Towards Incremental Development of Human-Agent-Robot Applications using Mixed-Reality Testbeds

Michal Jakob, Michal Pěchouček, Michal Čáp, Peter Novák and Ondřej Vaněk

Abstract—Testing and evaluation is an essential part of developing applications involving teams of humans, agents and robots. In such applications, individual algorithms cannot be reliably tested in isolation because their performance depends on complex interactions with the environment and with other humans, agents and robots involved. At the same time, testing with real robots and human individuals is costly and potentially dangerous. We therefore propose an incremental development framework employing mixed-reality testbeds, which aims at reducing development costs and risks by fully or partially substituting parts of the application and the surrounding reality with computational models. The framework parameterizes such mixed-reality testbeds in terms of their fidelity and size, and comes with guidelines for managing the two parameters through a sequence of iterations so as to maximize the effectiveness of system development. The framework is illustrated on an example application in the domain of multi-UAV tracking of mobile targets.

Index Terms—Multiagent Systems, Simulation, Modeling, Autonomous vehicles, Methodology, Testing

I. Introduction

Technological progress has now reached a point where applications involving autonomous robotic assets become reality in numerous domains. With continual developments towards applications involving mixed human, agent and robot teams (termed HART applications hereafter), we witness a growing need for efficient methods and processes for developing such applications. Due to requirements on efficiency, reliability and robustness of such systems in real-world conditions, no development methodology for HART applications can be effective without strong support for realistic evaluation and testing.

In contrast to standalone software systems, the operation of HART applications depends on factors beyond the actual software logic, in particular on the characteristics of the hardware (sensors, actuators and communication links), the dynamics of the environment and the behavior of the humans involved. A reliable assessment of HART applications cannot be obtained without a testbed that approximates these factors with a sufficient level of fidelity. In general, the most reliable assessment is obtained if the application is tested in the full target configuration, i.e., with the complete set of hardware assets and human individuals operating within the target physical environment. Unfortunately, such full-configuration tests are very expensive in terms of cost, time and resources, and may carry substantial risks (consider e.g., testing a collision avoidance functionality between UAVs). This makes full-configuration testing uneconomical and impractical in early stages of application development, when large amount of evaluation and testing needs to be performed quickly to assess a multitude of alternative design options.

To reduce assessment costs and risks, developers of HART applications may employ simplified testbeds which approximate the target application setup. Environment, hardware and/or human actors can be fully or partially substituted with computational models, making assessment faster and less costly, albeit at the expense of introducing potential assessment errors. The practice of using computational models is widespread in many areas of engineering, including the development of robotic systems [1].

In the case of HART applications, however, determining which parts of the application should be approximated with computational models is difficult due to a large number of involved entities and their dependencies. In this paper, we address this issue and lay foundations for a development approach that allows balancing assessment cost and accuracy throughout the system development process. We propose and formalize the concept of incremental multi-level mixed-reality development that allows to use mixed-reality testbeds of various size and levels of virtualization, and utilize them in a way that maximizes the effectiveness of HART application development.

A. Example: Multi-UAV Tracking

Consider the tasks of mobile target tracking by teams of unmanned aerial vehicles (UAVs). The objective is to provide updates on the location and activity of a number of ground mobile targets, where the number of entities of interest may be much higher than the number of UAVs. The team of UAVs should operate autonomously with only high-level supervision by a human operator. Developing reliable control and coordination algorithms for autonomous teams of UAVs is a challenging problem, especially when taking into account complex operational environments (e.g., urban area) or the ability of targets to purposefully evade detection. Moreoever, in scenarios such as border patrolling, the monitoring assets may also include unattended ground sensors and on-the-ground human or robotic patrolling units.

All authors are with the Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, Karlovo námestí 13, CZ-121 35 Prague 2, Czech Republic, e-mail: firstname.lastname @fel.cvut.cz

Please cite as: Michal Jakob, Michal Pěchouček, Michal Čáp, Peter Novák, Ondřej Vaněk, "Mixed-Reality Testbeds for Incremental Development of HART Applications," IEEE Intelligent Systems, vol. 27, no. 2, pp. 19-25, Jan. 2012, doi:10.1109/MIS.2012.2

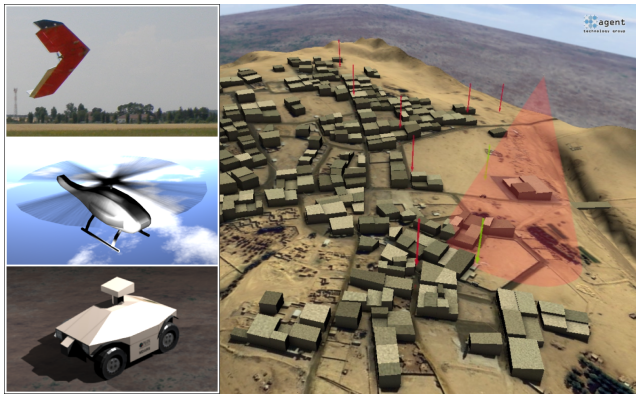

Figure 1: AGENTSCOUT: Mobile targets tracking and area/perimeter patrolling using unmanned aerial vehicles. Example application developed with the help of mixed-reality testbeds.

Developing coordination algorithms for such mixed teams of mobile robots, static sensors and human patrols requires an iterative development and testing process. As part of our work towards implementing the above outlined tracking capability in the AGENTSCOUT simulator[1] [2], we have developed and applied such a process to an extent. The experience obtained motivated the approach introduced in this paper.

II. MULTI-LEVEL MIXED-REALITY TESTBEDS

In general, there are different ways in which target deployment setup can be approximated in order to make application tests faster and less expensive. Since our interest is in multi-entity systems involving a number of human actors and robotic assets, we consider two principal dimensions in which the approximation can proceed: (1) the level of virtualization with which the target setup is represented, and (2) the number of entities in the test scenarios. Other approximation dimensions could be introduced as long as they allow trading test costs for test accuracy.

A. Approximation Dimensions

The *level of virtualization* denotes the extent to which the target application setup is virtualized in a given testbed configuration. That is, the extent to which parts of the setup are substituted with synthetic computational models. At one extreme, in the full target setup, no entities are virtualized, i.e., only physical hardware platforms and/or human actors are employed. Starting from the zero virtualization setup l_0, with the increasing virtualization, individual entities of the application are gradually replaced with computational models. The process of gradual virtualization gives rise to a sequence of *levels of virtualization* labeled $l_0, l_1, \ldots, l_n$ from the system fully deployed in physical reality, the zero virtualization setup, to the completely virtualized setup l_n with fully synthetic computational representation of system entities. In between the two extremes lie *mixed-reality* testbeds, such as hardware-in-the-loop simulations or testbeds involving human actors through virtual reality. Note that within a single testbed configuration, different levels of virtualization can be assigned to different entities. In most cases, a lower level of virtualization facilitates a more reliable testing but is more time and resource consuming.

The core idea underlying the development process proposed below is to start at a relatively high level of system virtualization and then iteratively decrease the virtualization until reaching the target deployment setup.

The other dimension along which the target application setup can be approximated is the number of autonomous entities with which the application is tested. Instead of having the same full number of robots and/or humans as in the full application, development and testing can initially be performed using only a subset of entities. We call the number of entities used the *size* of a testbed configuration.

The reduction of testbed size generally leads to cost savings, especially when physical hardware or human actors are involved. Reducing the number of entities below a certain threshold, however, can undermine the ability to test collective behavior properties of the application.

B. Testbed Configurations and Fidelity

In order to specify the distribution of levels of virtualization in a compact form, we define *testbed configuration* σ as a vector $\sigma = (s_0, s_1, \ldots, s_n)$, where n is the number of virtualization levels and s_i is the number of entities modeled at the level of virtualization l_i. The *size* of the testbed configuration, i.e., the number of entities used, then corresponds to $\eta(\sigma) = \sum_{i=0}^{n} s_i$. *Target configuration* σ_T represents the target system in full with zero virtualization. Typically σ_T will be of the form $\sigma_T = (k, 0, \ldots, 0)$ for some $k \geq 0$.

The space of all possible testbed configurations can be visualized in a plane as shown in Figure 2. The horizontal axis denotes the size of testbed configurations; the vertical axis denotes an abstract measure representing the *aggregate level of virtualization (LoV)*, defined as a weighted average of virtualization levels over all entities in the given testbed configuration.

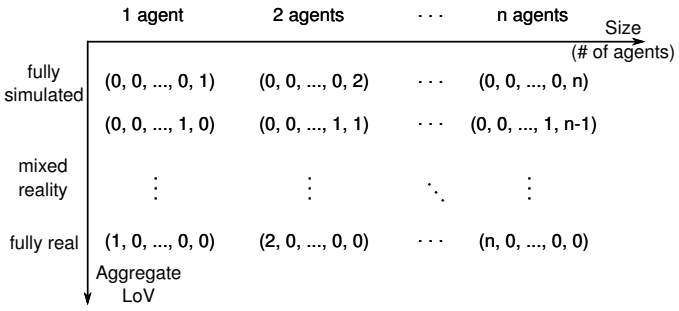

Figure 2: The space of testbed configurations arranged according to testbed size and its aggregate level of virtualization.

Recall that testbeds are used to test the performance of the developed application. Unless the testbed corresponds to the target configuration σ_T, the performance assessed on the testbed can differ from the performance in the target configuration. To capture this difference, we introduce the

[1] http://agents.fel.cvut.cz/projects/agentscout2/

concept of the testbed error and testbed fidelity. The *testbed error* $\epsilon(\sigma) \in [0, \infty)$ is the distance between state-space execution traces produced using the given testbed configuration and using the target configuration, averaged over all possible application runs. The *testbed fidelity* $\phi(\sigma) \in (0, 1]$ is then defined as

$$\phi(\sigma) = \begin{cases} 1 & \text{if } \epsilon(\sigma) = 0 \\ \tanh(\frac{1}{\epsilon(\sigma)}) & \text{otherwise} \end{cases}$$

If a testbed configuration σ fully replicates the behavior of the target setup, then $\epsilon(\sigma) = 0$ and $\phi(\sigma) = 1$. The higher the fidelity of a testbed, the higher the accuracy of assessments obtained on it. Note that a different sigmoid function can be chosen for fidelity calculation as long as it maps testbed error on $(0, 1]$ interval.

C. Comparing Testbeds

The core of the proposed approach to HART application development is the iterative evaluation of developed applications in gradually more and more realistic setups. Expressed in terms of testbed configurations, the incremental development process should employ a sequence of testbed configurations with increasing fidelity. Unfortunately, in practice, determining such a sequence is not directly feasible as testbed fidelity cannot be determined without executing tests on the full target configuration and comparing results.

Instead, we therefore use approximate fidelity ordering constructed based on the following assumption valid in most domains: a testbed configuration σ' is expected to have higher fidelity than σ, if either (1) testbed size increases while the level of virtualization does not increase, or (2) the level of virtualization decreases while the testbed size does not decrease. More formally, we define the *approximate fidelity ordering relation* $\succ$ on testbed configurations as follows. For configurations $\sigma = (s_0 \dots s_n)$ and $\sigma' = (s'_0, \dots s'_n)$, we denote $\sigma' \succ \sigma$ if and only if either

1) for all i we have $s'_i \geq s_i$ and there exists j, s.t. $s'_j > s_j$ and thus $\eta(\sigma') > \eta(\sigma)$; or
2) $\eta(\sigma') = \eta(\sigma)$, and there exists j, s.t. $s'_j > s_j$ and for all $i < j$ we have $s'_i \geq s_i$.

We use the approximate testbed fidelity ordering to navigate the space of testbed configurations during the development process.

D. Virtualization Levels in HART Applications

In the case of applications involving human-agent-robotic teams, specific virtualization techniques can be used. We present some of them below in the decreasing order of virtualization.

1) Robot/Hardware Entities:

Fully simulated hardware (L_{FS}): a hardware asset is substituted by a fully synthetic computational model such as out-of-the-box robotic simulators (e.g., *Gazebo* [1]) able to simulate physical and electronic properties of a number of different robotic platforms.

Hardware-in-the-loop (L_{HIL}): even the most sophisticated robotic simulators do not capture all the phenomena that may arise in a real hardware asset. Hardware-in-the-loop setup can be employed to increase the hardware fidelity of the testbed. In this setup, a hardware asset is tested in a laboratory setting with sensory input signals provided by a simulator and actuator signals controlling a simulated physical model. The approach is often used to verify the function of hardware platform electronic and communication subcomponents and is therefore useful in single-robot scenarios.

Augmented reality/hybrid simulation (L_{HS}): as a next step towards the full hardware testbed, an augmented reality/hybrid simulation setup can be used. In this setup, hardware assets operate in the target physical environment, but the simulator augments their sensory input with objects that only exist in the simulation. The approach is particularly useful in multi-agent scenarios where interaction between multiple robots is to be tested but fewer than the target number of physical robots are available.

Full hardware (L_0): target setup with robots assets represented by target robotic platforms operating and interacting in the target physical environment.

2) Human Agents:

Computational behavior models (L_{BM}): at the highest level of virtualization, human actors can be substituted by computational behavioral models. These models can be constructed manually based on an expert input or learned automatically from past observations of human behavior. Different models of human decision making can be employed, such as prospect theory, bounded rationality or quantal response equilibrium.

Data-feeds about human behavior (L_{DF}): in this setup, data feeds of past real human behavior are replayed by the testbed during application test runs. This type of virtualization typically provides only a unidirectional link between the human actors and the rest of the application (i.e., the humans cannot react on the output of the application).

Virtual reality (L_{VR}): in setups employing virtual reality, real humans are involved during testing. Instead of operating directly in the target environment, however, they are immersed in its virtual reality approximation. This allows testing aspects of human behavior which are difficult to capture via computational models and facilitates testing of phenomena such as bounded rationality, irrational behavior or decision making affected by immediate mental attitudes, such as stress, fear, anger, or joy.

Simulated human-machine interaction (L_{SI}): even closer to the physical reality, in this setup real humans are used in the target environment but some parts of their interaction with other entities remains simulated. This allows testing coordination algorithms even with incomplete hardware capability (sensory or other). The setup provides a human-oriented equivalent of hybrid simulations from the perspective of hardware assets.

Full human involvement (L_0): real humans acting in the target environment are employed.

III. Multi-Level Incremental Development

The main motivation for introducing the concept of a testbed configuration is to provide a framework for describing iterative strategies for HART application development. A key idea in determining such strategies is to use a sequence of testbeds with different test accuracy and cost in order to have each iteration provide maximum feedback on design and implementation choices faced in the development of the application.

A. Cost

In order to be able to express the above idea more accurately, we introduce the notion of *iteration cost* $cost(\sigma_1, \sigma_2)$, which is meant to represent the total cost associated with getting from an application that works correctly on a testbed configuration σ_1 to an application that works correctly on a testbed configuration σ_2. In general, the overall iteration cost can be decomposed into

1) *testbed cost*, the cost of providing a testbed with configuration σ_2;
2) *development cost*, the cost of modifying the application logic to work correctly on testbed σ_2; and finally
3) *test cost* of verifying the modified application works correctly on testbed σ_2.

B. Iteration Strategy

The ultimate challenge in our proposed methodology is to find an optimal iteration strategy through the space of testbeds configurations, i.e., a sequence of configurations $\sigma_0, \sigma_1, \ldots, \sigma_n$ such that $\sigma_n = \sigma_T$ and $\sum_{i=1}^{n} cost(\sigma_{i-1}, \sigma_i)$ is minimal.

Unfortunately, except for trivial cases, determining the optimal strategy *a priori* is not feasible in real-world cases due to domain-dependency and uncertainty in the estimates of all cost components. However, assuming that highly virtualized testbeds allow faster iterations than testbeds with low virtualization, a reasonable strategy is to start from highly virtualized, albeit likely lower-fidelity testbeds. Subsequently, as the uncertainty about the application design and the underlying logic decreases, the development should move towards higher-fidelity testbeds, even though these are likely to have higher test cost and therefore allow only a lower number of tests. In result, the fidelity of the employed testbed increases as the developed HART application matures. Such an iteration strategy can be captured by the following algorithm:

1) choose an arbitrary starting configuration σ,
2) develop a testbed with the configuration σ,
3) develop/modify/debug the application until it works correctly on σ,
4) unless $\sigma = \sigma_T$, choose another configuration σ', s.t. $\sigma' \succ \sigma$ and proceed to (2) with σ',
5) end otherwise.

Clearly, in typical scenarios involving several levels of virtualization in multi-robot systems featuring number of agents, there will be several ways to construct the sequence of testbed configurations leading from a reasonable synthetic testbed to the target deployment setup. For instance, it will be up to the developer to choose whether it is more reasonable to first scale the algorithms with respect to the number of simulated robots, and only then start to port the system to real hardware, or the other way round. The core of the iterative development strategy should however remain, i.e., to gradually refine the system setup to approach the target deployment scenario. Note that in general, the space of testbed configurations has as many dimensions as there are levels of virtualizations, making the number of possible iteration strategies huge. Better understanding of the structure and properties of the error, fidelity ordering and iteration cost function is therefore essential for obtaining intuition or possibly even more formal rules by which the iteration strategy should be determined.

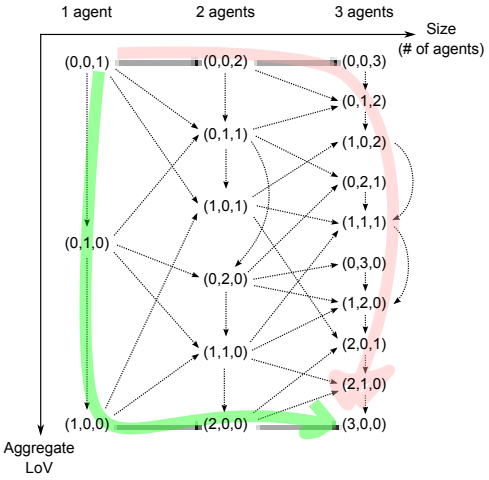

Figure 3: Space of testbed configurations for the multi-UAV tracking application, together with depiction of two possible development strategies.

IV. Example: Iterative Development of Multi-UAV Tracking Application

Consider the task of developing a coordination mechanism for a team of UAVs cooperatively tracking a number of humans in an urban environment. Specifically in our case, we want to use *Unicorn* UAV platforms by *Procerus Technologies*. Two such aircrafts were available to our team at the time of the example system development.

Let us consider three levels of virtualization of the system. Besides the fully simulated setup L_{FS} and the fully physical target setup L_0, we also consider the intermediate augmented reality setup L_{HS}. The intermediate setup is necessary because of only two hardware UAV platforms available for the development, while the coordination mechanism has to be tested with larger teams of UAVs.

Starting from the fully simulated setup L_{FS}, there are numerous ways to traverse the space of testbed configurations; Figure 3 depicts a fragment of the configuration space. The underlying directed graph connecting the depicted configurations corresponds to the approximate fidelity ordering relation introduced in the previous section. The two depicted development strategies correspond to two extreme cases described below.

According to the first strategy, the developer should first scale the coordination algorithms to the target number of simulated UAVs in a fully simulated environment (L_{FS}). Subsequently, the control code running on-board of individual UAVs should be ported one by one from the simulation to physical UAVs embedded in an augmented reality environment (L_{HS}). In each step, the development should go through the full port–extend/adapt–test–evaluate cycle, until the desired level of performance is reached. This way, the development would eventually reach a state when all aircraft control agents are deployed and working correctly on target hardware UAV platforms with augmented sensory input feeds. In the final round of iterations, individual UAVs should be disconnected from the augmented reality environment and placed in the target physical environment (L_0).

The alternative strategy would dictate to initially work with one UAV only. Immediately after having the UAV control logic correctly working in the simulation (L_{FS}), the logic would be ported to the UAV embedded in the augmented reality (L_{HS}) and, after having been adapted to work correctly, ported on a physical UAV (L_0). Only once the system worked correctly on a single hardware UAV in the physical environment, additional UAVs would be added, first in simulation (L_{FS}) and then gradually migrated through augmented reality (L_{HS}) to the full deployment setup (L_0).

The choice between the two strategies would typically depend on whether more implementation uncertainty lies with the ability of realize collective team behavior or deploying the control logic on the target hardware platform.

In parallel, human entities in the system, acting as object of tracking, can also be involved at different level of virtualization. Initially, all human subjects can be represented using computational behavior models L_{BM}. In the second step, some human actors can be approximated on the L_{VR} level of virtualization in the minimal virtual reality settings (e.g., graphical user interface and a joystick). In the third step, real humans at the L_{SI} level, i.e., moving in the target physical environment but with simplified interaction with the robotic assets can be used. In this particular case, in order to avoid the necessity to have on-board image recognition algorithms working, we could equip the human individuals with a location broadcasting device (e.g., based on GPS-enabled Internet-connected smartphones) which would continuously supply the location of the tracked individuals to the UAVs directly. This would allow tuning high-level aircraft coordination algorithms independently from developing and testing image recognition algorithms.

V. Related Work

Virtual reality is concerned with methods that allow a human user to observe and interact with non-existing virtual worlds [3]. In mixed reality—as we describe above—the agent observes a world that is partially real and partially virtual. This idea is captured by the concept of *reality–virtuality continuum* [4], a continuous scale ranging from fully real to fully virtual worlds. In between the two extremes stands mixed-reality, typically implemented either as an augmented reality (the perception of the real world is augmented with virtual objects) or augmented virtuality (the virtual world is augmented with elements of physical reality). Currently, the research in mixed-reality is mostly revolving around interface devices allowing a human user to observe and interact with mixed-reality worlds. We depart from the humans-only mixed-reality concept and apply similar techniques for cost effective evaluation of systems involving robots too. For this, we consider not only humans, but also both real and simulated robotic assets to observe and participate in the mixed-reality world.

Developers of control programs for autonomous robotic systems nowadays routinely use sophisticated simulators to test the function of their programs prior to the deployment on target hardware. To further increase the fidelity of such testing, one or more physical hardware assets can be included as a part of the simulation. Such simulations are, depending on the context, termed *hardware-in-the-loop simulations (HIL)* [5], *hybrid simulations (HS)* [6] or *mixed-reality simulations (MRS)* [7].

The idea of evaluating control algorithms for multi-robotic system in environments that mix both real and simulated entities is over twenty years old. An initial attempt dealing with simulation of industrial robots was published in 1989 [8]. More recently, mixed-reality simulations have been employed during the development of autonomous robotic assets. Chen et al. [7] introduced a mixed reality simulation library for the Gazebo [1] 3D mobile robot simulator. Using this library, a real hardware robot can interact with the Gazebo simulated world, which can be used both to augment the environment of the real robot with virtual objects and to provide a visual feedback on the state of the robot's perception through 3D visualization. Hybrid and hardware-in-the-loop simulations have been also reportedly used in the domain of autonomous underwater vehicles [6].

Unlike the above listed contributions, besides utilizing concepts such as the mixed-reality simulation, we additionally introduce a methodological approach towards better understanding of iterative development of mixed reality HART applications. Some of the ideas underlying the proposed framework have been proposed in our earlier work [9] which focused on using simulations to accelerate the development of multi-agent applications. The proposed approach has been also followed in the development of piracy counter-measure coordination system developed within the AgentC project[2] [10], as well as in the process of porting collision-avoidance algorithms to UAVs in the context of the project AgentFly[3].

VI. Conclusion

We have outlined a vision for accelerated development of advanced human-agent-robot teamwork applications. As the number of autonomous entities and the complexity of their interactions increase in HART applications, the testing strategy becomes a critical part of application development. To improve the speed and cost efficiency of the development process,

[2]http://agents.fel.cvut.cz/projects/agentc/

[3]http://agents.fel.cvut.cz/projects/agentfly/

testing should take advantage of mixed-reality testbeds that approximate the application's deployment setup with a variable degree of fidelity. We have provided a conceptual framework in which testbed configurations and testing strategies can be described and optimum strategies balancing testbed fidelity with assessment cost can be sought.

The presented results are only a first step towards a comprehensive methodology for incrementally developing HART applications. Further research is needed to better understand how different combinations of testbed sizes and virtualization levels affect testbed fidelity and individual components of iteration cost. Although it is likely that strong, prescriptive iterative development guidelines can only be found for specific subcategories of HART applications, the proposed common conceptual framework allows the comparison of different guidelines and promotes the transfer of methodological knowledge across domains.

ACKNOWLEDGMENTS

The paper has been originated during sabbatical of Michal Pěchouček at the University of Southern California supported by the Fulbright Commission. Furthermore, the presented work was supported by US Army CERDEC grants no. W911NF-10-1-0112 and W911NF-11-1-0252, Office of Naval Research grant no. N000140910537 and the Grant Agency of the Czech Technical University in Prague grant no. SGS10/189/OHK3/2T/13.

REFERENCES

[1] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2004.

[2] J. Vokřínek, P. Novák, and A. Komenda, "Ground tactical mission support by multi-agent control of UAV operations," in *Proceedings of 5th International Conference on Industrial Applications of Holonic and Multi-Agent Systems (HoloMAS)*, ser. Lecture Notes in Artificial Intelligence, vol. 6599. Springer.

[3] G. Burdea and P. Coiffet, "Virtual reality technology," *Presence: Teleoperators and Virtual Environments*, vol. 12, no. 6, pp. 663–664, 2003.

[4] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino, "Augmented reality: A class of displays on the reality-virtuality continuum," in *Proceedings of the Telemanipulator and Telepresence Technologies*, 1994.

[5] D. Lane, G. Falconer, G. Randall, and I. Edwards, "Interoperability and synchronisation of distributed hardware-in-the-loop simulation for underwater robot development: issues and experiments," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*. IEEE, 2001.

[6] B. Davis, P. Patron, and D. Lane, "An augmented reality architecture for the creation of hardware-in-the-loop; hybrid simulation test scenarios for unmanned underwater vehicles," in *Proceedings of the OCEANS*, October 2007.

[7] I. Y. H. Chen, B. MacDonald, and B. Wünsche, "Mixed reality simulation for mobile robots," in *Proceedings of the international conference on Robotics and Automation (ICRA)*. IEEE, 2009.

[8] F. Cao and B. Shepherd, "MIMIC: a robot planning environment integrating real and simulated worlds," in *Proceedings of the International Symposium on Intelligent Control*. IEEE, 1989.

[9] M. Pěchouček, M. Jakob, and P. Novák, "Towards simulation-aided design of multi-agent systems," in *Post-proceedings of the 8th International Workshop on Programming Multi-Agent Systems (PROMAS)*, 2010.

[10] M. Jakob, O. Vaněk, and M. Pěchouček, "Using agents to improve international maritime transport security," *IEEE Intelligent Systems*, vol. 26, no. 1, pp. 90–96, January 2011.

Michal Jakob is a senior researcher in the Agent Technology Center, Dept. of Computer Science, FEE, CTU. Michal Jakob received a PhD degree in Artificial intelligence and biocybernetics from Czech Technical University in Prague in 2008. Michal currently leads center's research on modeling and optimizing multi-modal transport systems, primarily as part of the international EC-funded SUPERHUB project. In addition, Michal leads the AgentC research project, supported by the U.S. Office of Naval Research, developing novel techniques for analyzing, modeling and ultimately disrupting maritime piracy. On a more theoretical level, Michal's research interests include agent-based simulation, computational game theory and multi-agent systems. Before joining the Agent Technology Center, he was a research scientist with British Telecommunications (BT), developing resilient decentralized architectures for service-oriented computing and network-centric information fusion.

Michal Pěchouček is a full professor in cybernetics at the Czech Technical University (CTU), the deputy head of the Department of Computer Science at CTU and the head of the Agent Technology Center at CTU. The research interests of Michal Pěchouček lie mainly in the fields of multi-agent simulation and modeling, coordination, social knowledge representation, multi-agent planning, multi-agent prototypes and test-beds and applications of agent-based computing into security related applications, UAV robotic coordination and air-traffic control. Michal Pěchouček has been a PI on more than 30 research contracts and grants provided by US Air Force, US Army CERDEC, and Office for Naval Research and a range of industrial partners. He has received a number of awards for technical excellence including the Czech Mind for Invention in 2010, Google research award in 2009, and Czech Engineering Academy Award 2007. Michal is an honorary member of Artificial Intelligence Application Institute at University of Edinburgh and a member of advisory board of the Center for Advanced Information Technology, University of Binghamton. He is a co-founder of Cognitive Security and AgentFly Technologies start-up companies.

Michal Čáp is a researcher at the Agent Technology Center, Czech Technical University in Prague. Michal Čáp received his MSc from the Utrecht University. His research interest include multi-agent systems, in particular agent-oriented programming, agent-based simulations, distributed coordination and design of multi-robotic systems.

Peter Novák is a post-doctoral researcher at Agent Technology Center of the Department of Computer Science and Engineering at the Czech Technical University in Prague, Czech Republic. Peter Novák obtained his PhD in computer science from the Clausthal University of Technology, Germany (2009). His research interests revolve around interaction of cognitive agents with dynamic environments, in particular cognitive robotics, multi-agent planning and multi-agent coordination.

Ondřej Vaněk is a researcher and a PhD student at the Agent Technology Center, Czech Technical University in Prague. His current research is focused on multi-agent simulations and application of cooperative and non-cooperative game theory on securing complex critical infrastructures. He currently works on an U.S. Office of Naval Research funded project from the maritime domain AgentC, researching techniques able to minimize negative impacts of modern maritime piracy on international shipping industry.

Ondřej Vaněk graduated from Faculty of Electrical Engineering Czech Technical University in Prague in Technical Cybernetics in 2008. Prior to his current position, he was visiting researcher in Rockwell Automation Center in Cleveland and he worked as a Java programmer and IT analyst for various companies.