Mechanisms for Environments in Multi-Agent Systems

Survey and Opportunities

Eric Platon (platon@nii.ac.jp)
National Institute of Informatics, Sokendai, Japan

Marco Mamei (mamei.marco@unimore.it)
Università di Modena e Reggio Emilia, Italy

Nicolas Sabouret (nicolas.sabouret@lip6.fr)
Laboratoire d'informatique de Paris 6, France

Shinichi Honiden (honiden@nii.ac.jp)
National Institute of Informatics, Japan

H. Van Dyke Parunak (van.parunak@altarum.org)
NewVectors, Altarum Institute, USA

Abstract. The environment has been recognized as an explicit and exploitable element to design Multi-Agent Systems (MAS). It can be assigned a number of responsibilities that would be more difficult to design with the sole notion of agents. To support the engineering of these responsibilities, we identify a set of *mechanisms* that offer solutions to software designers. We describe the mechanisms, their usage in representative projects, and potential opportunities for further research and applications. The purpose of this article is to clarify the notion of environment in terms of mechanisms, from their abstract description to their practical exploitation. Mechanisms are expected to provide agent-based software designers with a set of design elements to build MAS that take advantage of the environment.

Keywords: Environment, Mechanism, Multi-Agent Systems, Agent-Oriented Software Engineering

1. Introduction

The environment has been recognized early as an explicit element of Multi-Agent Systems (MAS), but current applications often consider it as an implicit part of the system (Russell and Norvig, 2003; Ferber, 1999; Decker, 1995; Demazeau, 1995; Weyns et al., 2004). Recent work has however demonstrated that the environment is an exploitable design element. In particular, the environment is defined as a first-class abstraction with the role of providing the surrounding conditions for agents to exist, the mediation of agent interactions, and access to resources (Weyns et al., 2006). In other words, the environment in MAS is thought of as a design element responsible for the functions of underlying agent activities, i.e. interaction and agent resource support. The responsibilities of the environment can be addressed by an ensemble of mechanisms to design an application. In our context, a mechanism is a technical approach to solve a particular problem in the design and development of the environment responsibilities for the target MAS.

The goal of this article is to discuss mechanisms used in the research community to address environment responsibilities. The contribution of this work is to survey mechanisms for designing environments, and to determine potential research directions and application opportunities. Our contribution fits the idea to consider the environment as an explicit part in the design of MAS.

The survey is not intended to be exhaustive. Its purpose is rather to set forth representative achievements of the community on the issue of environments for MAS. The article proposes a view on mechanisms based on an interpretation of ongoing research and it is consequently subjective.

The article is organized as follows. In section 2, we survey a number of mechanisms of the environment and illustrate several research implementations of these mechanisms. In section 3, we describe representative application scenarios in the agent literature that already exploit the surveyed mechanisms. In section 4, we propose research directions to the MAS community on the issue of environment in terms of mechanisms, and we present potential opportunities that lend themselves to the usage of such mechanisms. Finally, section 5 summarizes and concludes the article.

2. Mechanisms

We define a mechanism of the environment in MAS as a technical approach to address a particular responsibility of the environment. In this section, we present a survey of those mechanisms that we identified in the research literature.

To this end, we propose a classification to reveal commonalities and distinctions between mechanisms. In general, the mechanisms of the environment serve essentially two kinds of activities related to levels of support provided by the environment (Weyns et al., 2006):

– **Interaction Mediation.** Several mechanisms provide agents with powerful means to interact with each other in a flexible and reliable way. For example, a shared-memory allows agents to exchange data by posting and retrieving information in an uncoupled way (e.g. shared-memory as tuple-space (Gelernter and Carriero, 1992)).

– **Resources and Context Management.** Several mechanisms provide agents with means to acquire and manage resources or contextual information. For example, an event-dispatcher mechanism can notify agents when some relevant events happen in the environment (Eugster et al., 2003).

It is important to notice that the boundary between these two groups is not strict and it is possible to devise mechanisms that support both of the above group definitions. The aforementioned shared memory, for example, can be used both to create a message box to uncouple agent interactions and to store context information. The distinction between the two groups appears noteworthy, however, to make the role of the environment explicit in inter-agent activities (interaction mediation) and agent-resource activities (resources and context management).

In addition to this main classification, it can be useful to distinguish between centralized and decentralized mechanisms. *Centralized* mechanisms consist of a single interface to the component (or components) that provides the intended functionalities. Such mechanisms appear as a single entity when agents exploit them. For example, a shared memory is centralized in this definition. Agents perceive the shared memory as a single entity through an interface, even though the shared memory is executed on different servers in the background for scalability reasons. *Decentralized* mechanisms have multiple interfaces, logically or physically decentralized, to the components executing the intended functionalities. For example, a network routing algorithm, enabling agents to interact in an epidemic manner with each other (e.g. peer-to-peer), belongs to this category. Agents do not perceive any kind of entity that can be singled out to represent the mechanism. They perceive the mechanism as decentralized across the MAS (e.g. the interfaces to each peer).

In addition to the classification, we identified three characteristics for each mechanism that are suitable to detail each class coherently.

- The *constituent atoms* are the basic elements to design the mechanism. For example, the atoms of a mechanism to support pheromone-based interaction are the data elements scattered across the environment representing the pheromones.

- The *creation and maintenance* refers to how the mechanism is set up. This characteristic represents how the constituent atoms are created and maintained. In the pheromone example, this would consist of algorithms for the storage, diffusion, and evaporation of the pheromone data.

- The *usage* details how the mechanism is used by agents. For example, the agents follow pheromone trails in an ant colony simulation.

In the following we describe mechanisms with the above characteristics, and we illustrate each case with representative work in the research community.

2.1. Interaction Mediation

Environment-mediated interaction channels are mechanisms to enable agents interacting and communicating in a flexible and uncoupled way. These mechanisms support the discovery of interaction partners and their subsequent interactions into a single mechanism.

Name: Environment-mediated interaction channels.
Motivations: Need to create smart and flexible interaction channels to connect agents on the basis of runtime system situation.
Constituent atoms: Repositories to store messages and subscriptions.
Creation and maintenance: Protocols to let agents join multicast groups, connect to tuple spaces, place subscriptions, or similar.
Usage: Agents exploit interaction channels to easily send and receive messages.
Examples: Multicast, Tuple- and Event-based interaction (Balbo and Pinson, 2001; Gelernter and Carriero, 1992; Freeman et al., 1999; Eugster et al., 2003; Cugola et al., 2001; Carzaniga et al., 2001)

The *constituent atoms* of this mechanism are a repository to store the data being communicated (i.e. messages, tuples, events, or subscriptions) and the data itself. *Creation and maintenance* are the algorithms and protocols to join and leave a multicast group, to access a tuple space, or to subscribe and unsubscribe to events. Agents *use* interaction channels to exchange messages, or to publish tuples and events to be exploited by other agents.

A first example of interaction channels are infrastructures for multicast interactions, where data is sent to all the agents matching specific conditions (Balbo and Pinson, 2001). In this context the environment provides a centralized and configurable message-box that decouples and mediates agent interactions. Agents can exchange messages as usual, but they can also put ‘filters’ in the environment to configure the reception of any message that matches specific criteria (e.g. all agents that provide a specified type of service). The environment is then responsible for sending the messages to suitable agents. Similarly, shared-memory and tuple-based approaches (Gelernter and Carriero, 1992; Freeman et al., 1999) belong to this category. Adopting this mechanism, agents leave data in suitable repositories to be later retrieved via pattern matching. Also, event-based infrastructures can be considered as part of this category as they deliver information on the basis of a publish-subscribe schema (Eugster et al., 2003; Cugola et al., 2001; Carzaniga et al., 2001). In this context, the event-dispatcher can be seen as an environmental abstraction able to notify agents about the activities of others.

From the centralization/decentralization point of view, in our opinion, these mechanisms are centralized in that (in most of the proposed designs) there is an interface that can be singled out and that wraps the underlying technique, such as the event dispatcher or the shared-memory server. In this regard, it is possible that some of these mechanisms could be realized in a decentralized way by applying peer-to-peer algorithms or some load-balancing techniques.

Name: Centralized synchronization mechanisms.
Motivations: Need for supporting the simultaneity of actions for system consistency.
Constituent atoms: Tables storing the locks in the system and holding the resulting effect of simultaneous actions.
Creation and maintenance: Protocols to let new agents access the service and to define coordination policies.
Usage: Agents access this service to coordinate and synchronize their activities.
Examples: Influence-Reaction Model (Ferber and Müller, 1996)

In addition to the above class, other mechanisms have been proposed to avoid possible conflicts in concurrent or parallel actions in MAS. Specifically, the fact that agents are implemented as parallel threads of execution leads to issues related to the synchronization and the ordering of their actions in the environment. *Centralized synchronization mechanisms* have been proposed to support agent actions with regard to these concerns. A notable approach is the influence-reaction model (Ferber and Müller, 1996). In this mechanism, the environment collects actions that the agents intend to perform, calculates the resulting effects depending on the simultaneity, and applies state changes on entities.

It is worth noticing that all the mechanisms presented in this section can be replicated in a number of distinct places in the environment. This means, for example, that we can build MAS where a number of shared-memory servers is spread across the environment, and agents can disseminate data and information in the place (i.e. shared-memory) they are currently visiting (see for example the Agilla platform (Agilla, 2006)). This kind of design weakens the notion of centralization. However, once connected to a specific replica, the agent still perceives the centralization of the mechanism due to a single interface.

Another important class of interaction mechanism is intended to support the creation of interaction networks. It is based on *overlay networks* that rule and describe relationships among agents. The topology of such a network is built such that agents can interact with one another efficiently. In other words, *overlay networks* are mechanisms to allow agents to structure and analyze the topology of the interconnections with other agents.

Name: Overlay networks.
Motivations: To represent and allow the maintenance of agent relationships.
Constituent atoms: Tables representing the physical, interactive, or social surrounding of agents.
Creation and maintenance: Protocols to let new agents join and leave the topology, and to deal with reconfigurations.
Usage: Agents access the tables to interact with each other efficiently.
Examples: Distributed Hash Tables (Ratsanamy et al., 2001; Ratnasamy and Karp, 2002; Rowstron and Druschel, 2001), Social Dependency Nets (Sichman et al., 1994)

Overlay networks are distributed data structures providing agents with a suitable application-specific view of their network of acquaintance (Ratsanamy et al., 2001; Ratnasamy and Karp, 2002; Rowstron and Druschel, 2001). This mechanism is not only related to distributed computing, but can be applied to any kind of logically networked MAS.

The *constituent atoms* of an overlay network are a set of tables defining a topology (e.g. tables indicate the neighbors of a specific agent, describing an overlay topology). *Creation and maintenance* is performed by means of specific protocols that adjust the overlay network topology upon connection and disconnections of agents. Their *usage* consists in agents sending and routing data to each other on the basis of the topology.

To exemplify such mechanisms in practice, we illustrate the principles with the Content Addressable Network approach (Ratsanamy et al., 2001; Ratnasamy and Karp, 2002). Other approaches, like Chord (Rowstron and Druschel, 2001), are based on the same principles. The basic idea of the content addressable network is that interactions in MAS are difficult to manage because of the system dynamics (agents connecting, disconnecting, and moving) and because of the unknown and dynamic topology of the network of acquaintance among agents.

To overcome these problems, agents are projected in a virtual space with a predictable topology, where interacting is easy and unnecessary details for the application are abstracted. When an agent enters the system, it gets a location in the virtual space (i.e. an address) and a reference to its new neighbors. The location of an agent is not related to the physical network, but to application parameters. So, for example, all agents having similar properties can get located in a specific area of the space. With such a mechanism in place, interaction becomes easy because agents look for specific partners in those areas of the space where they should be located.

The advantage of this mechanism is that the environment is built on the basis of application-level needs, and that it is directly accessible by agents to support their interactions. In other words, the environment decouples agents interactions from the underlying physical network and it provides a suitable 'glue' adapted to application-specific needs.

A second mechanism in this category is for *decentralized synchronization*, whose typical instance is the regional synchronization (Weyns and Holvoet, 2004). This mechanism is used to resolve conflicts between concurrent actions and to reduce synchronization cost in distributed settings. A region is a group of agents that act simultaneously, and independently from other agents. Regions are determined by a decentralized synchronization algorithm at the time actions are performed. In each region, the effects of actions are then calculated and applied, similarly to the centralized synchronization mechanism introduced in section 2.1. The notion of region confines the calculation of effects to relevant actions only, thus reducing the computational needs.

Name: Decentralized synchronization mechanisms.
Motivations: Need for supporting the simultaneity of actions for system consistency in distributed settings.
Constituent atoms: Tables of synchronization locks between agents of each region.
Creation and maintenance: Decentralized synchronization algorithm.
Usage: Agents access the tables to coordinate and synchronize.
Examples: Regional Synchronization (Weyns and Holvoet, 2004)

Overlay networks and decentralized synchronization are decentralized mechanisms, since no element can be singled out. These mechanisms are actually used by agents through an arbitrary number of interfaces.

2.2. Resources and Context Management

Resources and context management controls the access of agents to resources and contextual data in MAS. The *constituent atoms* of this category consist of an interface to access resources and a repository (or a set of repositories) to store information. *Creation and maintenance* consists in a set of protocols to access resources, to deploy data in the repositories, and to maintain the coherence of repository states. The *usage* consists in letting agents interact with resources and access the repositories to acquire data.

Name: Resource and context manager.
Motivations: Need to represent context information and resources in an efficient way.
Constituent atoms: Handling primitives and repositories.
Creation and maintenance: Protocols to wrap new resources types (interfacing and deployment); algorithms to manage repositories.
Usage: Agents access the repositories to interact efficiently.
Examples: TuCSoN, Event Heap (Johanson and Fox, 2002; Cabri et al., 2002; Omicini and Zambonelli, 1998)

These mechanisms are named as *resource and context manager* to emphasize their role in storing context information and interfacing with resources. Examples of this category of mechanisms are programmable tuple spaces, such as the MARS middleware (Cabri et al., 2002), TuCSoN (Omicini and Zambonelli, 1998), and EventHeap (Johanson and Fox, 2002). A programmable tuple space is a middleware that offers two kinds of functionalities. On the one hand, it provides a shared data repository that can be accessed via pattern matching. It is possible to store in the tuple space any kind of contextual information (e.g. <temperature, 30°C>), which can be retrieved later and shared among multiple agents (e.g. with the pattern <temperature, *>). On the other hand, the tuple space allows specific actions to execute when agents exploit it, e.g. to easily access data, to aggregate information, or to enforce security constraints.

These functionalities rely on the programmable characteristics of the mechanism that consequently differs from standard tuple space (Gelernter and Carriero, 1992) and blackboard architectures (O'Hare, 1991). Despite these differences, it appears that these mechanisms are well-suited in supporting agent interactions (we already presented the shared memory in section 2.1): They can serve both to communicate context-data about the environment and to exchange information among agents.

Name: Notification of contextual events.
Motivations: Need for the production and delivery of event notifications to create dynamic agent contexts.
Constituent atoms: Event dispatcher repositories.
Creation and maintenance: Protocols to let new agents subscribe to event sources and be notified upon event happening.
Usage: Agents trigger reaction on the basis of the events received.
Examples: Interaction filters (Balbo and Pinson, 2001), LoudVoice (Busetta et al., 2002), Tag interactions (Platon et al., 2005a)

The *notification of contextual events* is a mechanism that provides agents spontaneously with information about surrounding events in the environment. Events (from other agents or resources) trigger information flows generated by the environment to inform agents about the evolution of their local context. The *constituent atoms* of the mechanisms are the event dispatcher repository. *Creation and maintenance* are supported by algorithms to subscribe to specific event sources, to generate the events, and to deliver the events to suitable recipients. Finally, agents *use* the mechanism depending on application needs, e.g. to revise their knowledge about the surrounding world (Kakas et al., 2004).

The 'Environment to Support Active Communications' (ESAC) is an example of context notification in MAS (Balbo and Pinson, 2001). The environment has the architecture of an extensible mail server. Extensions are 'filters' placed by agents in the environment with a configuration rule to process messages that flow in the system. Rules allow agents to define the types of messages they want to receive, in addition to the ones addressed to them (filters model an instance of publish-subscribe patterns). ESAC was applied to a bus fleet management system where, for example, bus agents configure filters to receive all messages about traffic congestion between their current positions and the terminal stop. Complex event notifications can be produced with a language that specifies the filters placed in the environment. In our opinion, the two mechanisms of this section are centralized in that the interface to the underlying data spaces or entities can be singled out.

Another important class of mechanisms aims at describing the operational context of agents, i.e. the resources and contextual data available for exploitation by agents. *Overlay data structures* are distributed data structures encoding specific aspects of the operational environment of agents (e.g. resources such as databases available on a grid infrastructure). These overlays are propagated across a network in order to be easily accessible by the agents and to provide context information. They can be accessed piecewise as the application agents visit different places of the distributed environment. That is overlays let agents access the right information at the right location. It is worth mentioning that overlay data structures can also be dynamically spread by an agent in order to represent and 'communicate' its own activities.

Two representative examples in this category are *pheromone-based* and *field-based* data structures.

– **Pheromones** are overlay data structures that can be deployed across a distributed infrastructure by moving agents. In particular, as agents move across the infrastructure they can store specific data in the place (i.e. part of the distributed infrastructure) in which they are actually located. Following this approach, the overlay is deployed according to the agents motion patterns.

– **Fields** are overlay data structures that can be deployed across a networked distributed infrastructure. Agents can instantiate fields that propagate over the topology of the environment. Fields can embed specific propagation rules prescribing how they should spread. Following this approach, the overlay is deployed according to the field propagation rules. While pheromones are constrained to the agent motion pattern, fields can propagate independently.

Both mechanisms allow agents to enrich their environment with information that can be used to gather context information or to coordinate with each other. Stigmergic systems exemplify such coordination cases (Bonabeau et al., 1999).

The *constituent atoms* consist of multiple data repositories to store information in a decentralized way. *Creation and maintenance* refer to the deployment of data in the repositories and maintenance algorithms to keep the data consistent and updated. The *usage* is the access by agents to the repositories depending on their activities.

Name: Overlay data structures.
Motivations: Need for efficient, expressive contextual information.
Constituent atoms: Multiplicity of data spaces to store the overlay data.
Creation and maintenance: Protocols to deploy overlay data structure, maintain their intended distribution, and maintain data consistency.
Usage: Agents access the overlay data structure to get contextual information.
Examples: TOTA (Mamei and Zambonelli, 2004b), ObjectPlaces (Weyns et al., 2005b), UAVs (Parunak et al., 2004), Swarm Linda (Menezes and Tolksdorf, 2003)

Interesting examples of this mechanism are those relying on distributed data structures implementing pheromone approaches (Parunak et al., 2004; Brueckner, 2000). Agents perceive pheromones produced and deposited by others in the environment. The pheromones belong to an overlay data structure providing contextual information and pointing to specific resources. For example, applications to the simulation of unmanned vehicles show how artificial agents can automatically recognize team members, threats, and mission targets depending on the type of pheromones spread over the overlay data structure.

An example of mechanism for the field approach is the TOTA middleware (Mamei and Zambonelli, 2004b). The key idea in TOTA is to rely on spatially distributed tuples, propagated across a network to implement gradient fields on the basis of application-specific rules for representing contextual information. In TOTA, there is no notion of centralized shared tuple space. Instead, tuples can be 'injected' into the network from any node and can propagate and diffuse according to tuple-specific propagation patterns. Agents can sense the distributed tuples and decide how to react. For example, TOTA tuples are used to create fields as data structures to let the agents coordinate their movements by following the gradient of those fields.

Finally, an interesting proposal is the agent middleware 'ObjectPlaces' (Weyns et al., 2005b) that offers support to share information among agents in mobile and ad hoc networks. Specifically to ObjectPlaces, agents can describe and enrich the environment with *objects* that can be temporarily stored across suitable *places* (that are virtual containers stored in the ad hoc network itself). Agents invoke operations to add and remove objects or to observe the content of a specific object-place (via a pattern-matching process). Agents can also create object-places dynamically and link them together to form an overlay data structure.

Similarly to overlay network, all these kinds of mechanisms are decentralized in that the overlay data structures are spread across the whole systems and they cannot be singled out as individual entities.

3. Application Scenarios of the Environment Mechanisms

Most mechanisms are not exploited in isolation when creating a MAS application, but they are combined with one another to tailor a solution for a given problem, similarly to the design pattern approach in object-oriented programming (Gamma et al., 1999). This section aims at presenting how mechanisms are exploited in existing applications, and how some 'legacy applications' that do not explicitly use the environment abstraction can be advantageously redesigned with it. One particular issue that we want to stress with legacy systems is the frequent existence of a straightforward mapping between application requirements and a proper combination of mechanisms.

The structure of this section follows our taxonomy to review the applications. All in all, the categories of the taxonomy refer mostly to the following types of applications, depending on their relations to the 'real environment', i.e. the real-world in which we live in.

- *Simulations* are applications that aim at reproducing characteristics of the real-world by modeling and creating a computational world (e.g., the road network and buildings of a traffic simulator). A distinctive characteristics is that simulations are uncoupled from the real-world dynamics. Multi-Agent Based Simulations (MABS) are representative examples in the MAS community (MABS, 2004; Helleboogh et al., 2006).

- *Pervasive applications* complete the real-world with a computational one that evolves in synchrony. They are coupled tightly with the real-world dynamics by mean of intermediate actuators and sensors, such as automated robot-arms (e.g., Mars rover's) or RFID tags (RFID standard, 2006). The variety of pervasive applications is rapidly expanding with the advances in sensor network technologies (Zhao and Guibas, 2004). The MAS community has been especially creative in such applications where users or physical entities are part of the agent population.

- *Virtual Societies* stand in-between the two previous categories. They are applications that get inspiration from the real-world and may reproduce some of its characteristics. They can be loosely coupled with the real-world, but they have their own dynamics, i.e. they do not evolve in synchrony with the real-world (also 'synchronization points' can be defined). The aim of Virtual Societies is often to support human activities such as accounting, library management, or identity verification. They are expected to expand the capabilities of current approaches to computational systems, and also to support itself as a real society (AgentCities, 2006; OpenNet, 2006).

The criteria to distinguish between the three types are therefore the coupling with the real-world and the degree of reproduction of real-world characteristics. The categories are similar to the classification of Valckenaers et al. (Valckenaers et al., 2006).

For each category of the taxonomy, we review representative applications and motivate the use of mechanisms with the following recurrent properties.

- *Design abstraction* justifies the use of mechanisms that serve the design of MAS by abstracting and hiding the complexity of underlying details from agents. Design abstraction also refers to reducing the complexity of agents by assigning explicit responsibilities to the environment.

- *Coordination* motivates the use of mechanisms that address the often complex coordination of agent behaviors or motions, for example in the cases of self-organizing systems and solutions based on a shared memory.

- *Separation of Concerns* focuses on how some mechanisms can be used to disentangle the different concerns in designing an application. It supports the idea that the environment cross-cuts MAS, so that a given system design can be extended with additional aspects by exploiting (or simply activating) specific mechanisms in the environment.

The following survey elaborates on these properties to describe the benefits of using mechanisms, either for applications that already exploit them or other designs that could leverage the environment abstraction.

3.1. Centralized Interaction Mediation

Current concerns in simulations, intelligent sensor networks, and electronic institutions demonstrate the importance of interaction mediation in several applications. Although target systems are often distributed in these domains, centralized mechanisms are retained for design or performance issues.

3.1.1. *Simulation*

MABS often require explicit time management, as many simulation systems. Typically, discrete-time simulations need a 'ticker' that defines the pace of the simulation execution. Time is a dimension of MAS that impacts the activity of all agents as a global property. The global scope leads to assign time management to the environment, instead of an agent that has just a local scope in the system. When required, the management is often realized by a mechanism of the centralized interaction mediation category.

In particular, time management is mostly supported by two mechanisms, namely the notification of context events and the centralized synchronization. The notification of context events yields system-wide time references such as the clock 'ticks' in discrete-time simulations. The synchronization mechanism deals with timing by ensuring that agent actions in the system are consistently executed, i.e. they verify proper action interleaving (Ferber and Müller, 1996). Both approaches are also simultaneously exploited as their contribution to the time dimension of the system are complementary.

The example of time management illustrates how mechanisms in this category can be advantageously exploited to design specific requirements of MABS. This example can be generalized to most requirements that can be identified as a single cross-cutting concern. In consequence, mechanisms in this category allow a separation of concerns and provide a design abstraction in the development process of such applications. Typical concerns that could leverage this approach are the Laws of Physics (Laws of Mechanics, Principles of Thermodynamics, etc.), when they are required in a simulation.

3.1.2. *Pervasive Applications*

In pervasive applications, significant efforts are devoted to fully distributed systems. Some functions are however difficult to decentralize for performance or technical reasons, so that centralized mechanisms are required. For instance, sensor network applications based on the Agilla agent framework feature a communication environment on every node, each of them becoming a 'mini-MAS' (Agilla, 2006). Such a node environment consists of a tuple space, i.e. an instance of environment-mediated interaction channel, that Agilla agents can use to exchange messages. When agents need to move over the network, their environment changes to the tuple space of the destination node (with appropriate coordination between the environments of the mini-MAS).

Such a centralized mechanism on each node is an appropriate design abstraction in the case of sensor networks, where agents are kept simple due to the availability of limited computing power. The mechanism encapsulates the interaction logics so that agents only contain code relative to their functional requirements. Consequently, agents can focus on high-level coordination issues, while delegating low-level ones to their supporting environment. This property is a typical benefit of seeing the environment as an infrastructure of the system, thus realizing a clear separation of concerns.

3.1.3. *Virtual Societies*

The environment has been implicitly exploited in the design of agent societies. Such societies are for instance auction systems (Chavez and Maes, 1996) and other electronic market places (Tsvetovatyy et al., 1997; Karacapilidis and Moraitis, 2001) proposed over the past decade. Although the environment is not a major concern in most of these systems, the current perspective on this legacy work sets forth specific traits of the environment. For example, the AuctionBot was a running agent platform where users could create an auction and welcome client agents to participate in. The platform provides virtual auction rooms to facilitate agent interactions and auction runs. These 'rooms' could be thought of as environment-mediated interaction channels. Agents would enter those rooms in a publish-subscribe fashion. Although the environment was not considered then as a first-class entity, it is arguably present to various extents in each of these platforms as shown with the room analogy. The mapping between the room requirement and an environment mechanism seems natural and reasonable.

In addition, the combination of environment-mediated interaction channels and resource and context managers could help designing the environment for such systems. An example of this compound use of these mechanisms is an agent-based conference management system, where the focus is however more on the resource and context manager mechanism (Zambonelli et al., 2003).

Beyond the possible applications in legacy systems, environment mechanisms can serve in the design of electronic institutions. Laws and norms are global and independent characteristics of such systems. Current approaches like AMELI can be thought of as centralized interaction mechanisms to ensure that agents respect norms in their interactions (Esteva et al., 2004). The functioning of the institution can be hidden from agents by providing appropriate mechanisms for a separation between agents internal logics and regulation of the system.

3.2. Decentralized Interaction Mediation

Decentralized mechanisms have applications in various domains, where distribution is wanted or imposed. The majority relies on the overlay network mechanism, but the usage of the decentralized synchronization mechanism is fully represented.

3.2.1. *Simulation*

Successful experiments with decentralized interaction mediation mechanisms were conducted over the past decades to reproduce behaviors found in nature, notably in ethology for ants and termites (Drogoul and Ferber, 1992; Parunak, 1997), swarm intelligence (Bonabeau et al., 1999), and artificial markets (Eymann et al., 1998; Guyot et al., 2005). Although the environment was not the main focus of these experiments, it appears essential in each setting of the agent activity. In the case of the behavioristic reproduction of ant or termite colonies, agents rely on pheromone infrastructures that evolve in complex patterns maintained by the environment (diffusion and evaporation of the pheromone information). Swarms exploit the environment to model the external factors that constrain and guide agent behaviors, such as the physical topology of a building in emergency drills (Nakanishi et al., 2003). In artificial markets, the environment provides renewable resources such as wood or coffee (Eymann et al., 1998; Guyot et al., 2005).

In such simulations, the overlay network mechanism allows to describe and maintain a topology that provides agents with various interaction means in the simulation. This mechanism was also combined with environment-mediated interaction channels to model complex social behaviors (Platon, 2006), and with the resource and context manager when agents are to exploit specific facilities in the environment such as wood or coffee.

The main contribution of the environment in the above applications is the coordination of agents in a decentralized way. The mechanism permits then a separate design of the coordination logics that follows the structure of the system. In particular, one expected property is to avoid performance collapses due to a centralized approach.

3.2.2. *Pervasive Applications*

The Agilla framework introduced in section 3.1.2 provides decentralized mechanisms to handle the interactions between the environments of the 'mini-MAS' of a sensor network. These mechanisms allow agents to communicate and move seamlessly from one MAS to another in the Agilla infrastructure by coordinating the different environments transparently.

Another representative application deals with the robot industry. This industry has been successful for long with centralized approaches where master-servers command remote robots. In order to cope with the soaring lack of flexibility and reliability of such multi-robot systems, this industry has been deploying new architectures where robots are thought of as peers, so that they can perform their tasks in a decentralized fashion. The case of the Automatic Guided Vehicles (AGV) relies on an explicit decentralized environment model to deal with the interactions between the robots (Weyns et al., 2005c). Each AGV has a software layer that consists of the agent that 'drives' the robot and an environment that represents a computational counterpart of local physical information (its hull, position from sensors, robots in the vicinity, etc.). The environment part of each robot is responsible for updating the local information thus allowing the agent to reason and avoid possible collisions with other robots. Such an environment is similar to the model of Agilla. In addition, the environment of each robot is responsible for real-time synchronization with other pieces of environments in the vicinity to update data that matter locally. The environment is therefore up-to-date and synchronized against surrounding environments only. This reduction of synchronization to the vicinity allows agents to focus on the relevant area for their calculation and reduces the computational cost in communication.

The overlay network mechanism represents the functionalities required for the interaction among the different elements of the above architecture. In the particular case of Agilla, the mechanism is a set of algorithms to forward messages between tuple spaces and update the state information of the two mini-MAS involved in the migration of an agent. In addition, the AGV case shows a usage of decentralized synchronization mechanisms. The real-time constraint of this application makes difficult and costly to synchronize all AGVs with a centralized mechanism. The decentralized approach is economical and, most importantly, allows agents to focus on relevant data.

3.2.3. *Virtual Societies*

In the video game domain, one notable example of virtual society is the popular video game 'The Sims' (Electronic Arts, 2006). This game is a pseudo real-time application where agents live (some of them guided by players). The environment consists of places to visit and objects that agents can use. The environment represents elements that surround agents in the game and manages the interactions among agents and objects. This functionality can rely on an instance of overlay network. One distinguishing particularity of this approach is that this mechanism is composed with an overlay data structure to automate some agent interactions (see section 3.4.3).

3.3. Centralized Resource and Context Management

Many applications provide access to resource and contextual information in a centralized fashion to ease the underlying management algorithms. Most of them rely on the resource and context manager, but also on the notification of context events mechanism in some settings.

3.3.1. *Simulation*

The SimCafe and SimCommod simulations have an explicit representation of the environment resources such as cafe and wood respectively (Guyot et al., 2005; Guyot et al., 2006). A purpose of the simulation is to study the interaction patterns among agents that involve the usage of the resources. In these simulations, environmental resources are managed by a specific component that determines some relevant dynamics. For example, SimCommod simulates the evolution of the wood resource as a cellular automaton.

The resource and context management in these simulations are centralized and leverage instances of the resource and context manager mechanism. The main advantage of this approach is the clear separation between agents and the dynamics in the environment. In particular, the environments of SimCafe and SimCommod have many common characteristics, so that the separation of concerns permitted to reuse much of the model and underlying implementation.

3.3.2. *Pervasive Applications*

The Agilla environment (see section 3.1.2) provides direct access to the sensors of the node (awareness of physical properties) and maintains a neighbor list that is updated according to agent mobility (awareness of other agents). The management of the neighbor list is totally transparent to application agents, as a service provided by the environment. This service appears as an instance of the resource and context management mechanism. The delegation of the neighbor list management to the environment is justified mainly by the limited memory of sensor network nodes. Instead of having agents maintaining individual neighbor lists, a single list is maintained for each node (individual lists would be redundant copies), which significantly reduces the memory requirements.

The resource and context manager participates in the environment to screen the complexity of interacting with low-level resource details. This property is another aspect of the separation of concerns provided by the mechanism. In addition, it provides a design abstraction when building application agents, since interactions are supported by the underlying framework.

3.3.3. *Virtual Societies*

Recent work exploit centralized resource and context management mechanism in electronic markets to augment the interaction capabilities of market actors. In addition to usual interaction protocols, agents can overhear conversations of others, as allowed in some markets (bazaar type), and they can also be annotated with relevant information such as the type of items they are searching in the market (Platon et al., 2005b; Platon, 2006).

The overhearing mechanism is executed by the environment to enforce the phenomenon in agent communications (some malicious agents could hide some messages). In the annotation approach, the environment is in charge of two responsibilities that are separate concerns: Publication and regulation of the annotations in the system. The publication automated by the environment allows seller agents to be notified about the presence of potential buyers, so that they can initiate sell attempts. The regulation by the environment prevents the spread of fake annotations that would mislead trading agents and harm the market. For example, a market can authorize the trade of some items only. Buyers cannot trade other types of items as the environment does not diffuse the corresponding tag informations to sellers.

Such applications rely on the notification of context events to model an appropriate solution for the diffusion of overheard and annotation messages. In addition, the resource and context manager is also used to support the regulation functionality of the environment, by enforcing access rights to atoms in the environment and publishing information independently from agents.

3.4. Decentralized Resource and Context Management

Decentralized management of resource and context rely on the overlay data structure mechanism. The research community has been creative and we present in this section the main applications.

3.4.1. *Simulation*

Overlay data structure mechanisms are especially used in simulation to reproduce complex coordinations in the motion of agents, as introduced in section **??**. Most achievements can be observed in MABS for reactive agents with the pheromone and coordination field abstractions. Typically, the mechanism can be identified in agent-based simulations of the behavior of social insects in ethology and the coordinated motion of agents in video games (Drogoul and Ferber, 1992; Mamei and Zambonelli, 2004a). In both cases, the mechanism provides agents with access to a rich data structure. The access rights depend on the agent context, so that ant agents can, for example, only sense pheromones in their vicinity. Farther pheromones do not influence the behaviors of ants until they are in the same vicinity.

3.4.2. *Pervasive Applications*

In an application of agent technologies, visitors of a museum were endowed with a PDA that hosts an agent to guide them to relevant spots and to coordinate the visit, for instance to avoid crowd accumulation (Mamei et al., 2004; Mamei and Zambonelli, 2004b). The coordination mechanism relies on the coordination field approach (Mamei and Zambonelli, 2004a). In the museum, agents emit repulsive gradient fields, whereas rooms emit attractive fields. Agents then tend to lead visitors away from each other (repulsion), while guiding them to different rooms (attraction).

The museum application illustrates the use of the overlay data structure for motion coordination in a pervasive computing context. In addition, the experiments show in a concrete domain how the computational environment practically uncouples the application logics for the interface agent from the coordination logics, which is a desired property for the development of industrial systems and services.

3.4.3. *Virtual Societies*

In the video game 'The Sims' introduced in section 3.2.3, the simulation of virtual life has a significant impact on the quality of the game for the player. The Sims exploits sorts of coordination fields named 'happiness landscapes' that span over a computational environment to represent the physical topology in which characters live. The landscapes drive the movements of non-player characters and they demonstrate advanced self-organization capabilities to create 'believable' agent behaviors. For instance, if a character is hungry, it perceives and follows a happiness landscape whose peaks correspond to places where food can be found, i.e. a fridge. After having eaten, a new landscape is followed depending on its needs. Coordination among multiple agents is achieved by having agents react to the same landscape. The landscapes proposed in The Sims is an instance of the overlay data structure mechanism. In the context of the game, the main advantage of the mechanism seems to put at the same abstraction level agents and items of interest (e.g. the fridge). This design abstraction may serve application designers to conceive new games based on the same principles easily.

In another domain, overlay data structures serve to coordinate agents in computational systems. One illustration is in agent-based web-sites with recommendation systems (Bandini et al., 2005). Visitors of the web-site become the application agents and the hyperlink structure of the site is considered as the topology of the environment. Registered users are tracked on the environment to identify emerging patterns from the repeated use of links of the site. The patterns form paths in the environment that become recommendations to the registered user and, to lower extent, simple visitors. Overlay data structures yield major techniques to track agents in the system and to adapt the environment contextual information accordingly. In combination, the resource and context manager is appropriate whenever centralized mechanisms can be used (for instance if one node of the web-site points to a database), but this point of view remains an extension of the work presented above.

4. Research Directions and Opportunities

The mechanisms surveyed in this article come from the agent literature and where already exploited extensively. In this section, we determine possible application opportunities and potential research to be conducted on mechanisms for environments.

4.1. Application Opportunities

We identified a number of domains that can exploit the environment and its mechanisms in a seemingly straightforward manner. First, normative systems impose norms on the behavior of agents, usually in a social agency metaphor (Vázquez-Salceda, 2004). Vázquez-Salceda refers to normative systems as electronic institutions: ‘An e-institution is a safe environment mediating in the interaction of agents’ (Vázquez-Salceda, 2006). In fact, the approach on e-institution is moving toward the introduction of an explicit regulating entity in MAS, whose functions naturally meet the notion of environment with a social level of support (Weyns et al., 2006). One opportunity of the environment for e-institutions is then for the regulation of agent activities, as it provides institution designers with an appropriate abstraction to deal with the regulating and normative aspects from design to runtime. In addition, the environment can also serve to regulate the interactions with external resources. This functionality is usually dealt with in an ad hoc way, and the environment can provide an adequate approach for integrating it in normative systems.

Interface MAS generalize the work done on animated and conversational agents with multi-agent settings and they also pertain to agent societies. Embodied Conversational Agents (ECA) appeared in the early 1990s as a new paradigm for improving human-computer interaction (Cassel et al., 2000) by the combination of several modalities, such as speech, gesture, and facial expressions. In practice, ECA are used within end-user applications, and recent research has raised issues of interactions among ECA (Ishida, 2002). Although research on ECA has focused on multi-modal interactions with the human-user for a long time, the results usually cannot be translated to interactions among ECA. Agent communication languages focus on Speech Acts and they are not always adequate to describe other modes of interactions. A computational environment has allowed to enrich possible interaction types among software agents, and the application of recent work to ECA based on the environment is expected to contribute significantly to the field (Tummolini et al., 2004; Platon et al., 2005a).

4.2. Research directions

Mechanisms appear as key enablers of the environment idea, allowing a transition from concepts ('what') to engineering ('how'). Mechanisms can become incentives to exploit the environment in MAS by providing abstractions. Yet, the mechanisms in this article originate in practical MAS applications that do not completely cover the research in MAS. Other research areas are to be explored for the potential discovery of new mechanisms and the revision of the current classification.

In this regard, future research could fruitfully address the following directions.

1. The survey work initiated in this article should be continued and refined with ongoing research activities. Similarly to the positive effect of design patterns on object-oriented programming, it is promising to provide developers with a set of mechanisms with clear guidelines on how to use them. To this end, two research directions can be relevant. On the one hand, it would be interesting to support designers in selecting suitable mechanisms according to the application requirements at hand. On the other hand, further studies can lead to engineering approaches on how to consistently compose existing mechanisms.

2. Novel mechanisms addressing under-investigated areas could be beneficial to the design of MAS. For example, in the specific case of decentralized interaction mediation, new mechanisms allowing agents to better structure and analyze a topology could lead to relevant results, especially for the role of the environment in the case of social interactions (Hales and Edmonds, 2005). Besides,

the aforementioned support for designers could help to determine innovative mechanisms in direct relation to the existing ones.

3. Most of the currently proposed mechanisms are at a low-level, in the sense that they enable interaction, provide context information, and wrap resources. Agents are expected to be at a higher level of abstraction and it would be interesting to search for mechanisms supporting agents in achieving their goals. Promising mechanisms at a higher level of abstraction may be the ones to hide the complexity of low-level issues, to regulate agent activities, and perhaps to provide mechanisms for trust management. Some other research endeavors also introduce potential extension of existing low-level mechanisms, such as the 'cognitive stigmergy' (Susi and Ziemke, 2002; Parunak, 2005).

5. Summary and Conclusions

Mechanisms are presented in this article to describe design element that deal with the responsibilities attributed to the environment in Multi-Agent Systems. The presentation is grounded on representative achievements in the MAS community and it is expected that this initial work will foster further endeavors to develop the potential of mechanisms in the design of environments.

All in all, this article allows to identify potential research directions and opportunities, most notably for the development of novel mechanisms and applications related to the growing work on coordination, regulation, and interface agents.

To date, the identification of mechanisms of environment results in two main consequences for the agent community. First, mechanisms reveal the internals of the environment exploited in current work and potential research directions for novel functionalities. The mechanism approach allows to decompose the environment into design idioms that are reusable and composable as the design patterns in object-oriented computing (Gamma et al., 1999). Second, the survey on environment applications demonstrates the existence of unexplored combinations of mechanisms that could impact the development of MAS. As a result, mechanisms provide research directions that can benefit to theory and practice. Most importantly, the identification of mechanisms bridges the research on responsibilities of the environment to the subsequent issues in engineering.

References

AgentCities: Accessed in March 2006, ‘AgentCities web site’. `http://www.agentcities.org/`.

Agilla: Accessed in March 2006, ‘Agilla, A Mobile Agent Middleware for Wireless Sensor Networks’. `http://www.cs.wustl.edu/mobilab/projects/agilla/`.

Balbo, F. and S. Pinson: 2001, ‘Toward a Multi-agent Modelling Approach for Urban Public Transportation Systems’. In: *Engineering Societies in the Agents World II*. Prague, Czech Republic: Springer.

Bandini, S., S. Manzoni, and G. Vizzari: 2005, ‘Web Sites as Agents’ Environments: General Framework and Applications.’. In (Weyns et al., 2005a), Springer.

Bonabeau, E., M. Dorigo, and G. Theraulaz: 1999, *Swarm Intelligence. From Natural to Artificial Systems*. Oxford University Press.

Brueckner, S.: 2000, ‘Return from the Ant — Synthetic Ecosystems for Manufacturing Control’. Ph.D. thesis, Humboldt University, Berlin, Germany.

Busetta, P., A. Donà, and M. Nori: 2002, ‘Channelled Multicast for Group Communications’. In: *Autonomous Agents & Multiagent Systems*. pp. 1280–1287, ACM.

Cabri, G., L. Leonardi, and F. Zambonelli: 2002, ‘Engineering Mobile Agent Applications via Context-dependent Coordination’. *IEEE Transactions on Software Engineering* **28**(11), 1040 – 1056.

Carzaniga, A., D. Rosenblum, and A. Wolf: 2001, ‘Design and Evaluation of a Wide-Area Event Notification Service’. *ACM Transactions on Computer Systems* **19**(3), 332 – 383.

Cassel, J., J. Sullivan, S. Prevost, and E. Churchill: 2000, *Embodied Conversational Agents*. MIT Press.

Chavez, A. and P. Maes: 1996, ‘Kasbah: An Agent Marketplace for Buying and Selling Goods’. In: *International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*.

Cugola, G., A. Fuggetta, and E. D. Nitto: 2001, ‘The JEDI Event-based Infrastructure and its Application to the Development of the OPSS WFMS’. *IEEE Transactions on Software Engineering* **27**(9), 827 – 850.

Decker, K. S.: 1995, ‘Environment Centered Analysis and Design of Coordination-Mechanisms’. Ph.D. thesis, University of Massachusetts, Amherst, USA.

Demazeau, Y.: 1995, ‘From interactions to collective behaviour in agent-based systems’. In: *European Conference on Cognitive Sciences*.

Dignum, F., S. Kraus, and M. Singh (eds.): 2005, ‘Fourth International Joint Conference on Autonomous Agents & Multiagent Systems, July 25-29, 2005, Utrecht, The Netherlands’. ACM.

Drogoul, A. and J. Ferber: 1992, ‘Multi-Agent Simulation as a Tool for Modeling Societies: Application to Social Differentiation in Ant Colonies.’. In: C. Castelfranchi and E. Werner (eds.): *MAAMAW*, Vol. 830 of *Lecture Notes in Computer Science*. pp. 3–23, Springer.

Electronic Arts: Accessed in March 2006, ‘The Sims web site.’. `http://thesims.ea.com`.

Esteva, M., B. Rosell, J. A. Rodríguez-Aguilar, and J. L. Arcos: 2004, ‘AMELI: An Agent-Based Middleware for Electronic Institutions.’. In: *AAMAS*. pp. 236–243, IEEE Computer Society.

Eugster, P., P. Felber, R. Guerraoui, and A. Kermarrec: 2003, ‘The Many Faces of Publish/Subscribe’. *ACM Computing Surveys* **35**(2), 114 – 131.

Eymann, T., B. Padovan, and D. Schoder: 1998, 'Simulating Value Chain Coordination with Artificial Life Agents.'. In: *International Conference on Multi-Agent Systems*. pp. 423–424, IEEE Computer Society.

Ferber, J.: 1999, *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley.

Ferber, J. and J.-P. Müller: 1996, 'Influences and Reaction: a Model of Situated Multiagent Systems'. In: *Second International Conference on Multi-agent Systems*. AAAI.

Freeman, E., S. Hupfer, and K. Arnold: 1999, *JavaSpaces Principles, Patterns, and Practice*. Addison-Wesley.

Gamma, E., R. Helm, R. Johnson, and J. Vlissides: 1999, *Design Patterns*. Addison-Wesley.

Gasser, L.: 2000, 'MAS Infrastructure: Definitions, Needs and Prospects.'. In: T. Wagner and O. F. Rana (eds.): *Agents Workshop on Infrastructure for Multi-Agent Systems*, Vol. 1887 of *Lecture Notes in Computer Science*. pp. 1–11, Springer.

Gelernter, D. and N. Carriero: 1992, 'Coordination Languages and Their Significance'. *Communication of the ACM* **35**(2), 96 – 107.

Guyot, P., A. Drogoul, and S. Honiden: 2006, 'Power and Negotiation: Lessons from Agent-Based Participatory Simulations.'. In: *Autonomous Agents & Multiagent Systems*. pp. 27–33, ACM.

Guyot, P., A. Drogoul, and C. Lemaître: 2005, 'Using emergence in participatory simulations to design multi-agent systems.'. In (Dignum et al., 2005), pp. 199–203, ACM.

Hales, D. and B. Edmonds: 2005, 'Applying a socially-inspired technique (tags) to improve cooperation in P2P Networks'. *IEEE Transactions in Systems, Man and Cybernetics — Part A: Systems and Humans* **35**(3), 385–395.

Helleboogh, A., G. Vizzari, and F. Michel: 2006, 'Multi-Agent Modeling and Simulation: the Role of the Environment'. In (Sycara and Wooldridge, 2006).

Ishida, T.: 2002, 'Q: A Scenario Description Language for Interactive Agents.'. *IEEE Computer* **35**(11), 42–47.

Johanson, B. and A. Fox: 2002, 'The Event Heap: A Coordination Infrastructure for Interactive Workspaces'. In: *Proceedings of the Workshop on Mobile Computer Systems and Applications*. Callicoon, New York, USA: IEEE CS Press.

Kakas, A. C., P. Mancarella, F. Sadri, K. Stathis, and F. Toni: 2004, 'The KGP Model of Agency.'. In: R. L. de Mántaras and L. Saitta (eds.): *European Conference on Artificial Intelligence*. pp. 33–37, IOS.

Karacapilidis, N. I. and P. Moraitis: 2001, 'Intelligent agents for an artificial market system.'. In: *Agents*. pp. 592–599.

MABS: 1998, 2000, 2002, 2003, 2004, 'Multi-Agent Systems and Agent-Based Simulation, International Workshop Series', Vol. 1534, 1979, 2581, 2927, 3415 of *LNCS*. Springer.

Mamei, M. and F. Zambonelli: 2004a, 'Motion Coordination in the Quake 3 Arena Environment: A Field-Based Approach.'. In (Weyns et al., 2004), pp. 264–278, Springer.

Mamei, M. and F. Zambonelli: 2004b, 'Programming Pervasive and Mobile Computing Applications with the TOTA Middleware'. In: *Proceedings of the International Conference On Pervasive Computing (Percom)*. Orlando, Florida, USA: IEEE CS Press.

Mamei, M., F. Zambonelli, and L. Leonardi: 2004, 'Co-Fields: A Physically Inspired Approach to Distributed Motion Coordination'. *IEEE Pervasive Computing* **3**(2), 52 – 61.

Menezes, R. and R. Tolksdorf: 2003, 'A New Approach to Scalable Linda-systems Based on Swarms'. In: *Proceedings of the Symposium on Applied Computer*. Orlando, Florida, USA: ACM Press.

Nakanishi, H., S. Koizumi, and T. Ishida: 2003, 'Virtual Cities for Real-World Crisis Management.'. In: P. V. den Besselaar and S. Koizumi (eds.): *Digital Cities*, Vol. 3081 of *Lecture Notes in Computer Science*. pp. 204–216, Springer.

O'Hare, M. R. . G. M.: 1991, 'Blackboard Systems: A Survey of their Application.'. *Artificial Intelligence Review* (May).

Omicini, A. and F. Zambonelli: 1998, 'TuCSoN: a Coordination model for Mobile Information Agents'. In: D. G. Schwartz, M. Divitini, and T. Brasethvik (eds.): *First International Workshop on Innovative Internet Information Systems*. Pisa, Italy, pp. 177–187, IDI – NTNU, Trondheim (Norway).

OpenNet: Accessed in March 2006, 'openNet web site'. `http://x-opennet.org/`.

Parunak, H. V. D.: 1997, "Go to the Ant': Engineering Principles from Natural Multi-Agent Systems'. *Annals of Operation Research* **75**, 69–101.

Parunak, H. V. D.: 2005, 'Expert Assessment of Human-Human Stigmergy'. Analysis for the Canadian Defence Organization, Altarum Institute, Ann Arbor, Michigan.

Parunak, H. V. D., S. Brueckner, M. Fleischer, and J. Odell: 2003, 'A Design Taxonomy of Multi-agent Interactions.'. In: P. Giorgini, J. P. Müller, and J. Odell (eds.): *Workshop on Agent-Oriented Software Engineering*, Vol. 2935 of *Lecture Notes in Computer Science*. pp. 123–137, Springer.

Parunak, H. V. D., S. Brueckner, and J. A. Sauter: 2004, 'Digital Pheromones for Coordination of Unmanned Vehicles.'. In (Weyns et al., 2004), pp. 246–263, Springer.

Platon, E.: 2006, 'Smart Environment for Smarter Agents in E-markets'. In: *19th Conference of the Florida Artificial Intelligence Research Society*. pp. 176–177.

Platon, E., N. Sabouret, and S. Honiden: 2005a, 'Environment Support for Oversensing'. In: M.-P. Gleizes, G. A. Kaminka, and S. Ossowski (eds.): *Proceedings of the European Workshop on Multi-Agent Systems 2005*.

Platon, E., N. Sabouret, and S. Honiden: 2005b, 'Overhearing and Direct Interactions: Point of View of an Active Environment, a Preliminary Study'. In (Weyns et al., 2005a), Springer.

Ratnasamy, S. and B. Karp: 2002, 'GHT: A Geographic Hash Table for Data-Centric Storage'. In: *Proceedings of the International Workshop on Wireless Sensor Networks and Applications*. Atlanta, Georgia, USA: ACM Press.

Ratsanamy, S., P. Francis, M. Handley, and R. Karp: 2001, 'A Scalable Content-Addressable Network'. In: *Proceedings of the SIGCOMM Conference*. San Diego, California, USA: ACM Press.

RFID standard: Accessed in March 2006, 'RFID on Wikipedia'. `http://en.wikipedia.org/wiki/RFID`.

Rowstron, A. and P. Druschel: 2001, 'Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems'. In: *Proceedings of the Conference on Distributed Systems Platforms*. Heidelberg, Germany: ACM Press.

Russell, S. and P. Norvig: Edition 2003, *Artificial Intelligence: A Modern Approach*. Prentice Hall.

Sichman, J. S., R. Conte, C. Castelfranchi, and Y. Demazeau: 1994, 'A Social Reasoning Mechanism Based On Dependence Networks.'. In: *European Conference on Artificial Intelligence*. pp. 188–192.

Susi, T. and T. Ziemke: 2002, 'Social Cognition, Artefacts, and Stigmergy: A Comparative Analysis of Theoretical Frameworks for the Understanding of Artefact-mediated Collaborative Activity'. *Cognitive Systems Research* **2**(4), 273–290.

Sycara, K. and M. Wooldridge (eds.): 2006, *Autonomous Agents and Multi-Agent Systems: Special Issue on Environment for Multi-Agent Systems*. Springer.

Tsvetovatyy, M., M. Gini, and Z. Mobasher, B.and Wieckowski: 1997, 'MAGMA: An Agent-Based Virtual Market for Electronic Commerce'. *Journal of Applied Artificial Intelligence* **11**(6), 501–523.

Tummolini, L., C. Castelfranchi, A. Ricci, M. Viroli, and A. Omicini: 2004, '"Exhibitionists" and "Voyeurs" Do It Better: A Shared Environment for Flexible Coordination with Tacit Messages.'. In (Weyns et al., 2004), pp. 215–231, Springer.

Valckenaers, P., C. Sierra, and J. Sauter: 2006, 'Applications of Environments for Multi-Agent Systems'. In (Sycara and Wooldridge, 2006).

Vázquez-Salceda, J.: 2004, *The Role of Norms and Electronic Institutions in Multi-Agent Systems, The HARMONIA Framework*, Whitestein Series in Software Agent Technologies. Springer.

Vázquez-Salceda, J.: Accessed in May 2006, 'From Human Regulations to Regulated Software Agents' Behaviour.'. Research report accessible at `http://www.lsi.upc.edu/~jvazquez/docs/WSbcn05.pdf`.

Weyns, D. and T. Holvoet: 2004, 'A Formal Model for Situated Multi-Agent Systems.'. *Fundam. Inform.* **63**(2–3), 125–158.

Weyns, D., A. Omicini, and J. Odell: 2006, 'Environment, First-Order Abstraction in Multiagent Systems'. In (Sycara and Wooldridge, 2006).

Weyns, D., H. V. D. Parunak, and F. Michel (eds.): 2004, 'Environments for Multi-Agent Systems, First International Workshop, E4MAS 2004, New York, NY, USA, July 19, 2004, Revised Selected Papers', Vol. 3374 of *Lecture Notes in Computer Science*. Springer.

Weyns, D., H. V. D. Parunak, and F. Michel (eds.): 2005a, 'Environments for Multi-Agent Systems, Second International Workshop, E4MAS 2005, Utrecht, The Netherlands, July 26, 2005, Revised Selected Papers', Vol. 3830 of *Lecture Notes in Computer Science*. Springer.

Weyns, D., H. V. D. Parunak, F. Michel, T. Holvoet, and J. Ferber: 2004, 'Environments for Multiagent Systems, State-of-the-Art and Research Challenges'. In (Weyns et al., 2004).

Weyns, D., K. Schelfthout, and T. Holvoet: 2005b, 'Exploiting a Virtual Environment in a Real-World Application'. In (Weyns et al., 2005a).

Weyns, D., K. Schelfthout, T. Holvoet, and T. Lefever: 2005c, 'Decentralized control of E'GV transportation systems.'. In (Dignum et al., 2005), pp. 67–74, ACM.

Zambonelli, F., N. R. Jennings, and M. Wooldridge: 2003, 'Developing multiagent systems: The Gaia methodology.'. *ACM Trans. Softw. Eng. Methodol.* **12**(3), 317–370.

Zhao, F. and L. Guibas: 2004, *Wireless Sensor Networks: An Information Processing Approach*. Elsevier, Morgan-Kaufmann.