Auton Agent Multi-Agent Syst (2007) 14:61–85
DOI 10.1007/s10458-006-9002-5

Applications and environments for multi-agent systems

Paul Valckenaers · John Sauter · Carles Sierra · Juan Antonio Rodriguez-Aguilar

Published online: 25 August 2006

Abstract This paper addresses multi agent system (MAS) environments from an application perspective. It presents a structured view on environment-centric MAS applications. This comprises three base configurations, which MAS applications may apply directly or combine into a composite configuration. For each configuration, the paper presents key issues, requirements and opportunities (e.g. time management issues, real-world augmentation opportunities and state snapshot requirements). Thus, the paper delineates what environment technology may implement to serve MAS applications. Sample applications illustrate the configurations. Next, electronic institutions provide an example of an environment technology, addressing norms and laws in an agent society, already achieving some maturity. In comparison, application-domain specific environment technologies still are in embryonic stages.

Keywords Environments for multi-agent systems · Multi-agent system · Multi-agent applications

1 Introduction

Recently, the concept of an environment for multi-agent systems (MAS) is being recognized as a promising research area, manifesting itself in successful AAMAS

P. Valckenaers (✉)
Department of Mechanical Engineering, Katholieke Universiteit Leuven, Leuven, Belgium
e-mail: paul.valckenaers@mech.kuleuven.be

J. Sauter
NewVectors LLC, Ann Arbor, MI, USA
e-mail: john.sauter@newvectors.net

C. Sierra · J. A. Rodriguez-Aguilar
IIIA-CSIC, The Artificial Intelligence Research Institute of the Spanish Research Council, Spain
e-mail: sierra@iiia.csic.es

J. A. Rodriguez-Aguilar
e-mail: jar@iiia.csic.es

workshops [10, 11]. This research on MAS environments originates from realizing the benefits and potential of making the environment explicit. Indeed, in many applications, this environment comprises much more than some computation and communication infrastructure [39].

This paper addresses MAS applications that are heavily influenced by the structure of an environment in which their agents operate. Sample applications are the control of a cooling system on a ship [23], of manufacturing systems [35], or of a group of surveillance robots [26]. This paper provides a structured view on what MAS environments may contribute to the development, installation and operation of such MAS in future.

1.1 Related work

Within this special issue, Weyns et al. discusses what a MAS environment, seen as a first-class abstraction, entails [38]. Other contributions in the special issue address technological aspects of MAS environments [14, 28, 37]. This paper is complementary by providing an applications perspective: what does the environment deliver to applications, what is its role/position within applications? It provides a structured view that constitutes a touchstone for conceptual and technical contributions: it facilitates the assessment of completeness and relevance, it contributes to identifying development opportunities, and it provides a base to position individual contributions by others in a bigger picture from an applications' viewpoint. For instance, the "abstract computational resources" put forward by the extended version of Gaia [39] support "sensing, affecting and consuming." This paper reveals what other services can be offered.

Helleboogh et al. discuss simulation in the context of MAS environments [14]. Both papers concur in key points. Helleboogh concludes that "the simulated environment must incorporate all necessary information to represent a single snapshot of the real environment." The composite configuration in Sect. 5 requires both read and write access to this state. The simulation configuration in Sect. 2 emphasizes time management functionality. This is a prerequisite to cope with Helleboogh's statement that "modeling, managing and maintaining over time all dynamism in the simulated environment is crucial."

Related publications on applications within this research field discuss, relative to this paper, a specific application [3, 5, 12, 22, 26, 33, 40]. They present valid solutions for a specific application and discuss what the environment brings to their (class of) applications. In contrast, this paper brings a generalized view. As such, these related research results have served as relevant input to this paper.

Most agent-oriented design methodologies focus on goals and decision processes and neglect addressing the environment explicitly [6]. Still, Klein and Giese present a MAS development approach that concurs with the structured view in this paper [19]. All relevant entities in the world-of-interest are modeled 'as they are'. Environment processes are modeled. The modeling supports emulation and reasoning. The approach distinguishes between environment models being correct and being appropriate (correctness being independent from the application). Environment services correspond to the environment augmentations in this paper. Overall, the approach of Klein and Giese answers the needs of environment-centric applications well.

1.2 Terminology

Within this paper, the term configuration refers to a combination of an agent system, an environment for multi-agent systems and a world-of-interest. The agents in the application constitute the agent system. A multi-agent system (or MAS) consists of an agent system and an environment. The intersection of the agent system and the environment is empty. Environment refers in this paper to a software system that does not include the world-of-interest.

The world-of-interest is application-specific and refers to the part of the world (in cyber space, the physical world...) that is relevant to the application. For instance, for an Internet search engine, this world-of-interest comprises web pages on the Internet, the links on these pages and the content of the pages. For a climate control application, this world-of-interest comprises the rooms in the building, the doors, the heaters, etc. This complies with the approach in [17], in which the stability of such a world-of-interest relative to the application (functionality) itself is recognized.

1.3 Paper overview

This paper discusses three base configurations for MAS applications. The first configuration addresses simulation. Next, applications interacting with the physical world are addressed. In the third configuration, applications interact with a world-of-interest in a virtual world. Next, composite configurations, combining several base configurations within a single application, are presented. An example illustrates each configuration. This provides an idea of the range of applications that MAS environments may address. Next, the discussion focuses on Electronic Institutions as an example of existing technology addressing laws and norms of an agent society in MAS environments. Finally, conclusions are given.

2 Configuration I—simulation applications

Configuration I targets multi-agent simulation [20, 23, 24]. This section first describes this configuration. Next, it discusses three generic issues for the environment in this configuration: emulation, the world-of-interest and time. Furthermore, an example illustrates the configuration and the contribution of the multi-agent environment therein.

2.1 The configuration

Figure 1, showing the agent system, the environment, the world-of-interest and their relationships, depicts Configuration I. Most importantly, the relationship between the world-of-interest and the multi-agent system consists of up-front modeling; there is no run-time connection during MAS execution. Through modeling, the agent system and its environment jointly reflect a world-of-interest.

Agents model the actors, which are goal-oriented decision-making entities within this world-of-interest. These agents can be models of other entities (e.g. humans) where the simulation aims to assess system behavior or validate the agents as a model of something else [4, 9]. Importantly, the agents often model themselves: the actor is the deployed version of the agent. The latter happens when the simulation is a first

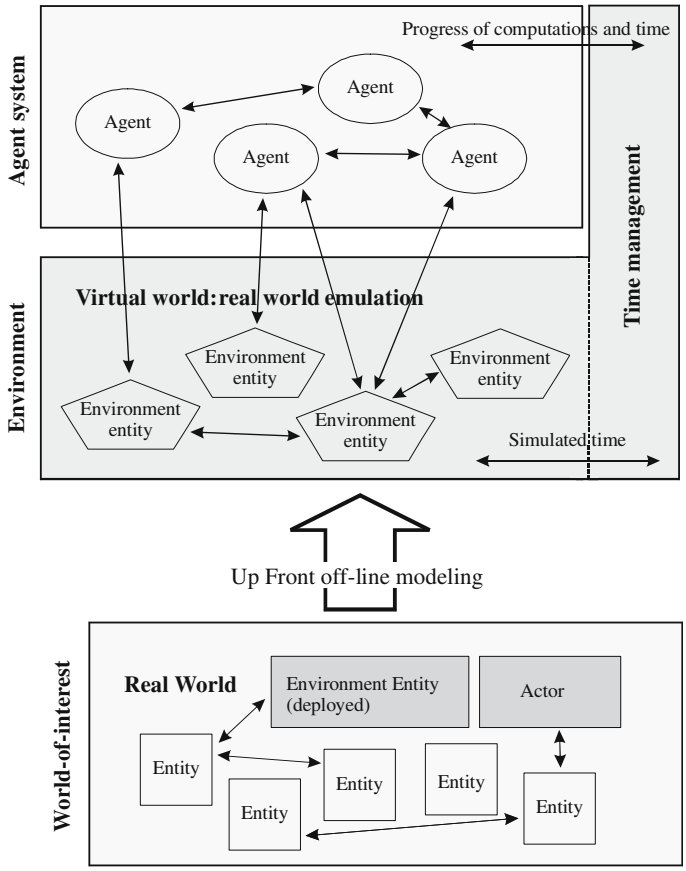

Fig. 1 Simulating the real world. The deployed environment entities are optional and correspond to an eventual augmentation of the real world (see Sect. 3.2)

phase in the development of a multi-agent system that is subsequently deployed in a real-world application (see 3.1).

Environment entities and laws reflect the remainder of the world-of-interest. This includes the modeling of:

- Behaviors of the environment entities,
- State trajectories of the entities,
- Interactions of the entities with each other,
- Response of the entities to inputs from the agents

Constraints and physical laws of the world-of-interest govern the above items. To the extent that it is feasible, these constraints and laws are modeled within the environment. The model is executable: it emulates the world-of-interest for the agent system.

Configuration I consists of the combination of this executable model (or emulation) provided by the environment and the agent system.

2.2 Emulation

Emulation implies that the environment replaces entities in the world-of-interest for the agent system with a virtual equivalent. Ideally, the agents are unable to distinguish the emulation from the actual world-of-interest. In practice, the perceptible difference must be negligible for the simulation purposes at hand. By ensuring this difference is negligible across the envisaged application range, reusable models are developed.

Validation of the emulation model begins by correlating the behavior of the model with the corresponding entity in the world-of-interest, independent of the context in which the emulation model is used. This activity verifies model correctness. However, validation of the overall simulation still requires testing the system as a whole to ensure that the system-wide behavior is consistent with the world-of-interest. This activity ensures that the emulation model is adequate. Some of the standard techniques used for verification and validation of simulation models can be used for this purpose [2, 18].

Importantly, the environment models can capture domain expertise and knowledge. Today, the performance of too many applications suffers from the naïve models embedded in their information systems (e.g. ignoring inertia in fast-moving robotic applications). The environment in MAS constitutes an opportunity to capture domain expertise, facilitating reuse and ensuring better quality in a wide community.

2.3 World-of-interest

There is great diversity concerning the worlds-of-interest targeted by MAS environments:

- The emulation may reflect some part of the real world. E.g., the emulation may model the existing European railway infrastructure.
- The emulation may correspond to something that could exist in the real world, but still needs to be realized or no longer exists. E.g., emulation models may reflect possible extensions of the high-speed railway infrastructure or emulate the railway system as it existed during World War II.
- The emulation may even correspond to parts of an imaginary world, which in principle cannot exist in the real world. This may happen in some computer game applications.

In the first cases, the emulation shares a desirable property with the real world. It strives to always remain in a globally consistent and coherent state. Everybody knows how our universe handles two cars following intersecting trajectories. The real world has further properties (e.g. conservation of mass and energy) that address important system design concerns. By greatly restricting the possible state trajectories that the entities in the environment can take, these properties are a powerful aid to the development of large complex models. An in-depth discussion is outside the scope of this paper [34]. However, it makes sense to keep such properties in mind even when designing fictitious worlds.

Hence, in Configuration I, the agent system is embedded in an environment with laws that impose consistency and coherence, as does the real world. Similar to the real world, these laws of the environment may sometimes act against the agents' objectives.

2.4 Time

Time management is a key component of the environment in this configuration. Indeed, the simulation must be sufficiently accurate and therefore can only tolerate a limited distortion in the progress of time. Still, users want simulations to execute fast. Together, these requirements translate into the need for (simulation) time management functionality in which the progress of time (and computations) in both the emulation and the agent system needs to be controlled. In a simulation, time can progress in three ways:

- Discrete steps. Time advances in discrete steps of not necessarily equal intervals. Discrete event simulators use this mechanism to advance time as quickly as possible during periods where no observable state changes are occurring.
- Continuous steps. Time advances in discrete steps of equal intervals. This is often used when emulations solve differential equations. Continuous time systems may change the granularity of the times steps (e.g. to cope with stiff systems), but this is done infrequently.
- Real time. The simulation system is tied to a real time clock with a factor governing how fast the simulated time advances with respect to the real time clock.

There are several situations requiring support for time management functionality:

- The system that is simulated is a distributed system. Relative progress of time and computations in the distributed system need to be properly accounted for. Mirroring the distribution in the simulation will not solve the problem because the system properties may be partially unknown and time-variant (e.g. behavior depends on the speed of Internet connections). The simulation needs to control such variations in relative progress across the distributed system, and must perform an adequate number of replications to assess the impact of variability and uncertainty.
- The execution of the multi-agent simulation is distributed to accelerate the simulation itself.
- Emulation takes longer than real-time. The agent system needs to be frozen whenever the emulation cannot respond in time. Otherwise, the agent system receives too much computation time. Then, conclusions drawn from the simulation may be painful and expensive illusions. This situation is typical when emulating physical phenomena by numerical solutions for differential equations [23].
- Emulation takes much less than real-time. This situation is common in discrete event emulations. This requires an environment in which time progresses as real time whenever agents are deliberating, and in which time instantaneously progresses until the next event on the event calendar whenever all the agents are idle. Such a system requires mechanisms (cooperative or otherwise) to detect agent activity and involves rerouting all timer functions (used by the agents) through the emulation (event calendar) [41].

In the first two cases, the environment must synchronize time across all systems. This is challenging in discrete event systems where time would advance at widely different rates if systems were left to execute independently. A description of synchronization mechanisms is beyond the scope of this paper. See [13, 14] for examples. The latter two cases require rather simple time management functions.

2.5 Example

This section discusses a sample application and highlights the role of the environment therein. Note that, since explicit support for the environment in MAS is lacking today, the application illustrates what explicit environment support could contribute. Other examples can be found in [9] and [23].

XSpec (eXecutable Specification) was an agent-based environment for manufacturing control in the early '90's [36]. XSpec emphasized the dual physical and control nature of agents in a manufacturing control application. The control elements in XSpec correspond to the agent system in Configuration I, while the physical elements correspond to the entities in the environment. E.g., a robot would consist of a control element representing its control logic and the physical element would represent a model of the dynamics of the robot arm and its motion in response to movement commands from the control element. Control elements sensed the state of and issued commands to the physical elements within their scope of control. Finally physical elements communicated the exchange or movement of matter or energy to other physical elements.

The authors claimed that the separation of the control (agent) and physical (environment) elements led to several advantages:

1. The physical models could be more readily reused since they represented a simple model of a physical entity irrespective of the application domain. In one case they achieved a 70% reuse of the physical elements.
2. The executable control models, freed from any physical modeling artifacts, could be transitioned directly into working control software.

XSpec relied on executable models to simulate the control and physical elements in a manufacturing system. XSpec recognized that there was no single simulation environment that adequately models all the control and physical elements. The logic for a control element was specified using an executable control specification language (e.g. IEC 848). The physical elements were modeled in an appropriate simulation tool. These models were integrated in a distributed simulation environment called XFaST (eXecutable Factory Simulation Tool). XFaST integrated the various control and physical simulation environments and synchronized time so the behavior of all elements in the system could be modeled together.

After validating the control logic against the physical simulation model, the specification was compiled or translated directly into an execution environment and the physical simulation models were replaced with the real world entities or modified to serve as interfaces to the real physical devices. XSpec was used to design and build several successful applications including an automotive weld line, an automotive door line, and a flexible assembly material transport system for an electronics manufacturer. Numerous design errors and performance problems were identified and corrected using the XFaST simulation environment greatly reducing the integration time. This demonstrates the use of a Configuration I system for developing a Configuration II system. The XFaST framework also demonstrates the use of time management in a distributed simulation environment.

In this application the environment was composed of the set of interacting "physical elements." These provided a model of the real world elements that were under the control of the MAS ("control elements"). The specific services provided by the environment were:

- Emulation of the real world entities. The behavior of the real world entities in response to control signals from the MAS, their internal states, and their interactions with other real-world entities was modeled offline. These models were implemented with kinematic or discrete-event simulators depending on the complexity of the behavior that needed to be emulated.
- Time management. Since multiple simulation tools were employed simultaneously, the environment provided the critical time synchronization services to coordinate and manage time across the different simulations. In particular time synchronization was managed between continuous and discrete event models.

The environment enabled the development of high fidelity models of the entities since they were readily re-used in multiple applications. Once the MAS was completely tested against the simulated real-world entities, they could be replaced by the actual real-world entities.

2.6 Summary and remarks

Configuration I targets simulation applications. Agents interact with the environment, which models entities in the world-of-interest. There is no run-time connection from the real world to the models in the environment.

In this area, repositories of reusable emulation models can capture important domain knowledge and improve the overall quality of the models employed in MAS. The fact that these emulation models correspond to parts of the real world facilitates integration and reusability significantly. To this end, emulation model developers must avoid relying on a specific context and only reflect the corresponding part of the world-of-interest.

Specific for the simulation context is the need for time management functions. Some of these functions are the subject of ongoing research [14]. In contrast, the ability to slow down agent computations when the emulation requires more than real-time, or the ability to emulate in real-time while agents are deliberating in combination with the speed-up of discrete-event simulation when agents are idling, are relatively simple time management functions. Nonetheless, such functionality is absent in existing simulation software and cannot be added later. Environment technology has the opportunity to provide suitable support at the core of its implementations.

3 Configuration II—real-world interaction

Configuration II targets applications rooted in the physical world. These applications interact with entities in the real world through some intermediary (sensors and actuators). E.g., when one agent makes its robot kick a soccer ball, other agents might perceive this through a vision sensor and act upon that. Likewise, an office climate control system reads temperatures and manipulates heaters, coolers and blowers to adjust temperatures.

This section first describes the configuration. Next, it discusses how the environment may augment the real world in this configuration. Finally, a sample application illustrates the configuration and the contribution of the environment therein.

3.1 The configuration

Figure 2 depicts Configuration II comprising the agent system, the environment, the world-of-interest and their relationships. This world-of-interest is part of the real world. Typically, the real-world entities evolve slowly from the perspective of computer processes. In contrast to simulation, the environment and the world-of-interest have a run-time connection.

Environments for this kind of application can be simple reflections of the current entities in the physical world, or a combination of real-world and simulated world emulation. In the simple version, the environment offers an abstraction of the current real-world entities and their state to the multi-agent system. Mixed environments include abstractions of real-world entities and fully simulated elements. Environments can also capture events in the past or provide projections of future states of the real world entities.

The agents only interact indirectly with the world-of-interest, using the environment as an intermediary. In practice, it is possible to develop multi-agent applications where the agents are directly connected to real-world sensors and actuators. However,

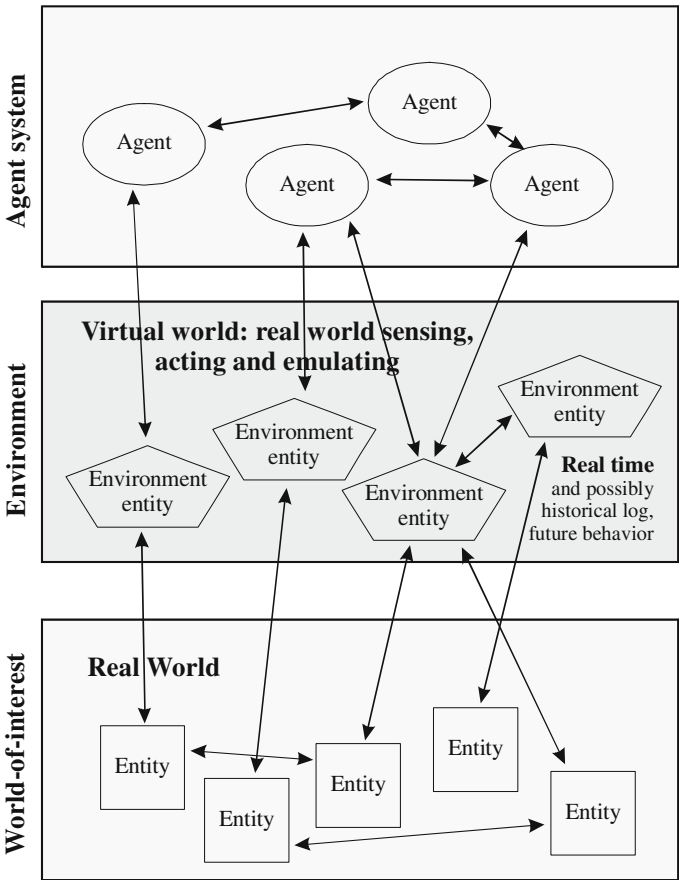

Fig. 2 Real-world interaction

except for the simplest applications, there exist compelling motivations to interact indirectly with the real world through environment entities reflecting their real-world counterparts.

The main incentive is the augmentation of the real world that can only be achieved by the environment when it is involved in, and to some extent in control of, all interactions between agents and the real world. This is similar to the way in which modern computer operating systems are a compulsory intermediary between normal users and the computer resources. This is particularly true for the environment augmentations enforcing proper usage of real-world entities, which are becoming more and more important in view of software increasingly becoming pervasive and mission-critical.

Furthermore, virtually all applications targeted by Configuration II are preceded by a Configuration I development. MAS controlling physical entities have many safety concerns (human and/or economical). Validation of these systems is a critical aspect of their development. Several standards governing the development of complex systems rely heavily on simulation as a tool for validating the system (IEEE 1012-2004 [16], US DoD 5000.2-R [32], and NIST Special Publication 500-234 [7]). Hence most multi-agent systems that interact with the real world undergo extensive testing and validation in a Configuration I. Hence, emulation models of the real-world entities exist when the Configuration II development starts. These models serve as a basis for developing the necessary intermediary environment entities between the agents and the real-world entities.

3.2 Augmentation of the real world

The key contribution of a MAS environment in Configuration II is augmentation of the real world. These augmentations offer various kinds of functionality, as presented below.

3.2.1 Enhanced access to the real world

The connection between the multi-agent system and the real world often represents a challenge to application developers. Sensors often measure only indirectly and approximately the required information and they are not for free. Actuators pose similar challenges. The environment may support developers in this respect providing:

- Sensor processing. Raw sensory information must be processed before it provides useful information. Sensor data may need to be fused to provide information of value. Uncertainty estimates may need to be calculated and conflicting sensory inputs resolved. The environment provides a mechanism to process and manage the raw sensory data so agents can get the information they need at the level of abstraction they require to perform their functions.
- Virtual sensors. No application is instrumented sufficiently to provide all the required information directly. An environment can fill in the missing pieces of information by extrapolating between sensor readings to provide a current estimate or maintaining state estimates using system models for which there is no sensory confirmation (e.g. the current position in dead reckoning control).
- Virtual actuators. The environment may provide command-processing facilities at abstraction levels that are better adapted to the agent system, often combining several simple actuators. Virtual actuators also require access to sensor data and real-time processing power when implementing e.g. a feedback control law.

- Common time base. Combining sensory information from multiple sources often requires the moment of data capture on a shared (synchronized) time axis. Synchronized execution of commands also benefits from synchronized clocks [21]. The environment may provide this. Failure to provide such time base typically results in unattainable bandwidth and response time requirements.
- Publish-subscribe services. Agents can subscribe to environment entities to be notified when certain conditions are fulfilled.

There is an important caveat concerning these enhancements provided by the environment: access to (raw) sensor data and (native) command interfaces must be subject to resource management (see 3.2.3). Ideally, an environment service imposes minimal requirements for its services.

3.2.2 On-line auto-updated documentation

Applications often use information about the world-of-interest that is not observable through sensors but is nonetheless available in machine- or man-readable formats. Discovering such information, validating it, making this information accessible to the application and keeping it up-to-date often is a daunting task. The environment provides the appropriate vehicle to make this information accessible. This includes:

- Technical specifications. The environment entities can augment their real-world counterparts by publishing their technical and physical properties: weight, dimensions, shape, temperature operating range, etc. Note that these specifications will be time-variant when, for instance, a device is ageing and deteriorating.
- Virtual maps. The environment can maintain information about local connections and relative positions of the real-world counterparts, both peer-to-peer (e.g. room connected to a corridor) and parent child (e.g. room belonging to a building). Agents are able to discover the system topology and spatial properties through interaction with suitably augmented environment entities. Agents may also discover new information about the topology when it is maintained by the environment to reflect changes (e.g. when a new machine is installed in a room). Note that environment entities only need to know their own real-world counterpart and (the relative position of and local connection to) their immediate neighbors to implement this.

Attaching information about a real-world entity to its environment counterpart offers a simple mechanism to keep such information consistent on condition that this information is valid for the entity regardless of its context. Also, attaching information to an environment entity equipped with an update-checking mechanism keeps it up-to-date.

3.2.3 Regulating the usage of real-world entities

Similar to modern operating systems controlling user access to computer resources, the MAS environment ensures and encourages proper usage of the real world entities:

- Ownership management and access control. The environment can enforce rules governing ownership or manage multiple requests to control the same entity. An environment can, for example, keep two agents from trying to tell the same

vehicle to move in two different directions at the same time. Conversely, a proficient resource allocation service discourages developers from having their agents monopolize resources unnecessarily.
- Safety management. Rather than requiring each agent to anticipate and maintain safe operation of the system, the environment can ensure that only safe operations are allowed on the entities it manages. This could prevent the system from being commanded to exceed its design limits, perform unsafe actions, or perform other actions that might cause damage to the equipment.
- Enforcing constraints. On a more sophisticated (and challenging) level, the environment can enforce constraints on allowed actions. These constraints could be imposed by physical laws, technological limitations (e.g. device speed limits), or design rules (e.g. allowed operating envelopes). This creates opportunities to make the environment more autonomic and to provide higher-level functionality (e.g. coordination fields indicating imminent deadlocks).

Note that these augmentations can be application-specific, thus contributing functionality to the application (e.g. an unmanned air vehicle environment may limit the flight envelope depending on the mission). This suggests that the environment needs interfaces that allow it to be configured correctly for different applications (missions).

Furthermore, these augmentations can be domain-specific, covering a multitude of applications. Similar to application-specific augmentations, domain-specific augmentations can exploit their specific context to offer higher levels of functionality. In addition, since they are enjoying a sizeable user community, these augmentations can go a long way to more accurately reflect the world-of-interest and avoid imposing artificial constraints. For instance, in 'one-size-fits-all' solutions, transfer of ownership often is restricted to resources in a reference state (e.g. a robot manipulator must be immobile at its home position). Domain-specific augmentations may impose more liberal restrictions that reflect the real-world constraints more closely.

3.2.4 Information processing infrastructures

The environment can be used to augment the world-of-interest with features and functionality, non-existent in this world-of-interest, to support the information processing needs of the agents:

- Virtual stigmergic worlds. Stigmergic systems rely on interactions through signs in the environment. While insects may be able to easily deposit and sense chemical pheromones in the real world, it may be impractical for MAS to do the same. An environment can support a virtual world counterpart to the real world where the agents can deposit and sense simulated pheromones. Since they are simulated, these virtual pheromones can be given capabilities beyond their real world counterparts: specially designed propagation patterns beyond what the laws of diffusion would normally allow, evaporation rates above or below what is physically possible, or information content more complex than can be carried by a few different pheromone flavors. Likewise, the environment may process information that is deposited (e.g. insert it in an agenda) analogous to and beyond physical/chemical interactions in the real world.
- User interfacing (for normal users and system managers/administrators). The environment provides a rich infrastructure to build user interfaces, user interaction

models, diagnostic, and performance analysis systems. It may provide a family of applications with a consistent omnipresent minimal user interface with negligible recurring implementation efforts.

3.2.5 Past, present and future

Finally, the MAS environment allows the applications to transcend the present. In addition to the current state, the environment maintains information about the past, and even estimates about the future:

- The past. The environment entities may keep a log or trace of their history. Agents may use this to perform diagnosis, construct statistical models, identify patterns in behavior, etc.
- The future. Augmented environment entities are able to produce information about their future (behavior, properties). By maintaining a model of its behavior an entity may be able to predict its future state irrespective of the state of the rest of the system. For example, given its friction, inertial mass and a change in the applied torque, a fan could predict its final speed and how long it will take to reach that speed.
- The future reflecting commitments and intentions. In a resource allocation context, the environment entity predicts its future availability given the current commitments. For instance, a hotel room entity is able to receive bookings from the authorized agents in the reservation department as well as from the building maintenance agents. Based on the information therein, the hotel room entity predicts its future availability. A sample application of this principle is [35].

3.3 Example

A sample application is the control of unmanned systems. Sauter et al. [30] describes a demonstration by Altarum and Johns Hopkins University (JHU) of multiple unmanned ground and air vehicles (UAV) cooperating in a simulated exercise. The MAS used two stigmergic algorithms to control the behaviors of the unmanned vehicles. Altarum used a pheromone-based algorithm to control the behavior of the UAVs for surveillance and convoy patrol. JHU's force field algorithms were used to control the ground robots that performed surveillance and perimeter patrol around a group of buildings.

In this application the environment was primarily used to provide an augmentation to the real environment. It served as an artificial space in which the MAS could deposit and sense pheromones. The environment was responsible for managing the pheromone dynamics, which included aggregating deposits of pheromones in the spatial grid, propagating pheromones through the grid, and evaporating those pheromones over time. The MAS agents were even allowed to do something that real entities could not do with real pheromones: they could remove pheromone from a location. In this way the MAS environment was truly an artificial environment since it allowed the development of pheromone operations that violated physical laws and constraints.

The unmanned vehicle control application used the environment to provide the following services:

- The environment provided a standard interface between the MAS and both the simulated and the actual hardware. The standard interface allowed the agents to

get information on current location, heading and speed of the vehicle as well as issue commands for new headings and speed. A similar interface for the ground robot and the unmanned air vehicle (UAV) made it easier to develop a single agent that could control either vehicle.
- The environment maintained state information on all the entities in the system. An agent had a single interface and place to go to find out about the location of all other entities in the system.
- The environment provided a virtual world for the stigmergic algorithms. The agents were able to deposit and read the state of digital pheromones. The environment maintained the pheromones by performing all the accumulation, propagation, and evaporation functions on the pheromones. It also maintained all the force fields used by the JHU algorithms
- The environment served as an integrating platform between the two stigmergic algorithms. When the UAVs controlled by Altarum's pheromone algorithm needed to communicate information to other UAV's they used deposits in the environment. When they needed to communicate similar information to the ground robots controlled by JHU they used the same mechanism. The UAV agent did not know or care what agent responded to the deposit, or even what kind of algorithm the agent used.

The advantages of the first item (maintaining a standard interface) became clear when it came time to integrate the agent algorithms with the hardware platforms. Initially the MAS applications were developed and tested in a simulated environment (Configuration I). When the algorithms had been validated in simulation they were loaded unchanged onto the hardware control platform for the ground robot. The MAS application worked the first time without modification. A similar experience was had when installing the algorithms on the UAVs. First it was possible to reuse the same agents that controlled the ground robots to control the UAVs. Secondly, the agents were installed unchanged on the UAV and the control software worked first time. In fact the only change that had to be made to the algorithms was the knowledge of the UAV turning radius. The simulation had not previously taken this into account and the algorithm required a slight modification to predict future headings far enough in advance to accommodate the restricted turning radius of the platform. The latter demonstrates the need for quality emulation models that can be reused (some model deficiencies will be hard to remedy).

3.4 Summary and remarks

Configuration II targets applications rooted in the physical world. Agents handle all decision-making aspects; the environment reflects the remaining entities in the world-of-interest. There is a run-time connection between the environment and the real world. The agents only interact with the real world through the environment as an intermediary.

In Configuration II, the environment provides important augmentations of the real world. The functionality discussed typically is reusable and modular. Indeed, many of the above-discussed environment entities can be developed based on self-knowledge (of the corresponding real-world entity) only. Hence, such augmented environment entities can be constructed to be reusable wherever and whenever an instance of the corresponding real-world entity exists. And, much of the functionality is application-independent.

4 Configuration III—Adaptive structured information systems

In Configuration III, the multi-agent system operates on an external system in cyber space. In contrast to previous configurations, the link to the external world is straightforward. This section first describes this configuration. Next, it discusses functionalities offered by the environment in this configuration. Then, a sample application highlights the contribution of the environment therein.

4.1 The configuration

Figure 3 depicts Configuration III. The world-of-interest is an information system external to the multi-agent system. The environment and the world-of-interest have a run-time connection. Changes in the information system are important events, which the run-time connection must address. Preferably, the agents only interact indirectly with the world-of-interest, using the environment as an intermediary, but it is not mandatory.

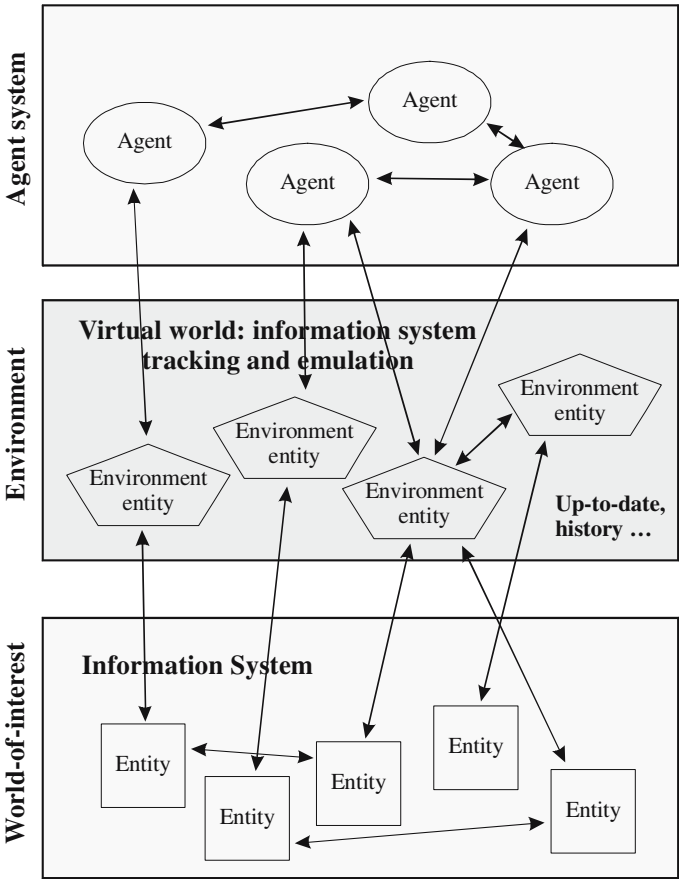

Fig. 3 Adaptive-structured-information-systems

Typically, environment entities reflect the information entities (e.g. web pages) in the information system as well as their structure (e.g. links between pages). For the configuration to provide added value to the agent system, the information system must have some structure that can be exploited. Moreover, the information system must evolve slowly from the perspective of multi-agent application. Nonetheless, the information system changes over time, which creates the conditions calling for an agent-based solution.

4.2 Augmentation of the information system

In Configuration III, the environment provides a reflection of the structured information system, which is kept synchronized, while adding extra functionality. More specifically, the environment may:

- Extend the information system:
  - Provide abstractions and aggregations of the information.
  - Maintain separation between the agent system and the peculiarities of accessing the information in a particular information system. This makes the agent system more reusable.
  - Cache information.
  - Preserve older versions of the information, no longer available in the external system.
- Monitor the information system:
  - Monitor the structured information system and provide a publish-subscribe service.
  - Collect information about usage by the agents, possibly correlated to a success or satisfaction indication.
- Add information processing structures:
  - Provide co-field- and stigmergy-supporting infrastructures.
  - Provide a supervision cockpit for the information systems operator(s).
  - Maintain and process additional information created by the agents and used to perform their functions.
- Moderate and constrain access, impose norms and regulations, possibly as a service used by the agents voluntarily.
- Create and maintain a relationship amongst several such adaptive structured information systems.

An important advantage of the environment concept is that the adaptive information system need not be aware of the functionality augmentation delivered by the environment. Likewise, the environment may collect information about agent activities without explicit cooperation from all these agents. However, some minimal cooperation by the agents may enhance the service level that can be supported (e.g. by indicating satisfaction and/or activity type).

4.3 Example

Bandini et al. discusses the development of a type III configuration to enhance access to web sites [3]. In this application, agents (human or otherwise) access the Internet through an intermediary system (environment) that observes user behavior.

Usage patterns are recognized and registered. This information is employed to offer enhanced services, which include:

- Caching the information on web pages, visited by the agents, while using web crawlers to discover changes. The crawling activity is driven by usage patterns.
- Cached information keeps information available after it has disappeared at its source. For instance, information that is regularly accessed by the users/agents will be preserved.
- A notification service informs agents of changes to web pages. Agents have to subscribe to this service.
- Web designers may exploit the usage patterns to enhance their web site.
- The agents are presented with an augmented view on the web pages that they are visiting, reflecting the usage patterns that are recognized by the environment.

This augmented view is based on both the behavior of the individual agent and of the agent community. The latter is hard to achieve without a proper environment; it also provides the most significant benefits to the agents. Consider the following. It is quite common that 'frequently wanted information' on a web site is hard to find and requires navigating along several links without informative labels (e.g. the input needed for a route planner on the web site of an airport). If sufficient agents execute such search patterns successfully, the environment recognizes this and provides shortcuts, eliminating loops and detours, also to agents that visit the web site for the first time.

4.4 Summary and remarks

In Configuration III the environment provides a mechanism to augment information systems. The augmentations are transparent to the information system and impose minimal requirements on the agents. The research community still has to explore what possibilities open up when the environment establishes and maintains connections and bridges between multiple information systems. Also, the research community has only started to discover the collection of suitable information systems (other than the Internet) and the corresponding augmentations.

5 Composite configurations—real-world interaction plus run-time simulations

The configurations discussed above constitute a basis for environment-centric multi-agent applications. However, not every application fits into exactly one of these configurations. Therefore, this section discusses Composite Configurations illustrating how the environment (and its software support) needs to integrate embedded base configurations. The discussion reveals services that cross the boundaries of base configurations and cannot be added as an afterthought. Which composite configurations are needed remains an open issue. The Composite Configurations discussed here originate from users concerned about deploying and operating their MAS.

Figures 4 shows the embedded configurations in a Composite Configuration as discussed in this section. In base Configurations I and II, the environment maintains the state of all entities at the same point in time. In Configuration I that is the current time in the simulation. In Configuration II it is the current wall-clock time of the real world. In the Composite Configurations, the environment not only maintains a model

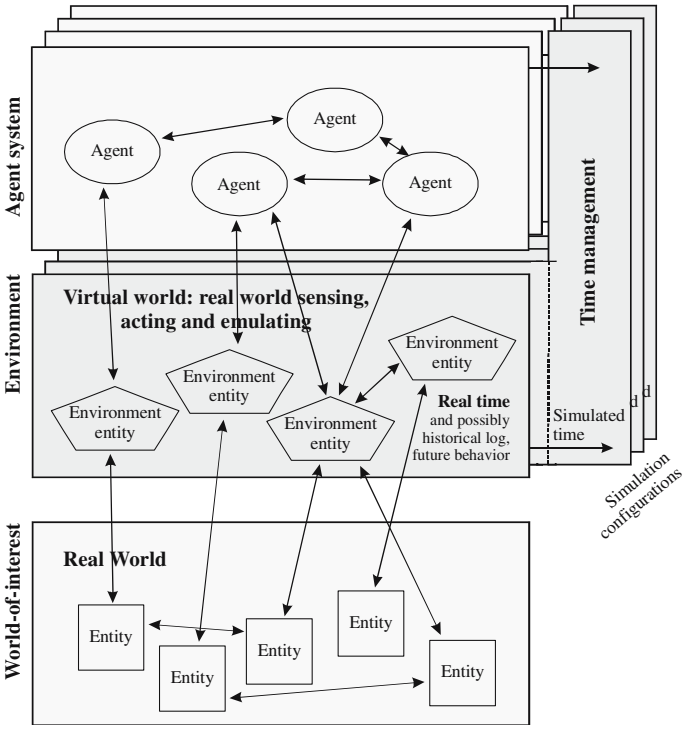

Fig. 4 Simulating the real world in physical-world-rooted applications

of the entities at a particular point in time, but it is able to support multiple models of the system at different points in time.

5.1 The composite configurations

Figure 4 depicts one possible alternative comprising the agent system, the environment, the world-of-interest and their relationships. Alternatively, the front layer may consist of a Configuration I model of the current state of the system. The layers behind this front layer each correspond to a Configuration I simulation of the system at alternative points in time and possibly varying states. Time progresses in the backend simulations much faster than time progresses in the front layer configuration. This enables multiple explorations of alternative system trajectories in the backend simulation before time has advanced significantly in the front layer configuration.

An important characteristic is the interaction between the front layer and the backend layers:

- The backend simulations can be initiated from the front layer configuration. The initial state of the simulation is a consistent 'image' extracted from the front layer configuration.
- Multiple backend simulations can execute in parallel with the front layer configuration to investigate alternative future scenarios.
- The configuration imposes an upper bound on the number of backend simulations, thus guaranteeing efficiency (given a computationally efficient MAS).

- Using the simulation results, the agents determine what actions to take in the present. These actions are implemented in the front-layer configuration.

Backend simulations can also be initiated from past states and projected future states:

- The maintenance of a history of environment state snapshots offers the possibility to initiate the simulations with 'images' from interesting states in the past. A typical usage would be to prepare adequate responses to major disturbances that have occurred and to which the agent system failed to respond well.
- Certain advanced Configuration II systems are able to forecast near-future states [15, 35]. This provides predicted snapshots of the system state that are used to initialize the simulation models. These simulations are allowed to execute until the time of the snapshot when the answers derived from the simulation results are injected into the front layer system.
- Backend simulations may also instantiate new simulations. This "forks" the simulation timeline so that multiple alternative futures can be investigated at any point in the backend simulation. The potentially exponential growth in the number of backend simulations is prevented by a suitable control mechanism.

These interactions reveal the need to provide access to the 'system state' in the embedded base configurations. The environment must support (1) the extraction of 'images' from base configurations, (2) the initialization of a Configuration I by such image and (3) the injection of 'simulation results' into the front layer configuration. Time and time management services are also prominent. Experience indicates that this kind of services cannot be implemented as afterthoughts.

5.2 Uses of the composite configurations

These backend simulations provide useful functionality to the MAS; the simulated environments provide a virtual playground for the agents to work with. The agents can use this as follows:

- What-if scenarios. The same complex nature of MAS that requires simulation for testing and validation during the design phase also hinders their behavior in the real world. Mapping multiple individual agent actions into an expected outcome is not always possible without simulating the system. By giving the agents a virtual environment to try out different actions, they can choose the best actions for accomplishing their goals without having to perform trial and error in the real world. These trial and error exercises in the backend simulations may start from an image provided by the front layer that corresponds to its current state. However, the front layer cannot be frozen and the developments therein may erode and eventually invalidate the course-of-action derived from the results of the backend simulations. To counter this, provided the front-end is able to generate short-term forecasts, the backend simulations may be initialized by predicted images of the front layer.
- Learning and adaptation. A simulated environment provides an opportunity for agents to use different learning mechanisms. For example agents could use evolution or other techniques to adapt to new, previously unforeseen circumstances by using future simulations to evolve the fittest behaviors.
- Explore possible futures. Most systems exhibit some stochastic and non-deterministic behavior. Simulation can be used to explore the range of possible futures that

the agents will encounter. This can be used for predicting possible future states as well as probabilities of reaching those states.
- Game playing. When the MAS is dealing with intelligent entities that are not under its control, game playing is an important tool for identifying best behaviors. Simulated environments can be used to model cooperating and non-cooperating entities in friendly and adversarial relationships.

5.3 Example

DARPA's RAID program is developing technologies to aid a commander in urban combat operations. Altarum is developing an Adversarial Reasoning Module (ARM) that attempts to predict the behavior of the adversarial (Red) units and advise the commander (Blue) on the best course of action [27].

ARM uses a composite configuration to solve the problem. Snapshots are maintained of the system state in the past, which includes the location of all the known entities, their actions, and the events that transpired. These snapshots are used to evolve a behavioral model for each of the Red entities in the system.

The model is used to predict the future behavior of the Red entity by running a simulation starting from the current state of the system and simulating up to 60 minutes into the future (using Configuration I). ARM uses a stigmergic approach to control the actions of Red, so the environment also provides support for the pheromone fields used in this simulation. Simultaneously, Blue agents are using the future simulations of Red's behavior to identify their best response. The models are stochastic so multiple simulations are executed to identify different possible futures. Based on repeated simulations, ARM is able to identify the most likely Red actions and most dangerous Red actions along with the best course of action for Blue to take.

ARM is currently being evaluated in a simulated wargaming exercise (a Configuration I simulation of current state with multiple backend simulations of the future). In the future it will be deployed as a combination Configuration II (receiving real world inputs on entity states and real-world events) and multiple backend simulations for estimating future states.

The RAID urban combat advisor application uses the environment extensively in a number of ways:

- It serves as an emulation of the real-world entities, modeling their behavior, and maintaining their state.
- It emulates the sensors in the real world.
- The environment was used to maintain a recent history of past states and events.
- The environment was used as an augmentation to the real environment similar to the UAV control application. It provided a spatial grid that agents could use to deposit and sense pheromones.
- Finally the environment was used as a future prediction tool. In addition to pages representing past states, pages also represented potential future states of the system. Agents representing the known and suspected units in an area would make various attempts at reaching their anticipated goals by simulating from the current state of the system forward in time, faster than real time. In this way several alternative futures all starting from the current state could be explored.

RAID represents a sophisticated use of the environment. By simultaneously emulating real-world entities, providing an artificial world to support the pheromone

algorithms, maintaining complex state histories, and supporting the simulation of multiple future scenarios, it provides rich suite of services to the multi-agent system for managing a complex application. By developing these services as an environment, the application developers were able to focus more effort at optimizing the execution environment so it was able to manage hundreds of entities over a city-wide landscape while keeping 60 minutes of state history and still having sufficient time left to investigate hundreds of future scenarios an hour into the future in a matter of seconds.

5.4 Discussion

Composite configurations are important for two reasons. First, they bring forward integration issues, which are likely to remain unresolved unless recognized early on. Second, they address key functionality for the deployment and operation of multi-agent systems, especially for Configuration II. The availability of a simulation 'playground' in the multi-agent environment makes a difference where it really counts: it convinces the human decision makers that are involved.

6 Electronic institutions and environments for multi-agent systems

Research into Electronic Institutions has been investigating MAS environments for several years [25]. E-Institutions are a technology to enforce and monitor the norms and laws that apply to the agent society in a given environment. Several Configuration I applications have been developed in the past (for e-commerce [29], for power management [31]) as well as a Configuration II in the actual trade of fish where humans and agents compete in real time [8].

Figure 5 depicts how the E-institution controls the interactions between the external agents and the world-of-interest. The E-Institution is part of the environment and is realized by a collection of so-called staff agents. The application's agent system consists of so-called external agents (i.e. external to the institution). External agents only interact through the environment via a set of specialized agents called governors. Furthermore, there is a collection of norms that structures all interactions in the environment. In other words, the dynamics of the environment is restricted to those that satisfy the social laws represented by the norms and enacted by the coordinated actions of governors and staff agents. As to the external world, it is considered to be similar to the configurations described above.

The flexibility of the concept comes from its clear separation of concerns between the internal behavior of agents and their external interaction (environment modeling). From the perspective of an agent, the real world is modeled in this respect as several components:

- As a number of agents (usually called staff agents) that model/expand their counterparts in the real world (as in Configuration II systems) or that simply behave according to an internal model (as in Configuration I systems).
- As a number of norms that restrict the behavior of agents preventing them to behave in unacceptable/impossible ways. In this respect norms can be thought as physical laws (hence in the direction of Configuration II systems) or as social conventions that shape/constraint the evolution of interactions (needed in both Configuration I and II systems).

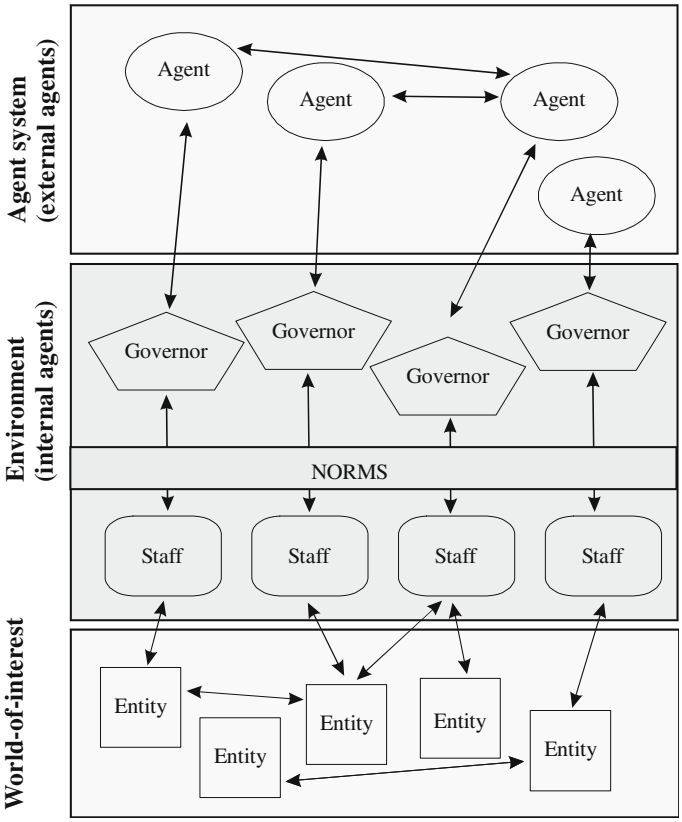

Fig. 5 Regulating the agent society

- As an explicit agreement on language. An institution contains as a component a particular Ontology and a set of illocutionary particles.
- As an explicit set of activities. Activities represent tasks solved by groups of agents and are nodes within a network that models the flow of agents. Actions within activities are further fixed as a protocol that will only permit certain dialogues among the agents.

Institutions establish conventions on behavior, language, and protocols that force agents to behave in particular and restrictive ways. In a sense the environment is given structure, so the agents have an easy comprehension of its working laws. These restrictions help a lot in the programming of agents as, by restricting the set of actions agents have to consider at each moment in time, it facilitates addressing the frame problem by limiting the set of options that agents have to think about.

Summarizing, Electronic Institutions are a representative example of more mature research concerning technology for MAS environments. E-Institutions provide opportunities to enhance the world-of-interest, specifically by inducing the agent society to obey norms and rules. They add value by offering guarantees and separation of concerns. Such technology complements the yet-to-be-developed environment technologies that bring a specific world-of-interest (application domain) to the agent world.

7 Conclusion

7.1 Summary

This paper presents three base configurations for environment-centric MAS applications. These configurations address applications that respectively execute simulations, perform real-world interaction and act upon structured information systems.

Simulation applications require the environment to emulate the world-of-interest and to support time management functionality. The nature of the world-of-interest helps to prevent inconsistencies and incoherence within the emulation. The agents in the simulation often model themselves, typically when they are to be deployed in a real-world application subsequently.

In the second configuration, the environment augments the real world. This includes enhanced access to the real world, auto-updated documentation, controlled access to and usage of real-world entities, information processing infrastructures, and virtual extensions into the past and future. A simulation along the first configuration often precedes these real-world applications.

In the third configuration, the environment augments adaptive structured information systems. These augmentations extend the information system, monitor the information system and add information processing structures. Furthermore, the environment captures the behavior of the agents themselves, allowing the agents to benefit from past experience of the agent community. Moreover, the environment may provide links between information structures.

Next, the paper discusses composite configurations and ascertains how support for base configurations must anticipate integration requirements. The functionality identified in the composite configuration includes access to the 'system state' in base configurations. Time and time management services are also important.

Finally, the paper addresses software support. Electronic Institutions are discussed as an example of relevant technology. E-institutions provide the means to enforce and monitor the norms and laws within an agent society. Similar technologies for specific application domains (e.g. manufacturing or traffic) still remain to be elaborated.

7.2 Future work

The above discussion uncovers a multitude of environment services and infrastructures, which are only supported by ad hoc implementations within today's applications. The development of more systematic support remains on the research agenda. Such support allows many applications to share development efforts, mobilizing the necessary resources to increase the quality and service level significantly.

The precise shape of software support for MAS environments also remains an open issue. Before the environment was recognized explicitly, many ad hoc implementations presented themselves as a collection of agents to the 'real agent system' (environment entities have an agent hull). Future research needs to investigate which embodiments of the environment are capable of providing the proper environment services, infrastructure and functionalities effectively and efficiently.

This paper discusses electronic institutions [1] as a sample environment technology to enact a particular type of environments for MAS. But, it is an open issue whether and how this technology can be further extended. More generally, the distribution of responsibilities over environment technologies and subsystems remains

an open issue. MAS environments are natural candidates to capture application domain expertise, but developments addressing this opportunity are still embryonic and isolated.

Finally, the designs of MAS environments and agent systems have different starting points. The design of a MAS environment starts by reflecting the world-of-interest or, more ambitiously, the relevant application domains. Suitably adapted design methodologies still have to be developed.

Acknowledgements This paper presents work funded by the Research Fund of the K.U.Leuven (Concerted Research Action on Autonomic Computing for Distributed Production Systems), and the projects Web-i(2) and OpenKnowledge.

References

1. Arcos, J. L., Esteva, M., Noriega, P., Rodríguez-Aguilar, J. A., & Sierra, C. (2005). Engineering open environments with electronic institutions. *Engineering Applications of artificial intelligence, 18*, 191–204.
2. Balci, O. (1997). Verification, validation, and accreditation of simulation models. In *Proceedings of the winter simulation conference*.
3. Bandini, S., Manzoni, S., & Vizzari, G. (2006). Web sites as agents' environments: General framework and applications. To appear in *Lecture notes in computer science, 3830*, Springer.
4. Beurier, G., Simonin, O., & Ferber, J. (2002). Model and simulation of multi-level emergence. In *Proceedings of IEEE ISSPIT* (pp. 231–234). Marrakesh.
5. Brueckner, S. A., & Van Dyke Parunak, H. (2005). Swarming distributed pattern detection and classification. *Lecture Notes in Computer Science, 3374*, 232–245.
6. Bussmann, S., Jennings, N., & Wooldridge, M. (2004). *Multiagent systems for manufacturing control, a design methodology*. Springer.
7. Christie, A. M. (1988). Simulation – An enabling technology in software engineering, SEI Report, 1988. Available at http://www.sei.cmu.edu/publications/articles/christie-apr1999/christie-apr1999.html.
8. Cuní, G., Esteva, M., Garcia, P., Puertas, E., Sierra, C., & Solchaga, T. (2004). MASFIT: multi-agent systems for fish trading. In *16th European conference on artificial intelligence (ECAI 2004)* (pp. 710–714). Valencia, Spain.
9. Dumont, B., & Hill, D. (2001). Multi-agent simulation of group foraging in sheep: Effects of spatial memory, conspecific attraction and plot size. *Ecological Modeling, 141*, 201–215. Elsevier
10. E4MAS (2004). First International Workshop on Environments for Multi-Agent Systems. In D. Weyns & V. Parunak & F. Michel (Eds.), *Lecture notes in computer science* (vol. 3374). New york: Springer Verlag.
11. E4MAS (2005). Second International Workshop on Environments for Multi-Agent Systems. In D. Weyns, V. Parunak & F. Michel (Eds.), *Lecture notes in computer science* (vol. 3380). Utrecht: Springer Verlag.
12. Helleboogh, A., & Holvoet, T. (2006). Testing AGVs in dynamic warehouse environments. To appear in *Lecture Notes in Computer Science* (vol. 3830) Springer.
13. Helleboogh, A., Holvoet, T., Weyns, D., & Berbers, Y. (2005). Extending time management support for multi-agent systems. *Lecture notes in computer science* (vol. 3415, pp. 37–48), Springer-Verlag.
14. Helleboogh, A., Vizzari, G., Uhrmacher, A., & Michel, F. Multi-agent modeling and simulation: the role of dynamism in the environment. *Autonomous Agents and Multi-Agent Systems* (this issue). Springer Verlag.
15. Holvoet, T., & Valckenaers, P. (2006). Beliefs, desires and intentions through the environment. *AAMAS'06*.
16. IEEE 1012–2004. Standard for Software Verification and Validation, June 2004. Available at http://ieeexplore.ieee.org/servlet/opac?punumber = 9958.
17. Jackson, M. A. (1983). System development. Prentice-Hall, ISBN:0138803285.
18. Kleijnen, J. P. C. (1995). Theory and methodology verification and validation of simulation models. *European Journal of Operational Research, 82*, 145–162.
19. Klein, F, & Giese, H. (2006). *Grounding social interactions in the environment*. (Vol. 3830). In *Lecture Notes in Computer Science*. Springer Verlag.

20. Klügl, F., Fehler, M., & Herrler, R. (2005). About the role of the environment in multi-agent simulations. In *Lecture notes in computer science* (vol 3374, pp 127–149).
21. Koninckx, B., & Van Brussel, H. (2002). Real-time NURBS interpolator for distributed motion control. *CIRP Annals-Manufacturing Technology, 51(1)*, 315–318.
22. Mamei, M., & Zambonelli, F. (2005). Motion coordination in the Quake 3 arena environment: A field-based approach. In *Lecture notes in computer science* (vol. 3374, pp. 264–278).
23. Maturana, F., Staron, R., & Hall, K. (2005). Methodologies and tools for intelligent agents in distributed control. *IEEE Intelligent Systems, 20(1)*, 42–49.
24. Multi-Agent Systems and Agent-Based Simulation, International Workshop Series, vols. 1534, 1979, 2581, 2927, 3415 of LNCS. Springer, 1998, 2000, 2002, 2003, 2004.
25. Noriega, P., Sierra, C. (2002). Electronic institutions: Future trends and challenges. In *Lecture notes in computer science* (vol. 2446). London, UK: Springer-Verlag.
26. Parunak, H. V. D., Brueckner, S. A., & Sauter, J. (2004). Digital pheromones for coordination of unmanned vehicles. In *Lecture notes in computer science* (vol. 3374, pp. 232–263). Springer Verlag.
27. Parunak, H. V. D., & Brueckner, S. A. (2006). Extrapolation of the opponent's past behaviors. In A. Kott & W. McEneaney (Eds.), *Adversarial reasoning: Computational approaches to reading the opponent's mind*. Boca Raton, FL: CRC Press.
28. Platon, E., Mamei, M., Sabouret, N., Honiden, S., & Van Dyke Parunak, H. Mechanisms and Opportunities of Environments for Multi-Agent Systems. *Autonomous Agents and Multi-Agent Systems*. (this issue) Springer Verlag.
29. Rodríguez-Aguilar, J. A., Noriega, P., Sierra, C., & Padget, J. (1997). FM96.5 a java-based electronic auction house. In *second Intern. Conf. on the Practical Application of Intelligent Agents and Multi-Agent Technology* (PAAM97) (pp. 207–224).
30. Sauter, J. A., Matthews, R., Parunak, H. V. D., & Brueckner, S. A. (2005). *Performance of digital pheromones for swarming vehicle control*. In *Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems* (AAMAS). Utrecht, Netherlands.
31. SLIE, (2003). D 3.3. Final rule sets plus support. The validation of SADDE methodology in the electricity market.
32. US DoD 5000.2-R (2002). Mandatory procedures for major defense acquisition programs and major automated information system acquisition programs. Available at http://dod5000.dau.mil/DOCS/Master.020405.Regulation.doc.
33. Valckenaers, P., Holvoet, T. (2006). The environment: An essential abstraction for managing complexity in MAS-based manufacturing (vol. 3830). In *Lecture notes in computer science*. Springer Verlag.
34. Valckenaers, P., Van Brussel, H., Bochmann, H. O., Saint Germain, B., & Zamfirescu, C. (2003). On the design of emergent systems: an investigation of integration and interoperability issues. *Engineering Applications of Artificial Intelligence, 16 (4)*, 377–393.
35. Valckenaers, P., & Van Brussel, H. (2005). Holonic manufacturing execution systems. *CIRP Annals-Manufacturing Technology, 54(1)*, 427–432.
36. VanderBok, R., & Sauter, J. A. (1990). A Case study: Integrated methods and tools for optimization. In *Proceedings of AUTOFACT Conference* (pp. 14-1–14-15) Detroit, MI.
37. Viroli, M., Ricci, A., Zambonelli, F., Holvoet, T., & Shelfthout, K. Engineering the environment of multiagent systems. *Autonomous Agents and Multi-Agent Systems*. (this issue) Springer Verlag.
38. Weyns, D., Omicini, A., & Odell, J. Environment, first-order abstraction in multiagent systems. *Autonomous Agents and Multi-Agent Systems* (this issue) Springer Verlag.
39. Weyns, D., Van Dyke Parunak, H., Michel, F., Holvoet, T., & Ferber, J. (2005). Environments for multi-agent systems state-of-the-art and research challenges. In *Lecture Notes in Computer Science* (vol. 3374, pp. 1–47).
40. Weyns, K. S., & Holvoet, T. (2006). Exploiting a virtual environment in a real-world application (vol. 3830) In *Lecture notes in computer science*. Springer Verlag.
41. Saint Germain, B., Valckenaers, P., Zamfirescu, C., Bochmann, O., & Van Brussel, H. (2003). Benchmarking of manufacturing control systems in simulation. In *Proceedings 3rd International Workshop on Performance Measurement* (IFIP WG5.7 Special Interest Group on Performance Measurement), (pp. 357–369). IFIP, Bergamo.