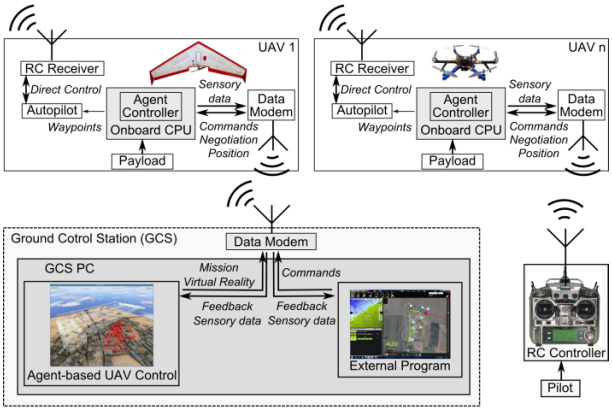


Fig. 10: Scheme of multi-UAV system deployment

The system has been deployed on the physical hardware according to Figure 10. It consists of two sub-systems: (i) on-board agent control and reasoning; and (ii) the ground control station (GCS) with mission control, simulation, and visualization. The agent control runs on-board of the UAV and data modem is used for the team coordination and communication with the operator. The GCS runs the simulation server and virtual UAVs all together with the simulated environment providing virtual moving ground targets, terrain, obstacles, etc. The GCS enables system operators to alternate the mission assignments, and third-party GCS software (such as UGCS, Virtual Cockpit, Mission Planner, etc.) can be connected for extended visualization options. Moreover, the control of each UAVs can be overtaken by a pilot using the Remote Controller (RC) to improve the safety of the operation that is usually required for legal reasons.

Adopting the strategy of the incremental development resulted in less flight test hours required for the system testing and verification, both for one, but mainly for two aircraft flying simultaneously. The proposed methodology shows itself advantageous for decreasing the deployment cost because the field tests were conducted only when they

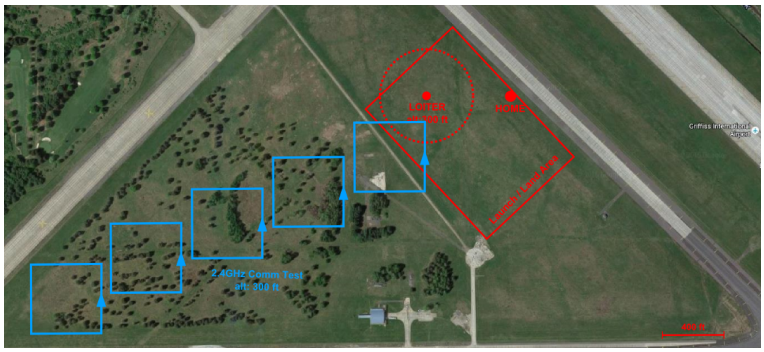


Fig. 11: Experimental setup for measurement of RF signal strenght.

were necessary for the verification of particular virtualization levels. Requirements and modeling of communication subsystems is a particularly important part of the methodology related to the practical deployment of the system to real UAVs. Therefore it is further described in the following subsection.

4.1.1 Communication Experiments

The communication subsystem produced the most significant differences between the computational models and measurements from the field experiments. The radio frequency bandwidth has to be wide enough to transmit telemetry, plans, commands, and other data required for proper system behavior. Also, the transmission delay has to be small enough to provide good response time and well-timed collision avoidance. The required minimal bandwidth and maximal delays have been estimated in simulated scenarios with a variable number of UAVs and tasks performed. However, experimental measurements of the signal strength have been performed to validate the used RF communication modems. Fixed wing aircraft with an omnidirectional antenna secured on its left wing has been scheduled to fly square patterns at various distances from the GCS, see Figure 11.

Early experiments show that a basic XBee-PRO DigiMesh 2.4 RF modules provide only limited communication bandwidth less than approximately 5 kBps, which is not sufficient for the system needs. Therefore, the modules have been replaced by Microhard nVIP2400 broadband modems capable of hypothetical data-rates of 54 Mbps and also feature TCP/IP protocol and communication encryption. The results showing the real signal strength and latency in various distances and heading configurations are shown in Figure 12.

The communication delay has an effect mainly in two situations: (i) the delay between the issuance of the operator's command and the start of the command execution; and (ii) the time from the moment a high-priority UAV successfully broadcast its plan to the moment another low-priority UAV with colliding trajectory starts execution of its collision avoidance (CA) maneuvers. Note that every UAV periodically broadcasts 50 seconds look-ahead part of its plan every 10 seconds. Variable throughput and delays have been caused by different channel properties and noise characteristics in every development stage. Together with the requirements on the bandwidth, and our available financial budget, these data have been used to select the most suitable

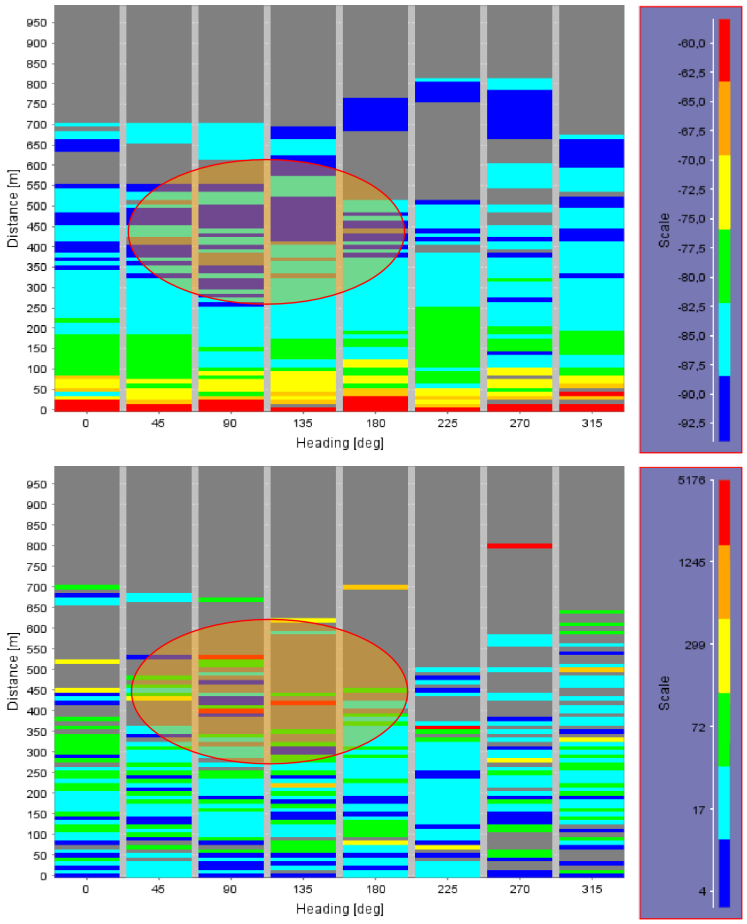


Fig. 12: Results of the communication tests - (a) RSSI [dBm] and (b) latency [ms] - with the Microhard nVIP2400 modem. The distance parameter is the distance between the aircraft and GCS and the heading parameter represents the relative heading attitude of the aircraft to the GCS. The highlighted area corresponds to the configuration where the UAV's antenna has been positioned away from the GCS.

communication hardware. The real measured throughput and latencies of the utilized Microhard nVIP2400 RF modems are depicted in Table 1. Relatively large delays in a case of the full hardware deployment have been still manageable for the correct system function of small UAV teams, but may strongly limit the scalability in the number of UAVs available in the system.

During these experiments, an anomaly in the signal propagation appears as follows. The signal quality is locally degraded in situations when the antenna points away from the GCS, and the aircraft's body and motor is in the way of the line of sight to the GCS (see the highlighted areas in Figure 12). This phenomenon has been observed by Meiser [31] and Mahdi et al. [32], but has not been studied in more detail yet. Very probably this RF signal hindering can be caused by aircraft's body and motor jamming.

Table 1: Data-link quality in different development stages.

System	**Data-link spec**	**Throughput** [Mbps]	**Execution delay** [ms]	**CA delay** [s]
Sim	localhost TCP/IP sockets	650.0	10	0
HIL	RF modems in lab	30.0	300	2
HW	RF modems in the field	0.5	1500	8

None of the existing wireless network simulators and no RF signal propagation models deal with this phenomenon, and thus the real world experiments have been necessary to discover this particular issue. Based on the experimental evaluation, this phenomena can be integrated into the system simulation to significantly increase its fidelity.

4.2 System for Verification of Collision Avoidance Methods

The next use-case of the presented development method is related to the project with Quanterion Solutions Inc., U.S. Air Force Research Laboratory (AFRL), and NUAIR alliance. The project goal has been to develop a robotic testing platform for verification of collision avoidance methods that would enable high-level command and control of multiple fixed-wing airplanes Desert Hawk III. The emphasis of the project has not been only on the flight safety in shared airspace and verification of cooperative collision avoidance among aircraft (in which the testing is crucial for achieving the desired reliability), but also on the training of operators of the final system.

The system is similar to the first use-case, and thus the individual layers of the architecture are similar to those in Figure 8. Additional extra RC and extra GCS with third-party software (Virtual Cockpit 3D) in the HMI layer of every single UAV have been utilized (for emergency situations only) because of increased safety requirements. A common GCS serving for multi-agent command and control has been provided, see Figure 13 for the overall schema of the deployed system. Besides, a new autopilot control protocol (AFRL's Common Communications Client for Payload Operations (C3PO)) with the increased level of control safety has been implemented and tested.

It is worth noting that the system has not been deployed and operated by us, but by members of the Quanterion Solutions with our remote assistance only. Thanks to the system modularity and existing implementations of system configurations of various virtualization levels, the operators have been trained iteratively. They started with training with a pure software simulation of the system, then using various degrees of mixed reality setting, and finally training with the real Desert Hawk III airplanes. The fast and successful training of the operators refers to simplicity and portability of the resulting system. Moreover, it also supports the suitability of the incremental development method even for the training activities. An example of the Desert Hawk III launch and field test settings is shown in Figure 14.

During this project, there have been extensive testing sessions. Not only for the reasons of the system development and verification but also for the system operators

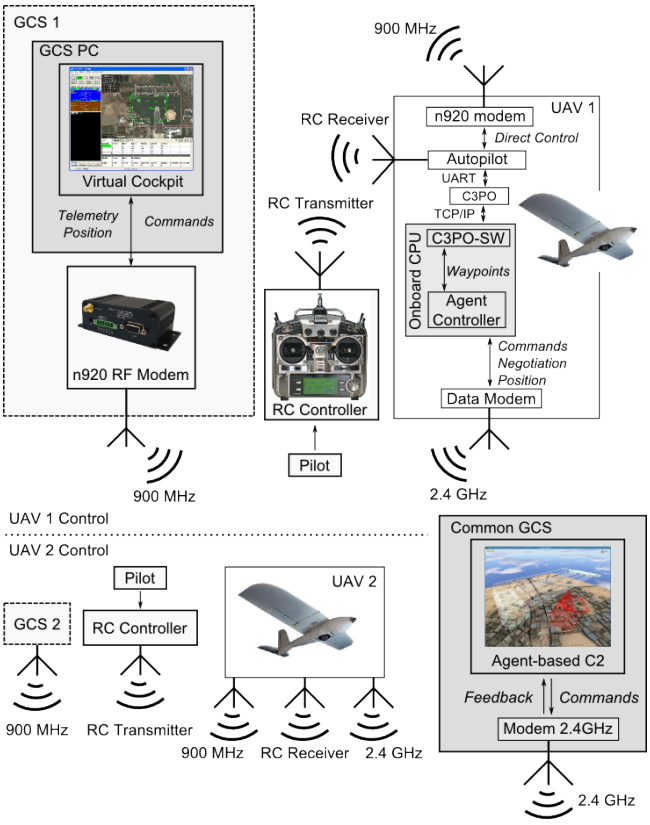


Fig. 13: Scheme of system deployment on Desert Hawks

Fig. 14: Flight tests with Desert Hawks III

to get familiar with the system safely. These operators then provided useful feedback that resulted mainly in changes of the HMI layer and integration of the third-party

software for monitoring and visualization (Virtual Cockpit[2] software). The modularity of the system and possibility to easily change its virtualization level allowed for the fast, safe, and successful training of the operators.

4.3 Safefly - System for Flight Safety of Light-Sport Aircraft

The third use-case where the herein presented architecture has been used for the incremental development is the project called *Safefly* in which we aim to build a system for increasing flight safety of light-sport aircraft. During this project, on-board sensors of the aircraft have been extended with collision avoidance sensors, communication equipment, and on-board computer. Algorithms for the trajectory planning and cooperative and non-cooperative collision avoidance have been deployed to plan the most suitable collision-free trajectories. The plans are provided as recommendations to the pilot, or, if possible, uploaded to a particular autopilot of the light-sport aircraft.

The system consists of hardware equipment that has been selected to satisfy the requirements of the final system on the computational power, reliability, communication bandwidth and latency, weight, and size as follows.

- ***TL-1000 Integra*** - an electronic flight instrument system (EFIS) that integrates all primary flight instruments, navigation, and 3-D terrain, together with a glass cockpit for various visualization options and with a possible connection to autopilot for the roll and pitch control. EFIS Integra systems have been deployed on two light-sport airplanes 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 the transponder responses from other aircraft to estimate their presence and position. Since a great majority of civil aircraft is equipped with a transponder, this system has been meant to provide non-cooperative detection and localization of surrounding aircraft.
- ***Microhard n920F*** - an RF module for cooperative plan exchange and negotiation of evasion maneuvers among an aircraft equipped with the Safefly system. It is also used to transmit the mixed reality data and information for visualization and debugging to/from GCS.
- ***Gumstix Overo FireStorm*** - an onboard ARM computational unit that is connected to all other system parts and runs the reasoning algorithms for planning, collision avoidance, and negotiation.

The respective layers of the system architecture are shown in Figure 15. We omit the software models in the individual layers since they are similar to those in Figure 8.

Accordingly to the concept of the incremental development, a flight dynamics model of the Tecnam aircraft has been implemented based on their BADA performance characteristics [33]. First, the algorithms for the trajectory planning [34] and cooperative negotiation-based collision avoidance [35] have been tested in a pure virtual simulation, see Figure 16.

Furthermore, from the simulated encounter models and from the requirements on the safety, the bandwidth requirements on the communication equipment have been estimated (see Figure 17). These data-rates reflect the aircraft speed and should be

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

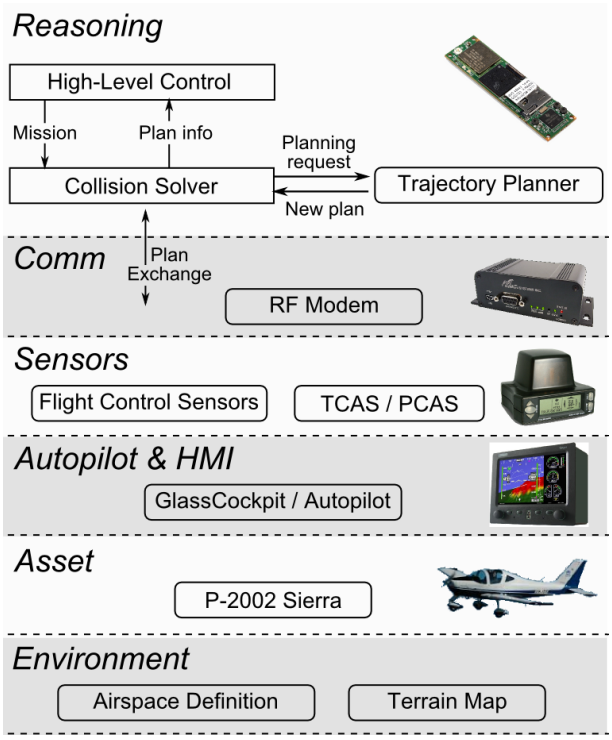


Fig. 15: Architecture of the system for light sport aircraft control with hardware components used for its realization

Fig. 16: Model of P-2002 Sierra utilizing BADA performance characteristics used for MR simulations

large enough for the negotiation algorithms to provide collision-free trajectories at least 30 seconds before a collision time for 1.5 km separation between airplanes.

Later, a model of the Zaon PCAS, based on its theoretical characteristics stated by the manufacturer has been integrated into the system to provide for non-cooperative collision avoidance. When the software model of the Integra EFIS has been obtained

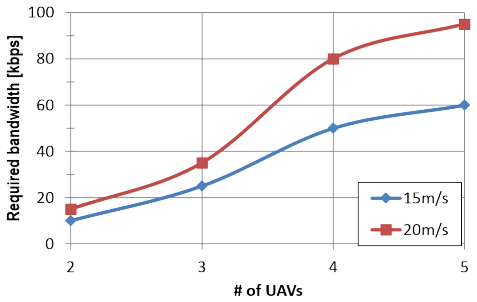
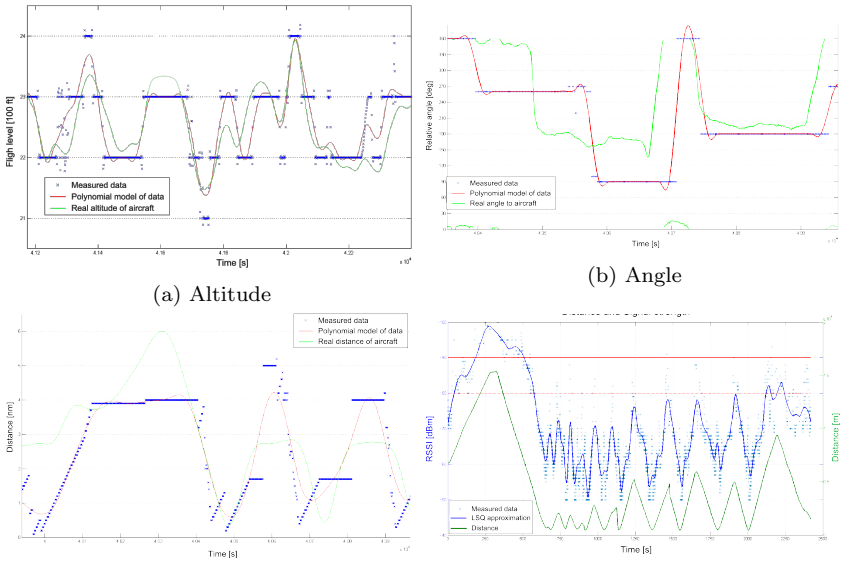


Fig. 17: Estimation of communication requirements based on software simulation results



(a) Altitude

(b) Angle

(c) Distance

(d) RSSI

Fig. 18: Results of the Zaon PCAS and RSSI measurements compared with the real data. Note that the estimation of the neighbor aircraft's altitude is relatively precise; however, the precision in the estimation of its angle and distance is low and delayed.

from its manufacturer, the interaction protocols for upload of plan waypoints into the EFIS have been implemented.

In the next stage, a series of field test experiments have been conducted to test the respective physical modules for reliability, to measure the bandwidth and range of RF modems, and real behavior of the Zaon PCAS, see the obtained results in Figure 18. There has been an intensive interaction with the aircraft pilots in various situations to verify the suitability of the proposed plans with the aim to increase the precision of the respective software models.

After the discussion with pilots, a plan visualization widget has been designed (see Figure 19) that is shown in the glass cockpit together with other flight data. It is

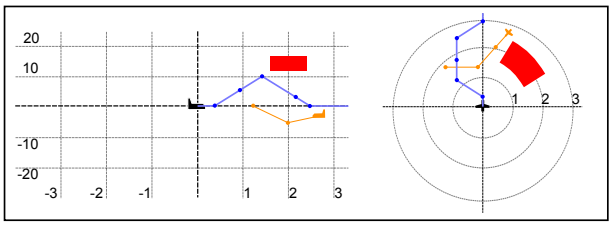


Fig. 19: Plan visualization in the Integra Glass Cockpit used for the situation awareness. The observations provided by the Zaon are in the red, and a trajectory of near cooperating aircraft is in the orange. The recommend flight path is the blue curve.

Fig. 20: Aircraft pilot panel with the integrated Safefly system – build-in EFIS Integra, Zaon PCAS with GPS on the flight panel and rest of the system integrated under the panel

helpful to the pilots in following the provided best collision-free path. Note how the imprecision of Zaon PCAS position estimation is reflected by larger dimensions of the red area.

Next, the Safefly system has been deployed in one of the experimental Tecnam planes, shown in Figure 20. It has been fed with the data of a virtual plane in the MR simulation, and the pilot reactions on the plan recommendations have been observed. After the integration of pilots' feedback (mostly related to visualization of the plan) and their reaction times into the system, and solving newly discovered hardware issues (mostly related to the inter-connection of the EFIS Integra and used Gumstix board as the on-board computer), the system has been fully deployed on two experimental airplanes.

5 Lessons Learned

The herein presented strategy of the incremental development with the utilization of the MR simulation showed to be useful in all the presented use cases. The MR simulation provided valuable insights into the systems that would be otherwise difficult to obtain in regular real-world experiments. It also helped to increase the fidelity of the first simulation scenarios without additional risks and costs. The following paragraphs summarize the most valuable lessons learned during the development of the presented systems.

- **Re-use of System Modules** The modularity of the multi-level architecture enables to re-use many of the system modules in various projects even if they are of different type. For example, the planning and collision avoidance algorithms, and also the communication protocols are usually very similar independently on the target application. In addition, it saves time and effort when repeatedly searching for hardware requirements since they may have been already estimated before. In particular, the communication device and onboard computers in the Safefly system have been selected based partially on our measurements of the system requirements from the previous projects.
- **Interchangeable Modules** Using the MR simulation with the incremental decreasing of the virtualization level makes a replacement of individual system parts a straightforward task. Since all the modules on the same architecture layer share the same interface, they are easily interchangeable. This way, the virtualization level of the system can be quickly altered back and forth to provide the best testbed configuration level needed at the moment. Moreover, an integration of a new module to the system then requires just an implementation of the device interface according to respective system layer and its step by step verification in particular development stages. This has been the case in the Safefly project when the manufacturer of Zaon XRX PCAS device stopped its production and support and some of its alternatives had to be used, i.e., the PowerFLARM.
- **Incremental Training of System Operators** Utilization of the MR simulations can significantly help target system operators to get familiar with the system without risk of damage to hardware or third-party property and also without any legal issues. The incremental decreasing of the virtualization level can be used in a learning process and teach the operators to use the system step by step. This is especially useful for aerial systems where the safety and legal issues are of the highest importance, as it has been found in our AFRL project where the operators had to undergo extensive hardware-in-the-loop testing before real field experiments.
- **Localization of Problems** The incremental development process helps in localizing problems that would not appear in the pure software simulation, and that would be difficult to isolate in the full hardware deployed complex system. For example, this has been the case of the effect of the communication delays and limited data-rate. Besides, similar issues have been observed for the autopilot behavior during path following.
- **Bridging Development Delays** In a case of cooperation of multiple parties on a project, such as we experienced during the development of the Safefly project, the MR simulation showed to be very useful for substituting delayed hardware parts. During the project, there have been significant periods where software or hardware delivery from the project partners has been delayed and would disable further work of the rest of the team. In these cases, the virtualization and modeling of the respective system layers showed to be beneficial since these models can be utilized as a substitute for the hardware without the need to delay further system development.
- **Problematic Wireless Communication Modeling** The MR simulations are difficult to be used for studying some of the issues related to wireless communication. The problem here lays in the fact that even when 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 their modems do not correspond to the assumed locations of the virtual entities cause incorrect

assumptions on the RF signal propagation, and thus possibly incorrect simulation results. This issue will be addressed in our future work.

6 Conclusion

In this paper, a methodology for the incremental development and verification of complex multi-robot systems with the use of the Mixed Reality simulations is formulated, and risk and cost analysis is provided to show its benefits. This approach has been validated, and its universality has been demonstrated by its utilization in 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.

Acknowledgements 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*, 27(2):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*, pages 393–406, 2009.
3. A. Komenda, J. Vokřínek, M. Čáp, and M. Pěchouček. Developing multiagent algorithms for tactical missions using simulation. *IEEE Intelligent Systems*, 28(1):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)*, pages 5382–5387, 2015.
5. I.Y.H. Chen, B. MacDonald, and B. Wunsche. Mixed reality simulation for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 232–237, 2009.
6. M.A. Day, M.R. Clement, J.D. Russo, D. Davis, and T.H. Chung. Multi-uav software systems and simulation architecture. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 426–435, 2015.
7. I.H.B. Pizetta, A.S. Brandao, and M. Sarcinelli-Filho. A hardware-in-the-loop platform for rotary-wing unmanned aerial vehicles. *Journal of Intelligent & Robotic Systems*, 84(725):725–743, 2016.
8. 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)*, pages J3–1. IEEE, 2015.
9. 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)*, pages 1407–1408, 2013.
10. M. Selecký and M. Rollo. Distributed control of heterogeneous team of autonomous uavs. In *Proceedings of EXPONENTIAL 2016: Association for Unmanned Vehicle Systems (AUVSI)*, pages 707–717, 2016.

11. Martin Selecký, Jan Faigl, and Milan Rollo. Mixed reality simulation for incremental development of multi-uav systems. In *Unmanned Aircraft Systems (ICUAS), International Conference on*, pages 1530–1538. IEEE, 2017.
12. 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)*, pages 269–275, 2011.
13. S. Demers, P. Gopalakrishnan, and L. Kant. A generic solution to software-in-the-loop. In *Military Communications Conference (MILCOM)*, pages 1–6. IEEE, 2007.
14. A.H. Goktogan and S. Sukkarieh. Distributed simulation and middleware for networked UAS. In *Unmanned Aircraft Systems*, pages 331–357, 2008.
15. D. Šišlák, P. Volf, Š. Kopřiva, and M. Pěchouček. Agentfly: A multi-agent airspace testbed. 2008.
16. A. Burkle, F. Segor, and M. Kollman. Towards autonomous micro UAV swarms. *Journal of Intelligent & Robotic Systems*, 61(1):339–353, 2011.
17. 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*, pages 89–96, 2008.
18. J.L. 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*, 84(1-4):779–797, 2016.
19. P. Chudy, P. Dittrich, and P. Rzucidlo. HIL simulation of a light aircraft flight control system. *IEEE/AIAA 31st Digital Avionics Systems Conference (DASC)*, pages 6D1–1, 2012.
20. M.E. Aydemir. Design and implementation of a compact avionics instrument for light aviation. *Turkish Journal of Electrical Engineering & Computer Sciences*, 24(5), 2016.
21. 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)*, pages 7A1–1, 2011.
22. K. Rydlo, P. Rzucidlo, and P. Chudy. Simulation and prototyping of FCS for sport aircraft. *Aircraft Engineering and Aerospace Technology*, 85(6):475–486, 2013.
23. 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*, 43(6):583–594, 2013.
24. J. Pellebergs and Aeronautics. The MIDCAS project. *Saab Aeronautics*, 2012.
25. 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)*, pages 5A1–1, 2015.
26. I.Y.H. 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*, pages 388–399, 2012.
27. M. Selecký and T. Meiser. Integration of autonomous UAVs into multi-agent simulation. *Acta Polytechnica*, 52(5):93–99, 2012.
28. L. Chiariglione. FIPA: Foundation for intelligent physical agents, 2001. [cited 5 May 2017].
29. F. Jensen and N.E. Petersen. *Burn-in: an engineering approach to the design and analysis of burn-in procedures*. Wiley, 1982.
30. M. Xie and C.D. Lai. Reliability analysis using an additive weibull model with bathtub-shaped failure rate function. *Reliability Engineering & System Safety*, pages 87–93, 1996.
31. T. Meiser. Distributed topology control in MANETs. Master's thesis, Czech Technical University in Prague, 2012.
32. Mahdi Asadpour, Bertold Van den Bergh, Domenico Giustiniano, Karin Hummel, Sofie Pollin, and Bernhard Plattner. Micro aerial vehicle networks: An experimental analysis of challenges and opportunities. *IEEE Communications Magazine*, 52(7):141–149, 2014.
33. 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*, 24(10):850–866, 2010.
34. David Šišlák, Přemysl Volf, and Michal Pěchouček. Accelerated A* path planning. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 1133–1134. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

35. David Šišlák, Přemysl Volf, Antonín Komenda, Jiří Samek, and Michal Pěchouček. Agent-based multi-layer collision avoidance to unmanned aerial vehicles. In *Integration of Knowledge Intensive Multi-Agent (KIMAS), International Conference on*, pages 365–370. IEEE, 2007.

Martin Selecký received his B.Sc. and MSc degrees in artificial intelligence from Czech Technical University in Prague, where he is currently working towards Ph.D. degree in Artificial Intelligence and Biocybernetics. His research interests include communication networks, development of complex autonomous multi-UAV systems, trajectory planning, and mixed-reality simulations.

Jan Faigl is an Associate Professor of Computer Science at the Faculty of Electrical Engineering (FEE), Czech Technical University in Prague (CTU). He received his Ph.D. (2010) in Artificial Intelligence and Biocybernetics and Ing. (2003) in Cybernetics from CTU. In 2013/2014, he was Fulbright visit scholar at the University of Southern California. He has been awarded the Antonin Svoboda award from the Czech Society for Cybernetics and Informatics in 2011. Since 2013, he is heading Computational Robotics Laboratory at the Artificial Intelligence Center (AIC), FEE, CTU. His current research interests are in unsupervised learning, self-organizing systems, navigation of mobile robots, and path and motion planning techniques.

Milan Rollo received his Ph.D. degree from Czech Technical University in Prague. He works as a senior researcher at the AI Center, Faculty of Electrical Engineering, CTU in Prague. His main research interests are agent-based systems and robotics, with focus on deployment of free-flight concept approaches on autonomous UAVs. He is a member of NATO NIAG SG-205 Sense and Avoid Feasibility and Certification for UAS Flight in Non-Segregated Airspace and EUROCAE WG-105 Unmanned Aircraft Systems (UAS).