Coordination in Multi-Agent Systems: A Declarative Approach using Coordination Spaces

Ante Vilenica, Alexander Pokahr, Lars Braubach, Winfried Lamersdorf
Distributed Systems and Information Systems, University of Hamburg
{vilenica}@informatik.uni-hamburg.de

Jan Sudeikat, Wolfgang Renz
Multimedia Systems Laboratory, Hamburg University of Applied Sciences
{sudeikat,wr}@informatik.haw-hamburg.de

Abstract

Coordination is an important aspect of multi-agent systems (MAS). Although being decisive for many MAS applications, it is rarely modelled and implemented explicitly. Instead, coordination is realized implicitly and hard wired among other agent functionalities. Therefore, this work aims at providing an approach towards explicit coordination in MAS in which developers can focus more on *what* and not so much on *how* to coordinate. More specifically, this paper presents both a declarative approach to modelling coordination in MAS and a reference implementation for an agent framework. In this implementation, so-called *coordination spaces* that are part of the agent environment, process the declarative description of the coordination and thus release the developer from programming coordination manually. Consequently, this approach to declarative coordination offers benefits like reusability and interoperability of coordination aspects of MAS as well as the possibility to coordinate agents with heterogeneous architectures.

1 Introduction

Multi-agent systems (MAS) are a well known approach to model and implement complex distributed systems and applications. Due to the naturally decentralized architecture, this paradigm provides appropriate concepts for realizing systems that offer inherently non-functional requirements such as scalability, robustness and failure tolerance. However, in order to ensure proper system functionality of MAS on a global level the local activities of agents need to be *coordinated.* Coordination is therefore one of the key aspects of MAS and can be defined as *the management of dependencies* [Malone and Crowston, 1994]. The basic idea of the work presented in this paper is that all aspects related to this management should be handled separately in MAS within a dedicated environment. This environment encapsulates all concerns related to coordination and enables an easy way to apply different coordination strategies without the need to change the agent functionality. Thus, in this paper an approach is proposed that advocates a clear separation between application functionality (handled by agents) and coordination (handled by the environment).

The concept of environments as part of MAS has been widely recognized. Agent definitions point out the occurrence of an environment where an agent is situated [Wooldridge and Jennings, 1995]. Moreover, researchers have argued to understand environments as first-class entities besides agents in MAS [Weyns *et al.*, 2005; Keil and Goldin, 2006] and have therefore pointed out the need for an explicit concept to handle concerns apart from the core agent functionalities. However, most agent development frameworks, such as JADE or Jack do not explicitly provide approaches for modelling environments as a separate entity within MAS. Even more, existing agent infrastructures often realize environment responsibilities implicitly [Viroli *et al.*, 2007]. Therefore, these approaches often intertwine coordination and functionality.

The work presented here proposes concepts to handle coordination in MAS explicitly - while treating environments as first-class entities. Therefore, a model is presented that enables a declarative description of coordination entities and mechanisms in MAS. This description can be automatically processed within the environment and, therefore, relieves developers from implementing the coordination functionality manually. Within this work, such dedicated environments for coordination are called *coordination spaces.* A beneficial consequence of this approach is that aspects related to coordination can be modularized and therefore exchanged easily as well.

This paper is structured as follows. The next Section discusses related work; Section 3 describes the concept of coordination spaces in general whereas Section 4 presents the implementation into an existing agent framework as well as a case study before Section 5 concludes the paper.

2 Related Work

Coordination among agents is about managing their activities. In general, the type of coordination can be differentiated whether it is realized by direct or indirect interaction. Due to the specific focus in this paper only approaches concerning indirect interaction will be treated in the following. Indirect interaction has proven to be suitable for coordination because it decouples the identity of the interaction actors *(anonymity)* as well as the time *(asynchrony)* and space *(locality)* of interaction [Keil and Goldin, 2006]. In order to systematize the work the approaches have been broadly classified into the categories of *integrated agent environments*, *generic environment infrastructures* and *specialized coordination frameworks*.

Integrated agent environments consider environments as part of a MAS and intertwine both. Such an approach is followed by most agent simulation toolkits like NetLogo[1] and SimSeSAm[2]. In many cases these toolkits offer an explicit environment for agents, in which they can interact indirectly by e.g. positioning objects on specific locations. Additionally, often environments also provide ready-to-use functionalities for sensing the local environment or for initiating environmental processes like diffusion. Such approaches foster the use of agent coordination via the environment and support it via prebuilt functionalities, but also restrict developers to these possibilities and make coordination an explicit part of the agent code. Furthermore, simulation toolkits do not offer a viable path for the transition of coordination logic to real applications, which is what we aim at.

In the area of generic environment infrastructures especially artifacts need to be considered. According to the A&A (agents and artifacts) metamodel [Ricci *et al.*, 2007] a MAS consists of agents and artifacts, which represent elements of the environment that can be used by agents. In order to support the indirect coordination via artifacts specific coordination artifacts can be devised [Ricci *et al.*, 2007]. Examples for such coordination artifacts include blackboards, maps or task schedulers. The artifact approach transfers the responsibility for coordination to the environment, which is similar to our approach. Nonetheless, it also explicitly expects the agents to use the artifacts for coordination. This means that coordination has to be part of the functional system design of the agents. Coordination spaces shall perform their activities invisible for the agents in order to understand coordination as a separate concern of the application development that is independent of the agent and environment design. Technically, artifacts could be used for the implementation of a coordination layer, e.g. to facilitate the usage in distributed systems.

Besides artifacts, also several specialized coordination approaches have been developed. Typically, these approaches aim at supporting one selected coordination mechanism. Examples range from ant environments like Anthill[3], over field-based mechanisms [Mamei and Zambonelli, 2005] to social norms used e.g. in 2APL[4]. Often such frameworks can be used not only in simulation but also as part of real applications. Nevertheless, their applicability is limited to the intended mechanism area, so that they cannot be considered as generic solution to a broad range of coordination problems.

[1]http://ccl.northwestern.edu/netlogo/
[2]http://www.simsesam.de/
[3]http://www.cs.unibo.it/projects/anthill
[4]http://www.cs.uu.nl/2apl

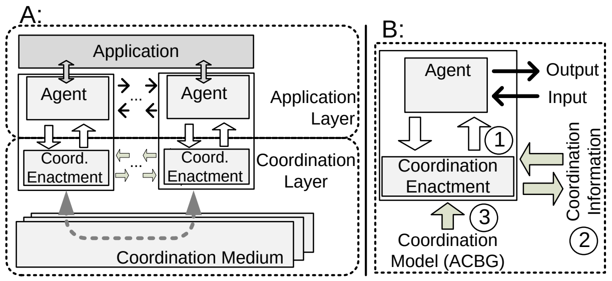


Figure 1: Coordination enactment architecture

To summarize, all described approaches do not provide an integrated approach that offers a clear model for different coordination mechanisms as well as an infrastructure supporting multiple mechanisms. The next sections will propose such an abstract model for coordination which also consists of a framework that is capable of executing theses models.

3 Purposeful Integration of Coordination into MAS

This section first introduces an existing architecture to integrate coordination into MAS. Then it shows how this approach can be extended in order to provide an integrated programming model for the integration of coordination in MAS.

3.1 Architecture for the Integration of Coordination

The software-technical utilization of coordination mechanisms as reusable design elements is addressed by a tailored programming model. This model provides a systemic modelling and configuration language (MASDynamics) [Sudeikat and Renz, 2009a] as well as a reference architecture [Sudeikat *et al.*, 2009] that enables the automated enactment of prescribed coordination models. The architecture separates the activities that are conceptually related to coordination of system entities from the entity functionality. Coordination models are treated as first class design elements that define application-independent patterns of agent interdependencies. The systematic integration of these patterns into application designs is discussed in [Sudeikat and Renz, 2009b].

An architectural blueprint for the integration of decentralised coordination mechanisms in MAS has been proposed in [Sudeikat *et al.*, 2009]. Figure 1(A) illustrates the conceptual layered structure. The application functionality is realized as an agent-based system within an *application layer*. Agents are part of the application design and serve as providers of distinct functionalities. The coordination of agent (inter-)actions is addressed by an underlying *coordination layer*. This layer contains two building blocks.

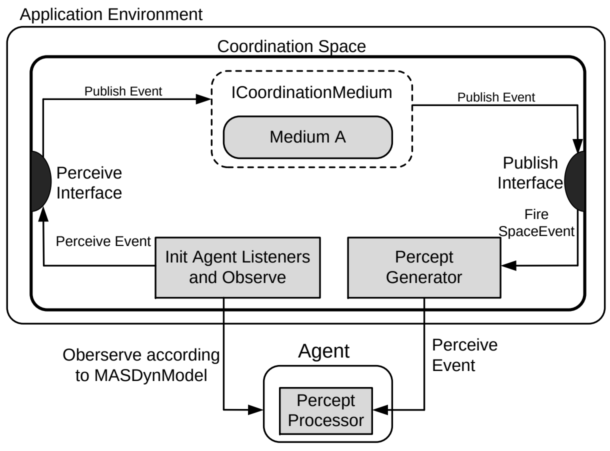


Figure 2: Conceptual model of coordination spaces

First, they provide a set of *coordination media* that provide direct or indirect coordination mechanisms. Media serve as infrastructures that allow agents to exchange application dependent information and contained mechanisms control the dynamics of information dissemination. Media are accessed by a generic publish/subscribe interface that decouples the participating agents. Secondly, the participating agents are equipped with *coordination endpoints* (cf. figure 1(B)) that automate coordination-related activities. Endpoints are able to observe and influence the agent activities (1). Via coordination media, these elements exchange information that is relevant for the coordination of activities (*coordination information*, 2). The local configuration of these activities, e.g. when to publish activities and how to process perceptions, are defined in a declarative, externalized *coordination model* (3).

The architecture provides a reference model to separate the coordination logic from the application logic. Coordination is declaratively prescribed to support the share and reuse of coordination pattern. This viewpoint is particularly inspired from the development of self-organizing applications where recurring coordination pattern are reused in differing application designs. Following the enactment architecture, the coordination is transparently enacted, i.e. the agent models are not modified but the coordination logic can be supplemented to an existing application design.

3.2 Using Coordination Spaces

This section presents the approach of coordination spaces that provide an integrated programming model for the integration of coordination into MAS. The aim of this concept is to support this integration with well-known principles from software engineering like reusability, interoperability and interchangeability. Furthermore, developers are relieved from implementing coordination manually since all aspects related to coordination can be described declaratively and on an implementation independent level. Also, coordination spaces offer a non-invasive approach, as agent code does not have to be changed in order to enable coordination.

The concept of coordination spaces has two important characteristics: First, it is based on the architecture and principles presented in Section 3.1. Secondly, it uses environments as first-class entities within MAS and therefore provides separate spaces within the environment that are explicitly in charge of coordination.

Figure 2 depicts the approach of coordination spaces. It shows that coordination spaces are dedicated places within the MAS environment that handle all aspects related to coordination. The space observes the agents via *listeners* and gets therefore a notification when an event occurs inside the agent. This event is published to the coordination medium which is in charge of processing the event. Agents are capable of receiving events via their percept generators and percept processors. Therefore, the task of coordination spaces consists mainly of three sub-tasks: observing agents, propagating events via the coordination medium and notifying agents about events. All these aspects of a coordination space are declaratively modelled and described within a separate configuration file. At system runtime, the coordination space initializes and performs the coordination according to the properties in the description file.

First, the space initializes listeners at the participating agents. These listeners observe the behaviour of the agent components, like its beliefs, goals, plans or whatever component might be of interest for the coordination. Whenever an observed element changes its state, the space observer sends an event to the *perceive interface* of the space, which again passes this event to the currently running *coordination medium*. This medium is in charge of processing and computing the coordination. The configuration of the medium determines how events are propagated to other agents. Like the listener and observer, the type and configuration of the coordination media is specified in a description file (cf. Figures 4 & 5). As the coordination medium is processing incoming events, it also needs a possibility to propagate the outcome of the processing to the participating agents. In order to do this, it uses the *publish interface* of the coordination space which causes the firing of a *space event*. Different agent architectures, e.g. that of the Belief-Desire-Intention-Model (BDI) [Rao and Georgeff, 1995] or just that of simple reactive agents, can define their own *percept generators* that are capable of receiving space events. Space generators notify the participating agents via their *percept processors* about changes in the application and these changes may affect the behaviour of the agents in turn. Therefore, the architecture of coordination spaces is inspired by the type of agent-environment relationship introduced in [Russell and Norvig, 2002].

Figure 2 also depicts that coordination spaces have three important interfaces that ensure a convenient exchange of coordination media as well as the independence of a concrete agent architecture. On the one hand, the *ICoordinationMedium* interface ensures that all coordination media implement certain basic methods and have a similar structure, which enables an easy replacement. On the other hand, the publish and perceive interfaces of the coordination space ensure that agents can propagate and receive coordination information. In order to support a certain agent architecture only one agent listener/observer as well as one percept generator/processor have to be implemented. These elements mediate between the agent and aforementioned interfaces of the coordination space. Therefore, it is even possible to use agents with heterogeneous architectures within the same MAS application.

The described approach of coordination spaces is guiding the process of managing the dependencies among agents. Therefore, it focuses on the question what to manage and not how. The developer is relieved from the *how aspects* because this concern is managed by the functions that are offered by the coordination space. The approach is non-invasive and does not require any changes at the agent side. It is transparent and provides a clear separation of concerns. There might also be more than one coordination space within one application. One space could be used to deal with spatial coordination aspects, like the movement in a 2D-Space, whereas another space might be in charge of organizational aspects, e.g. managing different agent roles, groups, positions [Gouaich and Michel, 2005; Weyns *et al.*, 2005].

4 Using Coordination Spaces within an Existing Agent Framework

This section will show the applicability of coordination spaces in practice. Therefore, selected elements of the coordination space are presented before they are then integrated into an existing agent framework. Thereafter, a case study is introduced that implements a MAS using a coordination space.

4.1 Configuring a Coordination Space

Every coordination space consists of a number of participating agents and at least one coordination medium. As mentioned in Section 3, coordination can be realized by direct or indirect interaction. Therefore, it has to be specified for each coordination medium which kind of interaction type is used. Since direct and indirect interactions differ completely from each other, a separate model is provided for each of them. Figure 3 shows exemplarily the configuration model for indirect, e.g. mediated, interaction. It consists of three main elements: *AgentConfiguration*, *ElementConfiguration* and *EnvironmentConfiguration*. The first element references the agent types that use the medium. The second element is the most important one since it offers the possibility to configure the behaviour of the elements that realize the indirect interaction in detail. Predominantly, these elements are some kind of *marker* like *(digital) pheromones* or *gradient fields* [DeWolf and Holvoet, 2006]. Markers are elements that can be produced, manipulated and deleted by agents and the environment. In addition,

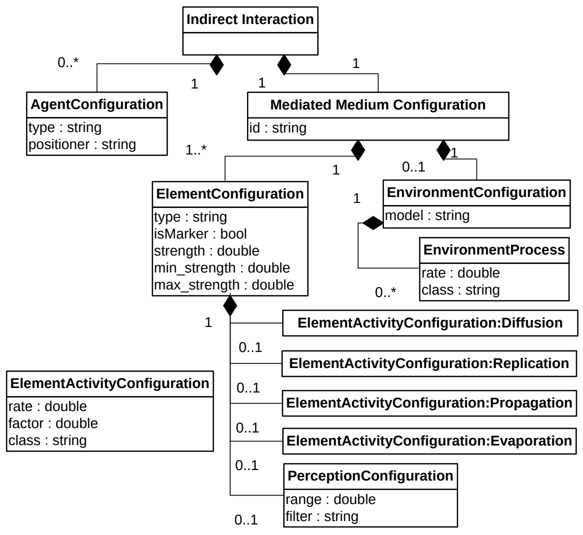


Figure 3: Abstract model of indirect interaction

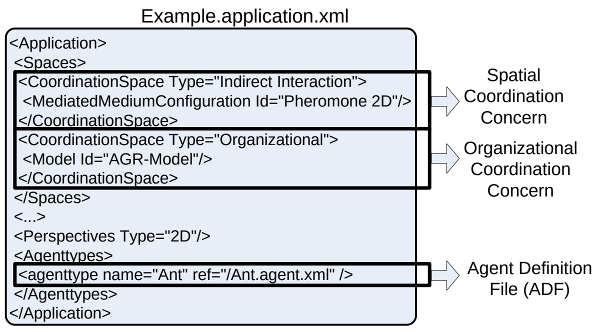
markers can be perceived by agents. Different types of markers can be defined and used within one coordination medium in order to realize different coordination aims. The model for markers allows to configure the way markers *diffuse, replicate, propagate* and *evaporate*. Each of these aspects can be simply specified with a value for the rate and factor as well as with a reference to a separate class if a customized implementation is needed. The third element (EnvironmentConfiguration) allows to customize and enhance the functionality of the space further by adding references to *EnvironmentProcesses*. These elements can manipulate the whole space. For example, they can be used to destroy all markers for which the strength falls below a limit or they can be used to produce new ones. Processes can be customized with a rate that specifies their execution frequency.

The configuration model (cf. figure 3) for coordination spaces has been specified as an XML-Schema.[5]

4.2 Framework Integration

The coordination space model has been implemented for the Jadex Agent Framework [Pokahr and Braubach, 2009a]. Jadex uses an application descriptor that defines a MAS in a declarative way [Pokahr and Braubach, 2009b]. Figure 4 shows exemplarily how such a file looks like. It focuses on only those elements which are of interest to understand the approach. Therefore, two aspects have to be pointed out: coordination spaces are recognized as first-class entities of MAS, and there can be multiple coordination spaces. The latter argument shows that even within coordination a separation of concerns can be applied meaningfully. Thus, organizational aspects are detached from spatial aspects and can be modelled independently. If the software developer wants to change the way spatial coordination is done, he does not have to deal with organizational aspects. The listing also

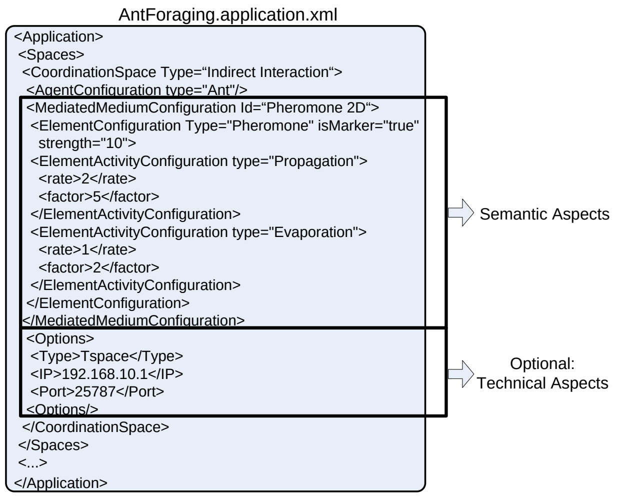
[5]http://vsis-www.informatik.uni-hamburg.de/projects/sodekovs/schemas


Example.application.xml

```xml
<Application>
 <Spaces>
 <CoordinationSpace Type="Indirect Interaction">
  <MediatedMediumConfiguration Id="Pheromone 2D"/>
 </CoordinationSpace>
 <CoordinationSpace Type="Organizational">
  <Model Id="AGR-Model"/>
 </CoordinationSpace>
 </Spaces>
 <...>
 <Perspectives Type="2D"/>
 <Agenttypes>
 <agenttype name="Ant" ref="/Ant.agent.xml" />
 </Agenttypes>
</Application>
```

Spatial Coordination Concern
Organizational Coordination Concern
Agent Definition File (ADF)

Figure 4: Instance of an application descriptor


AntForaging.application.xml

```xml
<Application>
 <Spaces>
  <CoordinationSpace Type="Indirect Interaction">
   <AgentConfiguration type="Ant"/>
   <MediatedMediumConfiguration Id="Pheromone 2D">
    <ElementConfiguration Type="Pheromone" isMarker="true"
      strength="10">
    <ElementActivityConfiguration type="Propagation">
      <rate>2</rate>
      <factor>5</factor>
    </ElementActivityConfiguration>
    <ElementActivityConfiguration type="Evaporation">
      <rate>1</rate>
      <factor>2</factor>
    </ElementActivityConfiguration>
   </ElementConfiguration>
  </MediatedMediumConfiguration>
   <Options>
    <Type>Tspace</Type>
    <IP>192.168.10.1</IP>
    <Port>25787</Port>
   <Options/>
  </CoordinationSpace>
 </Spaces>
 <...>
</Application>
```

Figure 5: Coordination space details

shows that agents are simply added to an application by referencing their agent definition file.

In this way, the integration of coordination spaces follows a declarative approach. Right now, the framework is realised in a prototype implementation for indirect interaction. Developers can use this existing implementation to speed up the development of their application. On the other hand, they can easily add new implementations for coordination spaces due to clearly specified interfaces for spaces and coordination media. It is also envisioned to provide a reference implementation for direct interaction.

4.3 Case Study: Ant Foraging

Finally, a concrete case study has been implemented to illustrate how, for example, an *ant foraging* [Mamei and Zambonelli, 2005] scenario can be realised using the concept of coordination spaces. It will also be demonstrated how this approach can be used to enable easy ways to optimize parameters of an application.

Ant foraging is one of the well researched examples from the field of nature inspired decentralised coordination [Mamei *et al.*, 2006]. In this example, ants leave pheromones (markers) in the environment on their way back to their nest, once they have found food. Pheromones influence the decisions of other foragers, which path to follow. These decisions are made stochastically and are based on the strength of the perceived pheromones. Due to the simplicity of the ant actions it is well-suited to study the applicability and efficiency of coordination mechanisms, as developers can focus on tuning the coordination. In computer science, the inspiration of ant foraging has been used for network routing [Sim and Sun, 2002] or other algorithm optimization [Dorigo *et al.*, 1999].

This case study uses the reference medium implementation for indirect interaction. The medium is based on *TSpaces*[6] which is a implementation of a *tuple space* [Gelernter, 1985] server offered by *IBM*. Tuple spaces are an associative memory often used in distributed computing and were first introduced with the *Linda* coordination language. In this scenario, it is used as the place where the pheromones are dropped by the ants. The behaviour and life cycle of pheromones is managed by the coordination medium, e.g. the tuple space, and can be specified within the coordination space. Figure 5 gives an example: it shows how *semantic aspects* like *evaporation* and *propagation* are specified. Furthermore, the developer can adjust technical aspects of the medium like the IP or port number if needed. Especially the first aspect shows the benefit of this approach. The developer is freed from changing code in order to change the behaviour of the application. Such changes can even be performed by non-programmers that have domain knowledge and want to optimize a system.

The example of ant foraging may be used as a model for a transport network. In order to optimize the system, essential parameters have to be identified and set first. For ant foraging e.g. evaporation and propagation are two important parameters. Running simulation experiments based on the coordination space, two things can be discovered: if the evaporation rate of the pheromones is high and the propagation rate is low, information about good paths disappears too fast and, therefore, the efficiency of the ants in finding optial paths decreases. If, on the other hand, the evaporation rate is low and the propagation rate is high, it happens that pheromones do not disappear fast enough. In consequence, then almost all paths are marked as being "good one" and, again, the efficiency of the ants decreases. The benefit of the concept of coordination spaces in such a scenario is that, in order to find an optimal configuration for the ant foraging, only parameters within the description file have to be changed whereas the program code itself can remain the same. Further benefits are that the coordination implementation (e.g. pheromone diffusion etc.) can be reused in different application scenarios and also that an ant-foraging-based transport network application can easily be adopted to make use of another ready-to-use coordination mechanism.

5 Conclusion and Future Work

One main concern of this paper was highlighting the importance of coordination as a vital aspect of many application areas. Nowadays, coordination is often

[6]http://www.almaden.ibm.com/cs/tspaces/Version3/

specified as part of the functional design of the target system. Thus, coordination becomes intertwined with other system facets and cannot be easily adjusted or exchanged. This paper argues that coordination should be considered as a separate dimension and realized as far as possible independently from the consituting parts promoting a clear separation of concerns.

As a possible solution an integrated approach, called coordination spaces, for the issue of explicit coordination in MAS has been proposed. On the one hand it consists of a well defined model of coordination properties, most importantly agents and coordination media that can be used to specify coordination in MAS in a declarative way. On the other hand it consists of a MAS infrastructure that is capable of automatically processing this coordination specification. Coordination spaces offer a non-invasive and transparent way of coordination in MAS, i.e. agent code does not have to be changed, making it possible to coordinate agents independently of their internal architectures. Therefore, heterogeneous agents, e.g. BDI and task-based agents, can be coordinated within the same MAS. Furthermore, coordination mechanisms can be easily exchanged without impacting agent functionalities.

Future work will on the one hand strive towards providing more coordination media implementations aiming at a catalog of reusable coordination patterns with ready-to-use implementations. On the other hand, it is investigated how to enable and manage the dynamic selection of coordination media at system runtime.

Acknowledgments

We would like to thank the Deutsche Forschungsgemeinschaft (DFG) for supporting this work in a project on "Self-organisation based on decentralized Co-ordination in Distributed Systems" (SodekoVS).

References

[DeWolf and Holvoet, 2006] T. DeWolf and T. Holvoet. A catalogue of decentralised coordination mechanisms for designing self-organising emergent applications. Technical Report CW 458, Dept. of Comp. Science, K.U. Leuven, 8 2006.

[Dorigo *et al.*, 1999] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artif. Life*, 5(2):137–172, 1999.

[Gelernter, 1985] D. Gelernter. Generative communication in linda. *ACM Trans. Program. Lang. Syst.*, 7(1):80–112, 1985.

[Gouaich and Michel, 2005] A. Gouaich and F. Michel. Towards a unified view of the environment(s) within multi-agent systems. *Informatica (Slovenia)*, 29(4):423–432, 2005.

[Keil and Goldin, 2006] D. Keil and D. Goldin. Indirect interaction in environments for multi-agent systems. In *Environments for Multi-Agent Systems II*, pages 68–87. Springer, 2006.

[Malone and Crowston, 1994] T. W. Malone and K. Crowston. The interdisciplinary study of coordination. *ACM Comp. Surv.*, 26(1):87–119, 1994.

[Mamei and Zambonelli, 2005] M. Mamei and F. Zambonelli. *Field-Based Coordination for Pervasive Multiagent Systems.* Springer, 2005.

[Mamei *et al.*, 2006] M. Mamei, R. Menezes, R. Tolksdorf, and F. Zambonelli. Case studies for self-organization in computer science. *J. Syst. Archit.*, 52(8):443–460, 2006.

[Pokahr and Braubach, 2009a] A. Pokahr and L. Braubach. From a research to an industrial-strength agent platform: Jadex V2. In *WI 2009*, pages 769–778. Oester. Comp. Gesell., 2 2009.

[Pokahr and Braubach, 2009b] A. Pokahr and L. Braubach. The notions of application, spaces and agents — new concepts for constructing agent applications. In *Multikonf. Wirtschaftsinf.*, 2009.

[Rao and Georgeff, 1995] A. Rao and M. Georgeff. Bdi agents: From theory to practice. In *Proc. of the 1. Int. Conf. on Multiagent Syst.*, pages 312–319. The MIT Press, 1995.

[Ricci *et al.*, 2007] A. Ricci, M. Viroli, and A. Omicini. The A&A programming model and technology for developing agent environments in MAS. In *ProMAS 2007*, pages 89–106. Springer, 2007.

[Russell and Norvig, 2002] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 2 edition, 2002.

[Sim and Sun, 2002] K.M. Sim and W.H. Sun. Multiple ant-colony optimization for network routing. *Int. Conf. on Cyberworlds*, 0:0277, 2002.

[Sudeikat and Renz, 2009a] J. Sudeikat and W. Renz. MASDynamics: Toward systemic modeling of decentralized agent coordination. In *Kom. in Vert. Syst.*, Informatik aktuell, pages 79–90, 2009.

[Sudeikat and Renz, 2009b] J. Sudeikat and W. Renz. Programming adaptivity by complementing agent function with agent coordination: A systemic programming model and development methodology integration. *Com. of SIWN*, (7):91–102, 4 2009.

[Sudeikat *et al.*, 2009] J. Sudeikat, L. Braubach, A. Pokahr, W. Renz, and W. Lamersdorf. Systematically engineering selforganizing systems: The sodekovs approach. Elect. Com. of the EASST, 3 2009.

[Viroli *et al.*, 2007] M. Viroli, T. Holvoet, A. Ricci, K. Schelfthout, and F. Zambonelli. Infrastructures for the environment of multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 14(1):49–60, July 2007.

[Weyns *et al.*, 2005] D. Weyns, M. Schumacher, A Ricci, M. Viroli, and T. Holvoet. Environments in multiagent systems. *The Knowledge Engineering Review*, 20(02):127–141, 2005.

[Wooldridge and Jennings, 1995] M. Wooldridge and N. Jennings. Intelligent agents: Theory and practice. *The Knowl. Engin. Rev.*, 10(2):115–152, 1995.