Towards Simulation-Aided Design of Multi-Agent Systems

Michal Pěchouček, Michal Jakob and Peter Novák

Agent Technology Center
Dept. of Cybernetics, Faculty of Electrical Engineering,
Czech Technical University in Prague
Czech Republic

Abstract With the growing complexity of multi-agent applications and environments in which they are deployed, there is a need for development techniques that would allow for early testing and validation of application design and implementation. This is particularly true in cases where the developed multi-agent application is to be closely integrated with an existing, real-world system of multi-agent nature.
Drawing upon our previous experiences with development of complex multi-agent applications, we propose *simulation-aided design of multi-agent systems* (SADMAS), a methodology tightly integrating simulations of the target system into the MAS application development process. In its heart lies the use of *mixed-mode simulation*, a simulation where parts of the deployed application operate in the target environment and parts remain simulated. We argue, that employing SADMAS process contributes to reduction of risks involved in development of complex MAS applications, as well as it helps to accelerate the process. Besides describing the capstones of the SADMAS approach and consequences of its application, we also illustrate it's use on a case-study of a next-generation decentralised air traffic management system.

1 Introduction

In today's world, we are increasingly surrounded by and reliant on complex systems and infrastructures. Often, these systems behave far from the optimum or even highly undesirably. Roads in our cities are congested, plane trips frequently delayed, computer networks routinely overrun by worms and electricity grids fail in split-second cascade reactions. Our systems have become massively interwoven and interdependent, making both highly positive and negative chain reactions possible in critical systems. They have also grown increasingly decentralised, interconnected and autonomous, with more and more decisions originating at the level of individual subsystems rather than being strictly imposed top-down.

The paradigm of *multi-agent systems* is being increasingly successfully applied in modelling and engineering of complex distributed systems. Examples of

The presented work was supported by the Czech Republic Ministry of Education, Youth and Sports, grant no. MSM6840770038.

current and future applications of the paradigm include e.g., *civilian air traffic* with the requirement to double the capacity of the airspace within next ten years; *smart energy grids* automatically balancing energy production and consumption between interconnected yet independent producers and consumers; *disaster and emergency management operations*, which in the future will rely on the coordination of heterogeneous ad-hoc teams of semi-autonomous robotic entities and networks of unattended sensors; or *intelligent buildings* comprised of large numbers of interconnected autonomous sensors and actuators adapting to the activities of their human occupants.

Development and deployment of such *complex multi-agent systems* is a challenging task. Large numbers of spatially distributed active entities characterised by complex patterns of mutual interaction and feedback links give rise to dynamic, non-linear emergent behaviours which are very difficult to understand, capture and, most importantly, control. We argue that because of the complexity of the above-described types of applications, it is no longer possible to develop such systems in a linear, top-down fashion, starting from a set of requirements and proceeding to a fully developed solution. Instead, more evolutionary, iterative methodologies are needed to successfully approach the problem of development of complex multi-agent systems.

In this paper, we give a preliminary outline of the *simulation-aided design of multi-agent systems (SADMAS)* approach, a development methodology relying in its core on the exploitation of a series of gradually refined and accurate simulations for testing and evaluation of intermediary development versions of the engineered application. In particular, we propose and argue in favour of using *mixed-mode simulations* in which the implemented application is evaluated against a partly simulated environment. I.e., some aspects of the test environment are real parts of the target system while some remain simulated. Over time, the extent of the simulation will be decreasing until the application fully interacts with the target system itself. We argue that this approach helps to accelerate the development of complex multi-agent applications, while at the same time keeps risks and costs associated with destruction or loss of the tested assets low. Our goal is not to give the ultimate answer to the problem of developing complex systems, but rather to synthesise our past experiences with building such systems and to initiate a discussion on the role simulations can play in making engineering of such systems more efficient.

In the following section, we introduce the conceptual framework, the processes of the SADMAS approach, and discuss the scope of its applicability. Subsequently, Section 3 distills our past experiences with the early version of the SADMAS approach. We put forward a set of methodological principles to be respected during application development, in order to facilitate successful application of the SADMAS methodology. Finally, in Section 4, we discuss tool support for the SADMAS approach, in particular the core features an ideal simulation platform facilitating the introduced methodological principles should provide. Finally, sections 5 and 6 conclude the paper by a discussion of related work and some final remarks.

Throughout the paper, the discourse is accompanied by a running example exemplifying the main principles of the SADMAS approach in a case-study application in the air-traffic management domain developed in our research centre. Before diving into the core of the paper, let us first introduce the case-study problem.

Running case study: free-flight-oriented air traffic control

The ever-growing volume of air traffic is approaching the stage when the current techniques for its management begin to constrain its further expansion. The main limiting factor is congestion of the predefined, reserved flight corridors used by air traffic controllers for long-distance routing of flights in the airspace. Additional grand challenge stems from the need to integrate autonomous unmanned aerial assets with manned air traffic. Small unmanned aerial vehicles (UAVs) are often used for tasks such as policing and emergency area surveillance, and need to be able to operate near airports with heavy civilian air traffic. Current air traffic management systems cannot efficiently support integration of such UAVs and at the same time handle future higher traffic densities. Sophisticated, intelligent technology is needed to enable further growth of the global, manned and unmanned air traffic.

A promising solution concept is represented by the *free-flight*-oriented approach which suggests moving away from the current centralised traffic control based on the predefined flight corridors towards decentralisation. In the extreme, the free-flight air traffic control should be based on on-line negotiation schemes and consequently moved on board of the (un-)manned aircrafts. This radical shift is expected to provide a more efficient use of the available airspace and improve support for dynamic flight trajectory re-planning, as well as collision avoidance. Autonomous decision making in such scenarios is especially important in the case of dynamic, partially unmanned operations in police and military settings.

As the running example for the following discourse, we will use *AgentFly* application, an agent-based free-flight-oriented air traffic management and control system developed in Agent Technology Centre of the Czech Technical University in Prague [17,13]. The aim of the project is to i) propose and implement decentralised flight control algorithms, subsequently ii) evaluate them in an experimental planning system for civilian air traffic control simulating the real traffic in the US National Airspace (NAS), and finally also iii) port and evaluate the proposed control algorithms on board of real aircrafts of different types (fixed-wing aeroplanes, as well helicopters).

2 Simulation-aided design of multi-agent systems

At the core of the SADMAS approach lies use of multi-agent simulations for iterative application evaluation. Results of the evaluations subsequently serve as a driver for further advancement of the process. In the following, we introduce

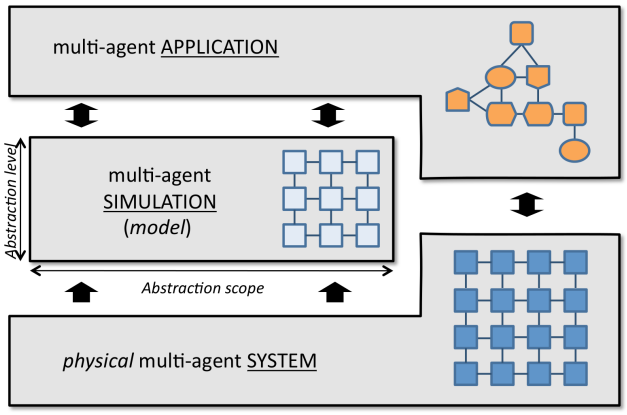

Figure 1. Conceptual scheme of the SADMAS approach.

the core concepts of the approach, sketch the application development process it induces and finally elaborate on conditions for its applicability.

2.1 Core concepts

SADMAS approach revolves around the following three multi-agent concepts:

target system: a real-world system which should be controlled by means of the developed *multi-agent application;*

multi-agent simulation: an agent-based simulation of the target system. In general, the simulation can have different (a) *level of abstraction* (how much is the target system simplified) and (b) *scope of abstraction* (which parts of the target system are simplified);

multi-agent application: a decentralised, multi-agent software system designed to control some (or all) aspects of the target system.

2.2 Development process

One of the main problems of using simulations as an intermediate tool for testing and evaluation of implemented application, which is nothing particularly novel *per se*, is maintenance of relevance of the simulation with the target system. In result, it might easily happen that while the implemented application runs smoothly in the simulated environment, it breaks down upon its deployment in the target system. We argue, that one of the main reasons for such failures is breaking of *relevance* of the simulation to the target real-world system. As an example of such a failure, consider a situation when in the simulation of an air traffic management system developers assume perfect communication links used in the multi-agent negotiation. Even though the assumption is not unreasonable

as such, the real reliability of communication links could be e.g., 95%, the actual deployment of the application validated against such a simulation could result in severe problems late in the deployment stages of the project. Since it is often extremely difficult, if not plainly impossible, to detect all such misalignments right at the beginning of the project, the development process should ensure that they are discovered as soon as possible in early stages of the development process. The SADMAS approach is based on incremental development of the application hand-in-hand with a series of ever more accurate simulations it is evaluated against. On the top level, the overall process of developing a multi-agent application using the SADMAS approach consists of the following steps:

1. Collect the (multi-agent) *application* requirements, assuming it will control, or interact with, the physical (multi-agent) *system*.
2. Repeat the following steps iteratively:
   (a) Choose the appropriate *simulation level of abstraction* and *application feature coverage* for the particular process iteration. The choice of the appropriate level of abstraction of the application iteration directly results in determining which parts of the simulation will be replaced with interfaces to the real-world *target system*.
   (b) Building upon the simulation from the previous iteration of the process, construct a refined multi-agent *mixed-mode simulation* (cf. Subsection 4.4) of the target system. The simulation should focus on the critical features of the iteration, such as the nature of inter-agent interactions and interaction with the environment so that it respects the chosen level of abstraction of the process iteration.
   (c) Based on the requirements collected in (1) design and develop the *application* (multi-agent control mechanism) w.r.t. the chosen application feature coverage.
   (d) Test, debug and evaluate the *application* (or application variants) developed in (2c) on the multi-agent simulation constructed in the step (2b).
   (e) Iteratively repeat the steps (2c) and (2d) until sufficient level of application reliability w.r.t. the chosen set of features on the chosen level of abstraction is reached.
3. Once the simulation is either completely replaced with the physical target system, or sufficiently tested, perform final evaluation and verification of the application directly interacting with the target system and *deploy* the application.

From some point on in the development process, the simulation refinement should result in replacement of some aspects of the simulation with direct interfaces to the target system. I.e., with the advancing stage of the application development, it is increasingly tested against relevant aspects the real-world target system. The consecutive replacement of aspects of the simulation with interfaces to the physical target system leads to subsequent refinements and adaptations of the mixed-mode simulations eventually resulting in complete replacement of

the initial simulation of the target system with the system itself. By keeping the tight alignment between the intermediate states of the system simulation and the application, the process ensures, resp. maintains relevance of the developed application w.r.t. the target system and the chosen level of abstraction at the process iteration.

In many cases, the implementation of the step (2c) will not lead to a straightforward process. Rather, in order to find the proper set of partial solutions to cover the target feature set at the appropriate level of abstraction, the developer will be often forced to implement several versions of the application. Only after performing the evaluation step (2d), on the basis of the collected experimental results the implementer is able to decide which solution version will be carried to the next iteration of the process. Thus, the proposed approach is not strictly linear, such as e.g., the waterfall model inspired processes.

Figure 2 visualises the iterative process described above and highlights the role of mixed-mode simulation in the overall architecture. In order to ensure that the application can be ported seamlessly through the series of ever more accurate simulations without significant additional expenses, the design of the multi-agent application has to meet several requirements which we expand upon later in Section 3.

2.3 Scope of applicability

In order for the SADMAS approach to be applicable to a particular problem domain, it is critical to analyse the relationship of the target system vs. the requirements on the developed application. In particular, the target system in which the application will be deployed has to be of multi-agent nature and should manifest some kind of emergent, collective behaviour on its own. Furthermore, we assume that the developed application will be deployed in hardware and situated in the target system. Due to its reliance on safe simulation-based evaluation, the SADMAS approach is particularly suitable for applications in which at least one of the following conditions hold

- the cost of an individual HW unit is high, and risk of a failure that may result in loosing an asset is not negligible;
- application testing may result in undesirable, possibly harmful changes of the environment, such as when the safety of material resources and/or humans would be endangered;
- the cost of running the HW experiments is high and proportional to the number of deployed assets; or finally
- the application operates in an environment which strongly influences the behaviour of the application, i.e., it is difficult and/or costly to set up test conditions so that all critical aspects of the application can be evaluated.

Example problem domains, target systems, potentially suitable for application of the SADMAS methodology include air traffic, public transport system, energy grids including consumers and producers, or peer-to-peer file sharing network,

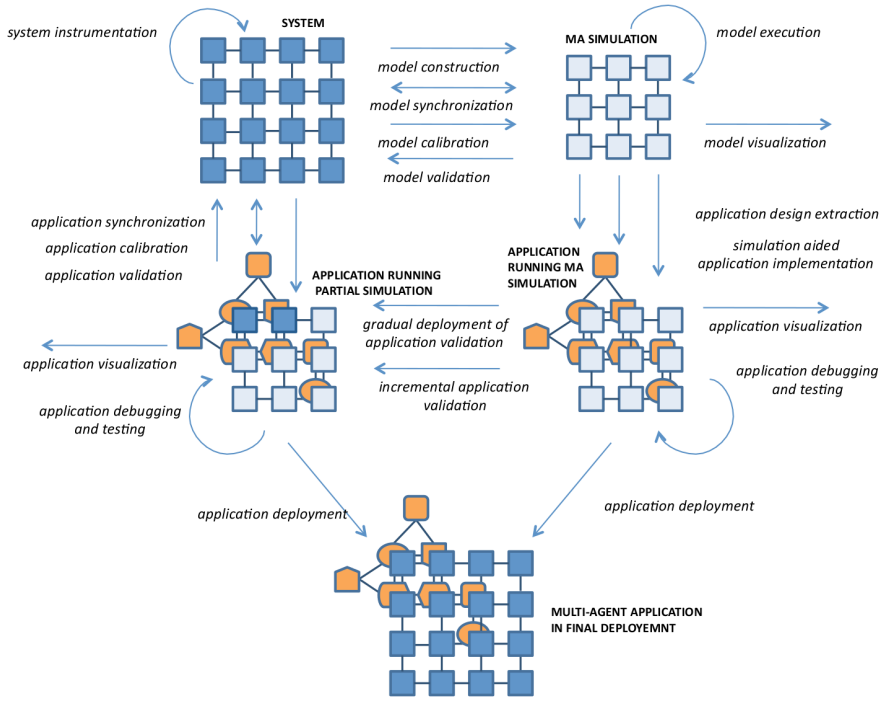

Figure 2. Relation and processes between the components components of the SADMAS approach.

etc. The possible developed applications in such domains include automated aircraft, resp. vehicle flight planning and collision avoidance mechanisms, control and management mechanisms for negotiating electricity consumption and production between entities on the smart grid.

The scope of simulations used as the intermediary testing platform would typically differ between different stages of the development process. The development may initially be done completely in an isolated simulation, while later its parts would be replaced by the real-world system and eventually, in the ultimate deployment setup, the application would solely interact directly with the target system.

AgentFly case-study analysis: Let us analyse the *AgentFly* case-study using the core concepts of the SADMAS approach. The *target system* is clearly the NAS air traffic system and the related infrastructure. Let's simplify the problem and consider only unmanned aeroplanes. These assets usually operate in geographically bounded environments containing various special-use air-zones, termed *no-fly zones*. The operation of an aircraft is determined by i) the take-off location, ii) the landing location and iii) set of time-space geographical waypoints, e.g.,

specifying a surveillance pattern. The multi-agent *application* comprises the set of autopilot control algorithms on board of each asset. The application should control the movements of a number of aircrafts along collision-free trajectories that pass through the specified geographical waypoints, while avoiding the no-fly zones. The functionality of the application can be decomposed into two layers:

- *individual planning layer* aiming at planning a smooth 4-dimensional (space and time) trajectory passing through the waypoints for the aircraft; and
- *multi-agent planning layer* aiming at collision avoidance of a set of aircrafts, i.e., detecting potential future collision and using peer-to-peer negotiation in order to determine the set of deconfliction manoeuvres.

The application further needs to fulfil a number of non-functional requirements, in particular i) near-real-time responsiveness, ii) scalability to a very high number of assets and iii) reliability. These requirements make it impossible to design the application in isolation from the target system.

3 Design considerations of the SADMAS approach

The SADMAS application development process leads to several issues, which should be considered and tackled early on in the development process. Below, we introduce some of the most important ones. Concretely, we discuss the bottom-up system evolution, an important stance to be adopted in the development process. We continue with stressing the need to support elaboration tolerance both of the application design, as well as the mixed-mode simulation used for evaluation of the application iterations. Finally, we conclude with a set of issues ensuring cross-platform portability of the application w.r.t. the target deployment infrastructure.

3.1 Bottom-up system evolution

Ideally, the SADMAS process will proceed from testing of the first application prototype against a full environment simulation, through replacements of the rudimentary and simpler interfaces to the physical target system, eventually to replacement of the simulation with the target system itself. In consequence, the design of both the application and the simulation cannot be based on a blueprint resulting from a green-field-style top-down analytical procedure. The discussion of the scope of applicability of the SADMAS process (Subsection 2.3) implies that typically, the nature of the implemented application will be such that it is being constructed *into* an existing target system, rather than being developed from scratch including its environment. In result, the overall behaviour of the application, together with the target system cannot be characterised in separation and must be considered as a whole. As an example, consider the running example, the *AgentFly* air traffic management system, which will be typically seamlessly deployed into the already set up and strictly regulated airspace infrastructure. From the point of view of a single agent within the system, an unmanned aircraft,

the behaviour of the other aircrafts is a part of the environment behaviour. However, these aircrafts are in fact components of the implemented application on par with the agent in consideration. The iterative SADMAS process is based on tight coupling between the application and simulation development and stresses growing accuracy of the simulation w.r.t. the target system. I.e., it supports the bottom-up approach to maintaining the relevance of the simulation to the deployment environment by gradual replacement of ever more significant parts of the simulation with the real-world APIs. Thus it provides a solid support for the evolutionary application development principle.

In AgentFly: The planning and collision avoidance application in *AgentFly* has been designed in an evolutionary manner respecting the bottom-up approach to interaction with the environment. The principle objective was to delegate the aircraft planning autonomy on board of the UAVs and thus minimise the role of the centralised point of control. Several variants of the negotiation mechanism used for collision avoidance have been developed (including the pair-wise and the multi-party negotiation schemes) and evaluated on a high-fidelity simulation of UAV flight and sensor behaviour. Experiments on a range of synthetic, as well as real-world inspired scenarios provided empirical evidence that the simpler and more robust pair-wise collision avoidance algorithms are sufficient even in the most extreme cases.

3.2 Elaboration Tolerance

As a consequence of the SADMAS process, at any specific development iteration, the application and simulation releases must be frozen and ideally further iterations should not require change reverse, or backward modifications of the features implemented in previously frozen releases. The strong emphasis of the process on evolutionary development implies that already the initial application design and layout provides sufficient flexibility w.r.t. future changes and adaptations. I.e., more than with other application development methodologies, SADMAS approach calls for strong emphasis on *elaboration tolerance* of both, the design, as well as the actual implementations of the application and the simulation. Paraphrasing the origin of the term in [12], a design is elaboration tolerant to the extent that it is convenient to modify it to take into account new phenomena or changed circumstances. In particular, transposed to multi-agent application development domain, this leads to a requirement that the design should be based on component-based practises with well-thought-of set of APIs, that, ideally, do not change during the development process.

While nowadays the call for flexibility of software design is relatively obvious and straightforward (mostly due to maintenance reasons), often it is difficult to ensure the flexibility w.r.t. right kind of future modifications. Without going deeper into this issue, we see a potential for methodological support assisting multi-agent systems programmers and designers to understand the elaboration tolerance implications on their designs in order to assist them to make the right and informed decisions in the early stages of the application development process.

In AgentFly: The planning algorithms developed in *AgentFly* project [18] have been designed so that they perform general, yet extremely efficient on-the-fly planning in complex 4D space (spatio-temporal planning). While this design decision did not break elaboration tolerance and kept options open w.r.t. the potential uses of the algorithms in various types of aircrafts, at certain stage we realised that the data structures used in planning algorithms were too closely linked with the implemented planning algorithms tailored for Cartesian 3D coordinates. At the point, when the systems needed to scale up to the level of US National Airspace, there was a requirement to upgrade the planning algorithms from Cartesian coordinates to GPS coordinate system, which however turned out to be a major issue. Due to the efficiency requirements, elaboration tolerance have not been met and was not foreseen in this aspect of code development. This design issue finally resulted in major implementation difficulties as large portions of the already constructed application had to be modified accordingly.

3.3 Cross-platform portability and deployment issues

As a consequence of the gradual shift of the series of mixed-mode simulations towards interfacing with the real-world target system, the SADMAS process also dictates gradual transfer of the application from synthetic settings to deployment on the target platform. In particular, e.g., in the air-traffic management domain this means gradual porting of the core algorithms for collision avoidance from synthetic personal computer environment to the actual hardware platform where it will be finally deployed, i.e., the embedded computers running on board of the target aircrafts. In order to ensure that such a gradual deployment of the application to the real-world system is possible, several additional design and implementation considerations should be respected.

Technically, the elaboration tolerant application development must be supported from early stages on. The decisions include appropriate choices of programming platforms, tools and supporting infrastructure so as to aim at maximum cross-platform portability of the application, as well as the relevant aspects of the mixed-mode simulation.

As far as the application design is concerned, we argue that in the multi-agent systems context, the initial application design should strive for maximum decentralisation of decision making of the application components, the agents. I.e., the multi-agent application logic has to be implemented so that it respects the actual constraints and properties of the target system, such as bandwidth, latency, etc.

In order to comply with general real-time properties of many real-world application domains, full asynchronicity of the application components, the agents, should be striven for whenever possible. Even though in general it is non-trivial and even relatively difficult to implement and especially debug asynchronous systems, we argue, these difficulties must be overcome in order to be able to routinely propose and implement elaboration tolerant multi-agent application designs. We see a great potential in methodological guidelines and design pat-

terns aiming at assisting developers to work out fully asynchronous multi-agent systems.

Finally, the design of the mixed-mode simulation must enable and facilitate the iterative gradual replacement of individual modules with interfaces for interaction with the real-world system. It should ensure from the project beginning on that the individual interfaces to the environment are clearly separated and defined in a general way so that they closely correspond to the respective aspects of the target real-world system. This clear separation and modularisation will later in the process enable the gradual and piece-wise replacement of the API's to simulated modules with interfaces to the actual sensors and actuators of the application to the target system.

Again, while each of the above considerations belongs to standard best practises of engineering of complex software systems, often they are not respected, or difficult to consider in development utilising simulations. It is important to consider in the earliest stages of application development the relevance of not only the application design w.r.t. the target system, but also of the design of the multi-agent simulation w.r.t. the target system. It is thus vital to keep a strong separation between the design of the application w.r.t. the design and implementation of the simulation itself. Only this way the developer can avoid the situation when the implemented application works flawlessly when evaluated against a complex simulation, however breaks down when finally deployed to the target system. Often the primary problem lies not in the design of the application itself, but in the level of abstraction assumed by the design of the simulation which simplified some crucial aspect of the target system too far from the real-world conditions.

In AgentFly: The main constraint in the design of the autonomous collision avoidance system is the bandwidth and reliability of aircraft-to-ground communication links, which prevents deployment of a centralised solution. The collision avoidance mechanism was therefore designed in a fully decentralised manner. Although each negotiation is managed by a master plane, this plane is chosen dynamically from within the collision pair and/or group. From the beginning, the mechanisms was designed as fully asynchronous and its computational and memory requirements were kept in line with the parameters of aeroplane's onboard computers. Altogether, these made the process of migrating the mechanism from simulated aircrafts to real hardware UAV platforms relatively straightforward.

4 Requirements on a SADMAS platform

One of the key components of an ideal pragmatic SADMAS toolkit is a platform for construction, calibration and execution of the series of mixed-mode simulations used for evaluation of the iterations of the developed application. In the following, we discuss some of the properties such a platform should feature in order to facilitate the SADMAS process.

4.1 Adjustable simulation fidelity

The simulation has to accurately and reliably replicate those aspects of the target system that are critical for the real-world operation of the developed application. Key aspects that need to be modelled include computation and communication capabilities of the target system, such as throughput limitations, communication delays, and communication link failures. In the case of multi-robot applications, interaction with the environment, i.e., sensors, actuators and physics of the component on which the application is to be deployed, need to be also modelled with sufficient fidelity, as it is done in robotic simulators (e.g., [11]).

High-fidelity simulations require high amount of resources, both at run-time but also at the design time when a sufficiently detailed model of the target system has to be designed, implemented and properly calibrated. The level of abstraction chosen for the simulation should be balanced with i) the ability to scale to the required size of the target systems and ii) obtain enough data about the behaviour of the target system to facilitate precise calibration of the simulation model. Depending on the focus of the application, the level of detail required may vary for different aspects of the target system and/or different parts of the simulation. Multi-scale simulation techniques [10] can be employed for this purpose.

In AgentFly: The lesson learnt from *AgentFly* development was that the granularity of the simulation does not need to be defined *a priori* and that it is rather important to decide upon the right level of fidelity for specific scenarios. Low simulation granularity does not make it believable and would not facilitate the application migration from the simulation environment to the final system. On the other hand way too high fidelity may become a resource overkill (both financially, as well as w.r.t. human resources involved). For the original *AgentFly* simulation, the application was modelled as a collective of micro UAVs flying in perfect conditions. At that stage, the development team was putting its focus into the fidelity of the physical dynamic model of the flying asset. In later stages, during *AgentFly* extension to support air traffic management in the US National Airspace, the fidelity of the simulation had to be refined towards providing i) a high precision aeroplane model based on the *Total Energy Model* of the BADA (Base of Aircraft DAta) aeroplane performance standards (cf. [7]), ii) full models of the geographical environment including landing, take-off locations, no-fly zones and special purpose air traffic sectors; and finally iii) full models of the weather. The design of the home-developed simulation platform *A-globe Simulation* [15] turned out to be flexible enough to accommodate the corresponding adjustments of the simulation without major frictions.

4.2 Rich Environmental Modelling

The problem domains SADMAS approach is suitable for usually concern agents situated in and interacting with a real-world physical environment. An ideal

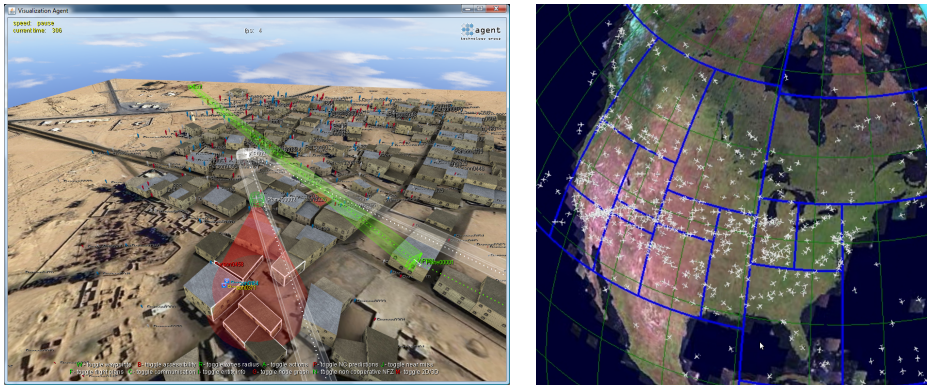

Figure 3. Left: rich environment model in *Tactical-AgentFly* simulation. Right: dynamic partitioning of the environment in *AgentFly* simulation.

SADMAS simulation platform has to provide abstractions and run-time support for simulating rich virtual environments. Depending on the type of application, different modes of interaction with environment have to be accommodated, including perception (via sensors) and acting (via actuators) in the environment, possibly also agent mobility within the environment. Possibly, some applications may require modelling of the environment dynamics itself (e.g., weather), and this might also need to be simulated with sufficient precision.

Rich environmental models are supported by robotic simulation platforms and engines for simulating digital virtual worlds. However, most existing popular multi-agent-based simulation platforms (e.g., [5,20]), so far only focus at highly abstracted environments (graphs, grids), though recently support for GIS concepts has been added. Structurally complex environments involving object such as buildings are still generally unsupported in general-purpose multi-agent-based simulation platforms.

In AgentFly: Important part of air traffic simulation is a detailed modelling of the landscape, the ground terrain and also weather, in particular wind. In the simulation used for testing collaborative UAV-based surveillance control mechanisms [13], we had to model urban terrain, sensors and both the physical, as well as logical movement of ground units. Figure 3 provides a snapshot of the simulation visualisation involving the urban environment with a number of ground agents and two UAVs collaboratively performing surveillance and tracking tasks over the area.

4.3 Simulation scalability

In many application domains, with the advancing SADMAS process iteration, the size of the simulated target (sub-)system will significantly grow. To accommodate to the variable simulation size, an ideal SADMAS simulation platform

design should emphasise scalability. Both, in terms of growing number of entities and components of the multi-agent application, as well as the increasing size and accuracy of the simulated environment. To support the scalability on the simulation model level, the platform should therefore also support scalability w.r.t. the growing number computational resources.

The multi-agent-based approach to simulation provides natural decomposition of the computation process. However, there are two main factors that may prevent scalability of the simulation. Firstly, it is the super-linear growth in communication requirements between agents with their growing number, and secondly, naive employment of the centralised environment simulation design. The first factor is application-specific and closely linked to the way individual agents in the target system interact with each other during their operation; there are no universal techniques by which the problem can be generally addressed. The bottleneck represented by the centralised environment simulation design could be eliminated by distribution of fragments of the simulation, i.e., partitioning the environment into a number of zones which are each hosted on a separate computational core. Distributed agent-based simulations are a relatively new concept and platform support for them is relatively limited. We can distinguish between two types of distribution: i) agents only and ii) agents and environment distribution. The latter is more complex because it requires partitioning the environment state and correctly synchronising it across multiple machines.

In AgentFly: As a part of the collaboration with the FAA (US Federal Aviation Administration), it was requested to model the entire US National Air Space traffic comprising of approximately 75,000 flights a day. Due to high-fidelity of the simulation required (cf. the above subsections), simulating such a number of flights turned out to be impossible on a single machine. A fully distributed simulation was therefore developed, where both the application logic and simulation of the environment was distributed. In the latter case, this was achieved by dynamically partitioning the environment into a number zones. The Figure 3 depicts the dynamic fragmentation of the environment where each zone was dedicated to a single host computer. With the growing number of simulated aircrafts in the airspace, the environment was re-fragmenting and re-distributing in order to ensure load-balancing of the entire distributed simulation among the hosts of the computational cluster running it.

4.4 Mixed-mode support

Mixed-mode simulation, the core element of the SADMAS approach, denotes a simulation with a capability to replace part of the simulation by the respective physical component whose state, its sensory inputs and actuator outputs, are reflected back into the simulation using a phantom simulated entity. Mixed-mode simulation enables a series of intermediate steps in the validation of the developed application, between a fully simulation-based evaluation and evaluation on a fully deployed application. In a mixed-mode simulation, parts of the application logic

Figure 4. *AgentFly* UAV hardware platforms. Left, Procerus fixed-wing aircraft with the development toolkit; Right, *LinkQuad* vertical take-off and landing quad-rotor aircraft.

can be tested on a real hardware platform in situations which involve multiple entities (e.g., an autonomous car driving through a congested city), without the need to have all the entities physically available. The latter could be either too costly, or potentially dangerous either for the assets involved, or the deployed environment itself.

Mixed-mode simulation might be the only way to evaluate the developed application (prior to a full-scale deployment) in scenarios where the target system is beyond our control and cannot be easily used for evaluation, such as e.g., in urban traffic or human crowds. A fundamental requirement that mixed-mode execution thus imposes on the simulation platform is the ability of the simulation to run in, at least, real-time (i.e., one wall-clock second corresponds to one simulation second). The agent-based tools and platforms would need to support various levels of state synchronisation so that an easy plug-and-play, resp. replace-and-forget functionality is provided.

In AgentFly: AgentFly has been successfully tested on physical hardware platform, *Unicorn* UAV test platform by Procerus Technologies [19] and we are currently working on porting the relevant application fragments to the Linksys *LinkQuad* vertical take-off and landing aircraft (cf. Figure 4). We already performed a series of successful experiments with *AgentFly* planning algorithms running on the platforms and we are currently adapting the *AgentFly* simulation platform to be able to accommodate a mixed-mode simulation with several physical flying assets (2 Unicorn UAVs and 1 LinkQuad aircrafts)

The second *AgentFly* mixed-mode line of evaluation is w.r.t. the large-scale air-traffic management systems of the US National Airspace. There we are working towards mixed-mode mixed-mode simulations of the domain populated with a number of computational models of the controllers, with several controllers instantiated with real human air traffic controllers provided by FAA.

4.5 Evaluation modes

The SADMAS process is in its heart evaluation-driven. I.e., it dictates advancement to the next development process iteration only when the actual release performs sufficiently well against the current choice of implemented features and level of abstraction embodied in the iteration of the mixed-mode simulation. Different modes in which the simulation is used result in several different simulation execution modes which should be supported by an ideal SADMAS simulation platform:

single instance mode: the goal is to execute a single simulation run in a minimum time. This requirement arises whenever the simulation is used during interactive development and testing of the application. In this case, minimisation of simulation time improves the productivity of the programmer.

batch mode: the goal is to execute a batch of simulation runs as fast as possible. This need arises when performing extensive simulation-based evaluations which involve simulating the system in different configurations.

real-time mode: the goal is to execute the simulation in real-time. This final requirement arises when performing mixed-mode simulation against the real-world target system.

If a single host is incapable of running the simulation sufficiently fast, the only way to perform the single instance and real-time modes is by distributing the simulation over multiple machines. We discuss the system distribution above, in Subsection 4.3.

As long as a single simulation instance can fit into a memory of a single machine, the batch-mode evaluation is often best performed by *not* distributing the simulation run, but fitting as many, possibly smaller-scale, simulation runs on a single host. Our experience shows, that such a setup generally leads to lower overheads, while we were still able to retrieve reasonable experimental data from the batch-mode simulations; though this might not hold universally.

In AgentFly: The actual development and debugging of *AgentFly* relied on the single-instance execution mode. The need for batch-mode evaluation arose and became prominent in testing efficiency of the developed collision avoidance methods. The capability of the multi-agent application has been tested on a series of super-conflict scenarios in which a number of aircrafts are set to collide in a single space point. In these scenarios, various aspects of the application performance were tested, such as i) total length of the flight trajectory, ii) flight safety expressed in the number of near misses, iii) bandwidth requirement or iv) total computational time. For such experimental setups involving real hardware assets, the possibility of employing mixed-mode simulations is crucial in order to minimise the potential risks of the project.

5 Discussion and related work

Some of the issues raised in the Section 3 on application design considerations have been addressed in the field of agent-based software engineering. Example

methodologies include *Adelfe* [3], a methodology for the design of adaptive software situated in unpredictable environments. *Gaia* [21], general methodology that supports both the micro-level, agent structure, as well as the macro-level, agent society and organisational aspects of agent development. *Prometheus* [14] methodology provides hierarchical structuring mechanisms which allow design to be performed at multiple levels of abstraction. Such mechanisms are crucial to the pragmatics of large-scale complex systems development. *Tropos* [4] methodology puts major focus on agent-oriented requirements engineering perspective.

Though some of the agent-oriented software engineering methodologies provide support for evolutionary application development, they assume the testing is done directly against the target environment. They do not explicitly support simulation-based development cycle in which the testing of the developed multi-agent application is first done on simulations of the target system. We argue, that the here introduced methodology of *Simulation-Aided Design of Multi-Agent Systems* is orthogonal to the existing methodological approaches and can be relatively easily combined with them. While traditional multi-agent systems development methodologies are to be applied to the overall application design, the SADMAS approach rather guides developers through the intermediary development life-cycle of the project and ensures that 1) the application is continuously evaluated against relevant and sufficiently accurate simulated environments, and as a consequence of this, 2) attention is paid to the relevance of the application w.r.t. relevant aspects of the environment continuously throughout the application life-cycle. Here we speak about both knowingly crucial issues and aspects of the target system, as well as those which are potentially crucial, however not easily recognised as such in the early phases of application development.

Although, to our knowledge, no general-purpose simulation-oriented agent-based software-engineering methodologies exist, the SADMAS idea has been partially applied in specific sub-domains. A good example is the *Anthill* framework [1] for design, implementation and evaluation of peer-to-peer applications based on ideas such as multi-agent and evolutionary programming. Besides the air traffic domain referenced throughout the paper, some of the core ideas of the SADMAS approach have also been employed in the general traffic and transportation domains, e.g., for designing agent-based demand-responsive transport coordination mechanism [8], for increasing maritime security [9] and for designing multi-agent control of smart grids [16]. In none of the above cases, however, the mixed-mode simulation technique has been used.

Regarding the relationship to general-purpose software engineering methodologies, to an extent, the SADMAS process is similar to Test-Driven Development [2], although SADMAS comes with a significantly more complex test evaluation mechanism employing a simulations. To an extent, the stress of the SADMAS approach on frequent evaluation against more and more accurate simulation models of the target system integrates elements of *continuous integration* [6] approach.

Let us finish with a remark on the cost-effectiveness of the SADMAS approach. At the first sight, the implementation of mixed-mode simulations in-

creases the overall cost of development due to introduction of an additional development step which brings in rather non-trivial costs. The additional costs of mixed-mode simulation can be hardly expressed *a priori* and are very application domain specific. However, we argue that in the cases when the risks involved throughout the application development life-cycle are relatively high, the SADMAS approach can significantly reduce the overall project risks, and safe costs in the face of severe costs in the case of project failure due to either loss of physical assets, or harm done to the target system possibly involving humans (cf. also Section 2.3).

6 Conclusions and outlook

With the increasing size and complexity of environments and target systems with which multi-agent applications have to interact, there is a growing need to support incremental, evolutionary development processes. In particular, in the context of engineering of large-scale complex and non-linear systems, where all the consequences of individual design decisions are hard to predict *a priori*, there is a need to rapidly evaluate the individual design and implementation decisions without requiring full-scale deployment to the target real-world environment. Such support can be provided by having a multi-agent-based simulation of the target environment available and using it as a testbed during the development of the application. With the advancing application development stages, mixed-mode simulations should be used as an intermediate step between application validation against pure simulation and full deployment to the target system. The *Simulation-Aided Design of Multi-Agent Systems* approach proposed in this paper builds on these ideas and in certain application domains has a potential to significantly reduce risks involved and even speed up the development process.

References

1. Ozalp Babaoglu, Hein Meling, and Alberto Montresor. Anthill: A framework for the development of agent-based peer-to-peer systems. In *International Conference on Distributed Computing Systems (ICDCS)*, pages 15–22, 2002.
2. Kent Beck. *Test Driven Development: By Example*. Addison-Wesley Professional, November 2002.
3. Carole Bernon, Marie Pierre Gleizes, Sylvain Peyruqueou, and Gauthier Picard. Adelfe: A methodology for adaptive multi-agent systems engineering. In Paolo Petta, Robert Tolksdorf, and Franco Zambonelli, editors, *ESAW*, volume 2577 of *Lecture Notes in Computer Science*, pages 156–169. Springer, 2002.
4. Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8:203–236, 2004.
5. N. Collier. RePast: An extensible framework for agent simulation. Technical Report 36, The University of Chicago, Social Science Research, 2003.
6. Paul M. Duvall, Steve Matyas, and Andrew Glover. *Continuous Integration: Improving Software Quality and Reducing Risk*. Addison-Wesley Professional, July 2007.

7. The European Organisation for the Safety of Air Navigation. EUROCONTROL BADA. http://www.eurocontrol.int/eec/public/standard_page/proj_BADA.html, 2011.
8. M. E. T. Horn. Multi-modal and demand-responsive passenger transport systems: a modelling framework with embedded control systems. *Transportation Research Part A: Policy and Practice*, 36(2):167–188, 2002.
9. M. Jakob, O. Vaněk, Š. Urban, P. Benda, and M. Pěchouček. Employing Agents to Improve the Security of International Maritime Transport. In *Proceedings of the 6th workshop on Agents in Traffic and Transportation (ATT2010)*, May 2010.
10. G. Jakovljevic and D. Basch. Implementing multiscale traffic simulators using agents. In *Information Technology Interfaces, 2004. 26th International Conference on*, volume 1, pages 519–524, June 2004.
11. N. Koenig and A. Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator. *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, 3:2149–2154, September 2004.
12. John McCarthy. Elaboration tolerance. http://www-formal.stanford.edu/jmc/elaboration.html., 1999.
13. Dušan Pavlíček Michal Jakob, Eduard Semsch and Michal Pěchouček. Occlusion-aware multi-uav surveillance of multiple urban areas. In *6th Workshop on Agents in Traffic and Transportation (ATT 2010)*, 2010.
14. L. Padgham and M. Winikoff. Prometheus: A practical agent-oriented methodology. *Agent-Oriented Methodologies*, pages 107–135, 2005.
15. Michal Pěchouček, David Šišlák, Dušan Pavlíček, Přemysl Volf, and Štěpán Kopříva. AGENTFLY: Distributed Simulation of Air Traffic Control Using Unmanned Aerial Vehicles. In *Proceedings of 2nd Conference for Unmanned Aerial Systems (UAS)*, March 2010.
16. M. Pipattanasomporn, H. Feroze, and S. Rahman. Multi-agent systems in a distributed smart grid: Design and implementation. In *Power Systems Conference and Exposition, 2009. PSCE'09. IEEE/PES*, pages 1–8, 2009.
17. D. Šišlák, P. Volf, and M. Pěchouček. Agent-Based Cooperative Decentralized Airplane-Collision Avoidance. *Intelligent Transportation Systems, IEEE Transactions on*, (99):1–11, 2009.
18. D. Šišlák, P. Volf, and M. Pěchouček. Agent-Based Cooperative Decentralized Airplane-Collision Avoidance. *Intelligent Transportation Systems, IEEE Transactions on*, (99):1–11, 2010.
19. Procerus Technologies. Procerus Technologies: Fly Light with world's smallest UAV Autopilot. http://procerusuav.com/, 2011.
20. U. Wilensky. Netlogo. Technical report, Center for Connected Learning and Computer-Based Modeling, Northwestern University, 1999. http://ccl.northwestern.edu/netlogo/.
21. Franco Zambonelli, Nicholas R. Jennings, and Michael Wooldridge. Developing multiagent systems: The gaia methodology. *ACM Trans. Softw. Eng. Methodol.*, 12(3):317–370, 2003.