Mixed-Reality Testbeds for Incremental Development of HART Applications

Michal Jakob, Michal Pěchouček, Michal Čáp, Peter Novák and Ondřej Vaněk, *Czech Technical University*

An incremental process for developing human-agent-robot applications uses mixed-reality testbeds of varying fidelity and size.

Thanks to technological progress in recent years, autonomous robotic assets now play a role in many real-world endeavors. With continued demand for applications involving a mixture of human, agent, and robot teams (HARTs), there's a growing need for efficient methods and processes to develop such applications. Because of the requirements of efficiency, reliability, and robustness of such systems in real-world conditions, no development methodology will 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 requires a *testbed* that approximates these factors with a sufficient level of fidelity. In general, testing an application in the full target configuration (that is, with the complete set of hardware assets and human individuals operating within the target physical environment) provides the most reliable assessment. Unfortunately, such full-configuration tests are expensive in terms of money, time, and resources, and could carry substantial risks—for example, testing a collision avoidance functionality between unmanned aerial vehicles (UAVs). This makes full-configuration testing impractical in the early stages of application development, when developers must perform a lot of evaluation and testing quickly to assess multiple design options.

To reduce costs and risks, application developers can employ simplified testbeds that approximate the target application setup. These testbeds can fully or partially replace the environment, hardware, and human actors with computational models, albeit at the expense of introducing potential assessment errors. Using computational models is common in many areas of engineering, including the development of robotic systems.[1]

1541-1672/12/$31.00 © 2012 IEEE
Published by the IEEE Computer Society

Figure 1. The AgentScout project focuses on tracking mobile targets and patrolling areas and perimeters with teams of unmanned aerial vehicles. Mixed-reality testbeds helped develop the algorithms for coordinating multiple vehicles.

In the case of HART applications, however, determining which parts of the application to approximate with computational models is difficult due to the large number of involved entities and their dependencies. In this article, we address this issue and lay the foundations for an approach that balances assessment cost and accuracy throughout the development process. We propose and formalize the concept of *incremental multilevel mixed-reality development*, which lets us use mixed-reality testbeds of various sizes and virtualization levels in a way that maximizes the effectiveness of HART application development.

This approach arose from our experience with the AgentScout project (see Figure 1), which focuses on developing coordination algorithms for mixed teams of mobile robots, static sensors, and human patrols.[2] Our work dealt with implementing the ability for teams of UAVs to track mobile targets autonomously, with only high-level supervision by a human operator. We have discussed some of the ideas underlying the proposed framework in previous writings about using simulations to accelerate the development of multiagent applications.[3] We also followed this approach in the development of a piracy countermeasure coordination system for the AgentC project, as well as in the process of porting collision-avoidance algorithms to UAVs in the AgentFly project.[4] For other work in the field, see the sidebar "Related Work on Mixed Reality."

Multilevel Mixed-Reality Testbeds

In general, there are different ways to approximate target deployment setup to make application tests faster and less expensive. Because our interest is in multi-entity systems involving several human actors and robotic assets, we consider two principal dimensions in which the approximation can proceed:

- the level of virtualization at which the target setup is represented, and
- the number of entities in the test scenarios.

We could introduce other approximation dimensions as long as they allow us to trade test costs for test accuracy.

Approximation Dimensions

The *virtualization level* denotes how much the target application setup is virtualized in a given testbed configuration—that is, how many parts are replaced with synthetic computational models. At one extreme, the full target setup uses no virtualized entities—only physical hardware platforms and human actors. Starting from the zero-virtualization setup l_0, we replace individual entities of the application with computational models. This process of gradual virtualization gives rise to a sequence of virtualization levels, labeled $l_0, l_1, \ldots, l_n$, in which l_n is a completely virtualized setup with fully synthetic computational representations of system entities. Between the two extremes lie mixed-reality testbeds, such as hardware-in-the-loop simulations or testbeds involving human actors in virtual reality. Within a single testbed configuration, we can assign different virtualization levels to different entities. In most cases, a lower virtualization level facilitates more reliable testing but consumes more time and resources. The core idea underlying our development process is to start at a relatively high system virtualization level and then iteratively decrease it until we reach the target deployment setup.

The other dimension along which we can approximate the target application setup 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, we can perform initial development and testing with only a subset of entities. We call the number of entities used the *size* of a testbed configuration.

Reducing the 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,

Related Work on Mixed Reality

Virtual reality deals with ways for a human user to observe and interact with nonexistent virtual worlds.[1] In mixed reality, the agent observes a world that is partly real and partly virtual. The concept of a *reality-virtuality continuum* embodies a continuous scale, ranging from fully real to fully virtual worlds.[2] Between the two extremes stands *mixed reality*, typically implemented either as *augmented reality* (in which perception of the real world is augmented with virtual objects) or *augmented virtuality* (the virtual world is augmented with elements of physical reality). Current research in mixed reality mostly concerns interface devices that let a human user observe and interact with mixed-reality worlds.

Developers of control programs for autonomous robotic systems routinely use sophisticated simulators to test their programs prior to deployment on target hardware. To further increase the fidelity of such testing, they can include one or more physical hardware assets in the simulation. Such simulations are, depending on the context, termed *hardware-in-the-loop simulations* (HILs), *hybrid simulations* (HS), or *mixed-reality simulations* (MRSs).[3–5]

The idea of evaluating control algorithms for multirobot systems in environments that mix both real and simulated entities is over 20 years old. An initial attempt dealing with simulation of industrial robots was published in 1989.[6] More recently, researchers have used mixed-reality simulations in the development of autonomous robotic assets. Ian Chen and his colleagues introduced a mixed-reality simulation library for the Gazebo 3D mobile robot simulator.[5] Using this library, a real hardware robot can interact with the Gazebo simulated world, which can both augment the real robot's environment with virtual objects and provide visual feedback on the state of the robot's perception through 3D visualization. Others have used hybrid and hardware-in-the-loop simulations for autonomous underwater vehicles.

References

1. G. Burdea and P. Coiffet, "Virtual Reality Technology," *Presence: Teleoperators and Virtual Environments*, vol. 12, no. 6, 2003, pp. 663–664.
2. P. Milgram et al., "Augmented Reality: A Class of Displays on the Reality-Virtuality Continuum," *Proc. Telemanipulator and Telepresence Technologies*, 1994, pp. 282–292.
3. D. Lane et al., "Interoperability and Synchronization of Distributed Hardware-in-the-Loop Simulation for Underwater Robot Development: Issues and Experiments," *Proc. Int'l Conf. Robotics and Automation* (ICRA 01), IEEE, 2001, pp. 909–914.
4. 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," *Proc. OCEANS 2007*, 2007, pp. 1–6.
5. I.Y.H. Chen, B. MacDonald, and B. Wünsche, "Mixed Reality Simulation for Mobile Robots," *Proc. Int'l Conf. Robotics and Automation* (ICRA 09), IEEE, 2009, pp. 232–237.
6. F. Cao and B. Shepherd, "Mimic: A Robot Planning Environment Integrating Real and Simulated Worlds," *Proc. Int'l Symp. Intelligent Control*, IEEE, 1989, pp. 459–464.

however, can undermine the ability to test collective behavior properties.

Testbed Configurations and Fidelity

To specify the distribution of virtualization levels in 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 is the number of entities modeled at the corresponding virtualization level. The testbed configuration's size (the number of entities used) then corresponds to $\eta(\sigma) = \sum_{i=0}^{n} s_i$. *Target configuration* σ_τ represents the entire target system, with no virtualization. Typically, σ_τ will be of the form $\sigma_\tau = (k, 0, \ldots, 0)$ for some $k \geq 0$.

We can visualize the space of all possible testbed configurations in a plane, as shown in Figure 2. The horizontal axis denotes the testbed configuration sizes; the vertical axis represents the aggregate virtualization level, defined as a weighted average of virtualization levels over all the entities in the given testbed configuration.

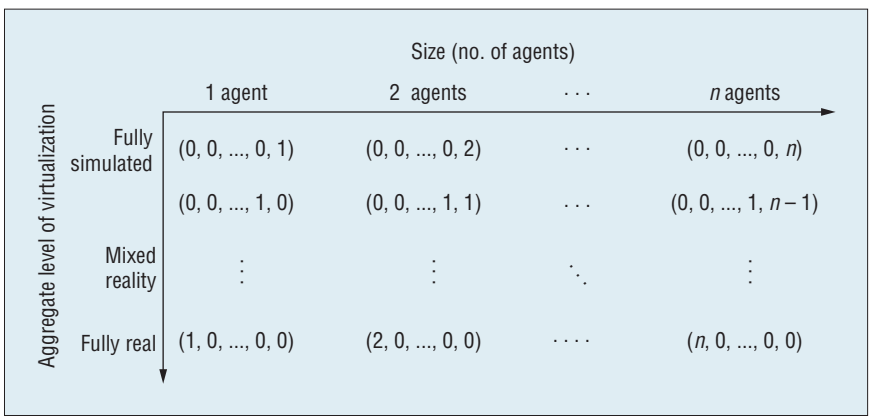

Figure 2. The space of testbed configurations with different combinations of testbed size and level of virtualization. The target configuration corresponds to the bottom right corner.

Unless the testbed corresponds to the target configuration σ_τ, the application's test performance can differ from its performance in the target configuration. To capture this difference, we introduce the concepts of *testbed error* and *testbed fidelity*. The testbed error $\varepsilon(\sigma) \in [0, \infty)$ is the distance between state space execution traces produced using the given testbed configuration and those using the target configuration, averaged over all possible application runs. The testbed

fidelity $\varphi(\sigma) = 1$ if $\varepsilon(\sigma) = 0$—that is, when a testbed configuration fully replicates the behavior of the target setup. Otherwise, $\varphi(\sigma) = \tanh[1/\varepsilon(\sigma)]$.

The higher the fidelity of a testbed, the more accurate the assessments obtained on it. We can use a different sigmoid function for the fidelity calculation, as long as it maps the testbed error in the (0, 1] interval.

Comparing Testbeds

The core of the proposed approach to HART application development is iterative evaluation in gradually more and more realistic setups. Expressed in terms of testbed configurations, the incremental process should use a sequence of testbed configurations with increasing fidelity. Unfortunately, in practice, determining such a sequence isn't directly feasible, because we can't determine testbed fidelity without performing tests on the full target configuration and comparing results.

Instead, we use an approximate fidelity ordering based on the following assumption, which is valid in most domains: a testbed configuration σ' is expected to have higher fidelity than σ if

- testbed size increases while the virtualization level does not increase, or
- the virtualization level decreases while the testbed size does not decrease.

More formally, we define the *approximate fidelity ordering relation* $\succ$ of testbed configurations as follows: for configurations $\sigma = (s_0, \ldots, s_n)$ and $\sigma'' = (s'_0, \ldots, s'_n)$, we denote $\sigma' \succ \sigma$ if at least one of the following holds:

- there exists a testbed configuration σ'' such that $\sigma' > \sigma'' > \sigma$ (transitivity);
- for all i, $s'_i \geq s_i$, and there exists j such that $s'_j > s_j$ and thus $\eta(\sigma) > \eta(\sigma)$; or
- $\eta(\sigma') = \eta(\sigma)$, and there exists j such that $s'_j > s_j$, and for all $i < j$, $s'_i \geq s_i$.

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

Virtualization Levels in HART Applications

For applications involving human-agent-robot teams, we can use specific virtualization techniques. For robotic hardware entities, these include the following, listed in decreasing order of level of virtualization (L):

- *Fully simulated hardware* (L_{FS}). A hardware asset is replaced by a fully synthetic computational model, such as an out-of-the-box robotic simulator like Gazebo, able to simulate the physical and electronic properties of several different robotic platforms.
- *Hardware-in-the-loop* (L_{HIL}). Even the most sophisticated robotic simulators don't capture all the phenomena that could arise in a real hardware asset. We can use a hardware-in-the-loop setup to increase the testbed's hardware fidelity. We test a hardware asset in a laboratory setting, with sensory input provided by a simulator, and actuator signals controlling a simulated physical model. This 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 or hybrid simulation* (L_{HS}). As a next step toward the full hardware testbed, we can use an augmented reality or hybrid simulation. Hardware assets operate in the target physical environment, but the simulator augments their sensory inputs with objects that exist only in the simulation. This approach is particularly useful in multiagent scenarios where we want to test interaction between multiple robots but fewer than the target number of physical robots are available.
- *Full hardware* (L_0). Robot assets are represented by target robotic platforms operating and interacting in the target physical environment.

Virtualization techniques involving human entities include the following, again listed in decreasing order of virtualization:

- *Computational behavior models* (L_{BM}). We substitute computational behavior models for human actors. We can construct the models manually according to an expert input, or they can be learned automatically from past observations of human behavior. We can employ different models of human decision-making, such as prospect theory, bounded rationality, or quantal-response equilibrium.
- *Data feeds about human behavior* (L_{DF}). The testbed replays data feeds of past real human behavior. This type of virtualization typically provides only a unidirectional link between the human actors and the rest of the application; in other words, the humans cannot react to the output of the application.
- *Virtual reality* (L_{VR}). Real humans are involved, but instead of operating directly in the target environment, they work in its virtual reality approximation. This lets us test aspects of human behavior that are hard to capture with computational models and facilitates testing of phenomena such as bounded rationality, irrational behavior, or the effect on decision-making of immediate mental attitudes, such as stress, fear, anger, or joy.

- *Simulated human-machine interaction* (L_{SI}). Real humans operate in the target environment, but some parts of their interaction with other entities remain simulated. This lets us test 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 act in the target environment.

Multilevel Incremental Development

The main reason for 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 so that each iteration can provide the maximum feedback on design and implementation choices.

Cost

To explain this idea more accurately, we introduce the notion of *iteration cost*, expressed as $cost(\sigma_1, \sigma_2)$, which represents the total cost for getting from an application that works correctly on testbed configuration σ_1 to an application that works correctly on configuration σ_2. In general, the overall iteration cost breaks down into.

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

Iteration Strategy

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

Unfortunately, except for trivial cases, determining the optimal strategy a priori is not feasible in real-world cases because it depends on the specifics of each application and on the uncertainty of cost estimates. However, assuming that iterations are faster on more highly virtualized testbeds, a reasonable strategy is to start with those, even though they might have lower fidelity. After that, as uncertainty about application design and underlying logic decreases, development should move toward higher-fidelity testbeds, even though they're likely to have higher test cost and therefore allow fewer tests. The following algorithm illustrates such an iteration strategy:

1. Choose an arbitrary starting configuration σ.
2. Develop a testbed with the configuration σ.
3. Develop, modify, and debug the application until it works correctly on σ.
4. If $\sigma = \sigma_\tau$, end.
5. If not, choose another configuration σ' such that $\sigma' \succ \sigma$, and start again from step 2 with σ'.

Typical scenarios involving several virtualization levels and multirobot systems will offer many ways to construct the sequence of testbed configurations. For instance, the developer can choose whether 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 around. Either way, the core of the iterative-development strategy should remain: to gradually refine the system setup so as to approach the target deployment scenario.

In general, the space of testbed configurations has as many dimensions as there are virtualizations levels, making the number of possible iteration strategies huge. A better understanding of the structure and properties of the error, fidelity ordering, and iteration cost functions is therefore essential for deriving intuitive or possibly even more formal rules for determining the iteration strategy.

Iterative Development of a Multi-UAV Tracking Application

We have used our approach to develop a coordination mechanism for a team of UAVs cooperatively tracking a number of humans in an urban environment. Besides a fully simulated setup (L_{FS}) and a fully physical target setup (L_0), we also considered an intermediate augmented-reality setup (L_{HS}). The intermediate setup was necessary because we only had two hardware platforms available—Unicorn UAVs from Procerus Technologies—whereas we had to test the coordination mechanism with larger teams of UAVs.

Starting from a fully simulated setup, 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 configurations corresponds to the approximate fidelity-ordering relation described earlier. The two development strategies shown correspond to the following two extreme cases.

Following the first strategy, we would first scale the coordination algorithms to the target number of simulated UAVs in a fully simulated

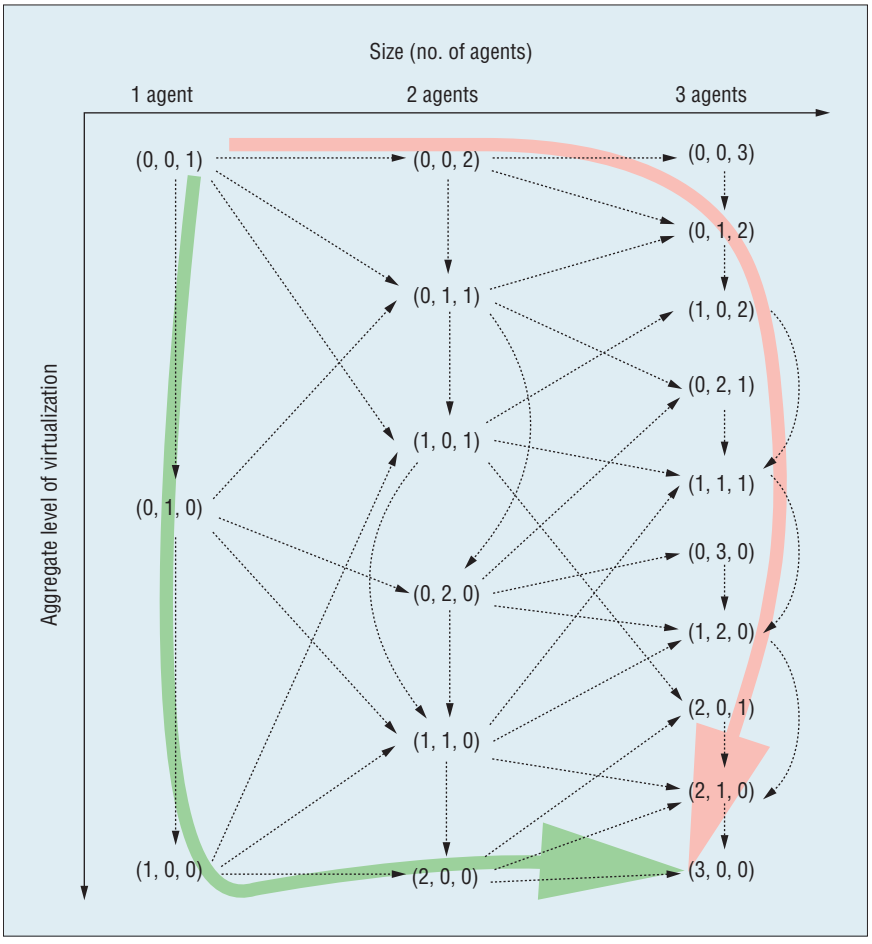

Figure 3. Space of testbed configurations for the multi-UAV tracking application. The arrows describe two possible development strategies.

environment. Then we would port the control code running on individual UAVs from the simulation to physical UAVs, one by one, embedded in an augmented-reality environment. At each step, we would go through the full port-extend or adapt-test-evaluate cycle until we reach the desired performance level. This way, development would eventually result in all aircraft control agents deployed and working correctly on target hardware UAV platforms with augmented sensory input feeds. In the final round of iterations, we would disconnect individual UAVs from the augmented-reality environment and place them in the target physical environment.

With the other strategy, we would initially work with only one UAV. Immediately after getting its control logic working correctly in the full simulation, we would port it to the UAV embedded in the augmented reality and then, after adapting it to work correctly, to a physical UAV. Once the system is working correctly on a single hardware UAV in the physical environment, we would add additional UAVs.

The choice between the two strategies typically depends on which is more uncertain: realizing collective team behavior or deploying the control logic on the target hardware platform. In this particular case, we chose the first approach.

Similarly, human entities that are the tracked objects can also be involved at different virtualization levels. Initially, we can represent all human subjects with computational behavior models. In the second step, we can approximate some human actors in the minimal virtual reality settings (for example, using a joystick with a GUI). The third step would involve real humans moving in the target physical environment but with simplified interaction with the robotic assets.

The approach described is just a first step toward a comprehensive methodology for incrementally developing HART applications using mixed-reality testbeds. 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 be found only for specific subcategories of HART applications, the proposed common conceptual framework allows for the comparison of different guidelines and the transfer of methodological knowledge across domains. ▪

Acknowledgments

This article originated during the sabbatical of Michal Pěchouček at the University of Southern California supported by the Fulbright Commission. Furthermore, the presented work was supported by US Army Communications-Electronics Research, Development and Engineering Center grants W911NF-10-1-0112 and W911NF-11-1-0252, Office of Naval Research grant N000140910537, and the Grant Agency of the Czech Technical University in Prague grant SGS10/189/OHK3/2T/13.

References

1. N. Koenig and A. Howard, "Design and Use Paradigms for Gazebo, an Open-Source Multi-robot Simulator," *Proc. Int'l. Conf. Intelligent Robots*

and Systems (IROS 04), IEEE, 2004, pp. 2149–2154.
2. J. Vokřínek, P. Novák, and A. Komenda, "Ground Tactical Mission Support by Multi-agent Control of UAV Operations," *Proc. 5th Int'l. Conf. Industrial Applications of Holonic and Multi-agent Systems* (HoloMAS 11), LNCS 6599, Springer, 2011, pp. 225–234.
3. M. Pěchouček, M. Jakob, and P. Novák, "Towards Simulation-Aided Design of Multi-agent Systems," *Proc. 8th Int'l Workshop Programming Multi-agent Systems* (PROMAS 10), LNCS 6599, Springer, 2012.
4. M. Jakob, O. Vaněk, and M. Pěchouček, "Using Agents to Improve International Maritime Transport Security," *IEEE Intelligent Systems*, vol. 26, no. 1, 2011, pp. 90–96.

THE AUTHORS

Michal Jakob is a senior researcher in the Agent Technology Center at the Czech Technical University in Prague. His research interests include agent-based simulation, computational game theory, and multiagent systems. Jakob has a PhD in artificial intelligence and biocybernetics from the Czech Technical University in Prague. Contact him at michal.jakob@agents.fel.cvut.cz.

Michal Pěchouček is a full professor in cybernetics and the head of the Agent Technology Center at the Czech Technical University in Prague, where he is also the deputy head of the Department of Computer Science and Engineering and head of the Agent Technology. His research interests include multiagent simulation and modeling; coordination; social-knowledge representation; multiagent planning; multiagent prototypes and testbeds; and applications of agent-based computing into security-related applications, unmanned-aerial-vehicle robotic coordination, and air traffic control. Pěchouček has a PhD in artificial intelligence and cybernetics from the Czech Technical University in Prague. Contact him at michal.pechoucek@agents.fel.cvut.cz.

Michal Čáp is a researcher and a PhD student in the Agent Technology Center at the Czech Technical University in Prague. His research interests include multiagent systems, especially agent-oriented programming, agent-based simulations, and distributed coordination and design of multirobot systems. Čáp has an MSc in agent technology from Utrecht University. Contact him at michal.cap@agents.fel.cvut.cz.

Peter Novák is a postdoctoral researcher in the Agent Technology Center at the Czech Technical University in Prague. His research interests include interaction of cognitive agents with dynamic environments, especially cognitive robotics, multiagent planning, and multiagent coordination. Novák has a PhD in computer science from the Clausthal University of Technology. Contact him at peter.novak@agents.fel.cvut.cz.

Ondřej Vaněk is a researcher and PhD student in the Agent Technology Center at the Czech Technical University in Prague. His research interests include multiagent simulations and application of cooperative and noncooperative game theory to securing complex critical infrastructures. Vaněk has an MSc in technical cybernetics from the Czech Technical University in Prague. Contact him at ondrej.vanek@agents.fel.cvut.cz.