The Knowledge Engineering Review, Vol. 30:4, 435–453.   © Cambridge University Press, 2015
doi:10.1017/S0269888915000090

Towards next generation coordination infrastructures

JUAN A. RODRIGUEZ-AGUILAR[1], CARLES SIERRA[1], JOSEP Ll. ARCOS[1], MAITE LOPEZ-SANCHEZ[2] and INMACULADA RODRIGUEZ[2]

[1]*IIIA, Artificial Intelligence Research Institute, CSIC, Spanish National Research Council, Campus UAB, 08193 Bellaterra, Spain;*
e-mail: jar@iiia.csic.es, sierra@iiia.csic.es, arcos@iiia.csic.es
[2]*MAiA Department, Universitat de Barcelona, C/Gran Via de les Corts, Catalanes 585, 08007 Barcelona, Spain;*
e-mail: maitedopez@ub.edu, inrna@ub.edu

Abstract

Coordination infrastructures play a central role in the engineering of multiagent systems. Since the advent of agent technology, research on coordination infrastructures has produced a significant number of infrastructures with varying features. In this paper, we review the the state-of-the-art coordination infrastructures with the purpose of identifying open research challenges that next generation coordination infrastructures should address. Our analysis concludes that next generation coordination infrastructures must address a number of challenges: (i) to become *socially aware*, by facilitating human interaction within a MAS; (ii) to assist agents in their decision making by providing *decision support* that helps them reduce the scope of reasoning and facilitates the achievement of their goals; and (iii) to increase *openness* to support on-line, fully decentralised design and execution. Furthermore, we identify some promising approaches in the literature, together with the research issues worth investigating, to cope with such challenges.

1 Introduction

Multiagent systems (MAS; Wooldridge, 2002) are composed of autonomous entities (agents) that interact within a dynamic environment to achieve their common and/or individual goals. The achievement of such goals typically requires the effective coordination of agents' activities. Coordination is necessary in cooperative settings, where agents interact to achieve a common goal, as well as in competitive ones, where each agent is self-interested and acts to achieve her own goals.

From an engineering point of view, building a MAS is a very intricate matter. On the one hand, a MAS is a particular type of distributed concurrent system. On the other hand, although a MAS is populated by autonomous agents, it must achieve global goals. The first MAS applications developed *ad hoc infrastructures* from scratch (Jennings *et al.*, 1998) to coordinate agents' interactions. Nonetheless, as agent technology has matured, a wealth of methodologies (e.g. Gaia (Zambonelli *et al.*, 2003), Tropos (Bresciani *et al.*, 2004)) and coordination infrastructures (JADE (Bellifemine *et al.*, 2001), FIPA-OS (Poslad *et al.*, 2000), Grasshopper (Baumer *et al.*, 1999), JACK (Howden *et al.*, 2001), WADE (Caire *et al.*, 2008), CArtAgO (Ricci *et al.*, 2011), Madkit (Gutknecht *et al.*, 2001), S-Moise+ (Hubner *et al.*, 2005), OR4MAS (Kitio *et al.*, 2007), AMELI (Esteva *et al.*, 2004), ALIVE (Vazquez-Salceda *et al.*, 2010), MACODO (Weyns *et al.*, 2010)) have been produced to ease and support MAS development. Coordination infrastructures have been developed as a particular type of MAS middleware responsible for mediating agent interactions and providing the means for agents to access the external world (containing physical objects, legacy applications and services). Furthermore, coordination infrastructures have evolved to become *general-purpose* infrastructures, which are built on top of some distributed middleware platform (e.g.

RMI, CORBA. SOAP) to provide coordination services that can be reused across multi-agent system applications.

The purpose of this paper is two-fold. First, to review the state-of-the-art on coordination infrastructures for MAS. Second, to identify open research challenges that the next generation of coordination infrastructures must tackle. Our analysis is based on understanding how state-of-the-art coordination infrastructures propose: (i) to enact interactions; (ii) to support coordination decisions (of the infrastructure itself regarding the adaptation of coordination mechanisms as well as of participating agents); (iii) to support coordination design; and (iv) to engineer the infrastructure. Such analysis allows us to conclude that

- Coordination infrastructures have largely focused on implementing functionalities to structure coordination via a wealth of coordination models (interaction protocols, teams, workflows, organisations or institutions). Software agents have been considered the main customers for such developments. Nonetheless, helping humans interact within a MAS remains rather unexplored with the exception of first explorations conducted by Bogdanovych (2007) and Trescak *et al.* (2011).
- Coordination support has been largely overlooked by research on coordination infrastructures. Therefore, endowing a coordination infrastructure with adaptation capabilities and assisting agents in their interactions stand out as two open research questions. Regarding the latter one, developments so far in coordination infrastructures have taught us how arduous and intricate is to develop software agents that interact in an MAS. Thus, coordination support is fundamental to help agents reduce the scope of reasoning with the aim of achieving their goals.
- The vast majority of research on coordination infrastructures, with the exception of Aldewereld *et al.* (2010) and Vazquez-Salceda *et al.* (2010), has focused on the off-line design of coordination. Since open MAS are very dynamic and heterogenous systems, we take the stance that coordination mechanisms must be dynamically composed at run-time. Thus, along the lines of Weyns *et al.* (2009: 5), coordination design must move from programming to flexible composition. The on-line decentralised design (by flexible composition) and enactment of coordination infrastructures appears as a future research challenge. Furthermore, coordination infrastructures should also consider how to allow agents to choose their own coordination mechanism, namely their own rules of interaction.
- Most coordination infrastructures have successfully explored distributed implementations, and yet there is still room for exploration. In particular, peer-to-peer (P2P) systems offer a high degree of decentralisation. Furthermore, there are additional benefits inherent to P2P systems that are potentially interesting for open MAS (e.g. self-organisation, resilience to faults and attacks, low barrier to deployment, etc.).

In the rest of the paper, we try to analyse in more depth the open research challenges identified above and we also discuss some approaches to cope with such challenges.

The rest of the paper is structured as follows. Section 2 reviews the state-of-the-art on coordination infrastructures for MAS with the aim of identifying open research challenges. Sections 3, 4, and 5 try to analyse in more depth such open research challenges. In Section 3, we devote our attention to supporting humans' interactions in MAS. Next, in Section4 we elaborate on coordination support services for agents in an MAS. In Section 5, we propose using P2P coordination infrastructures to support the on-line, fully decentralised design and execution of MAS. Finally, Section 6 draws some conclusions and sets paths to future research.

2 Coordination infrastructures: an abbreviated review

The purpose of this section is manyfold. First of all, in Section 2.1 we identify the role of a coordination infrastructure within an MAS. Next, in Section 2.2 we analyse the key requirements of the engineering of coordination infrastructures. Thereafter, Section 2.3 analyses how the coordination infrastructures in the literature have contributed to tackle such requirements and Section 2.4 summarises a comparison of such contributions. Finally, Section 2.5 builds upon the analysis in Section 2.3 to detect the requirements that state-of-the-art coordination infrastructures are not satisfying, and hence are worthy future investigation as research challenges.

2.1. Coordination infrastructures for multi-agent systems

The core responsibility of a coordination infrastructure for MAS is to *mediate* agent interactions. These can be either direct, via message passing, or indirect, via the environment (e.g. tags (Platon *et al.*, 2006), digital pheromones (Dyke Parunak *et al.*, 2005)). A coordination infrastructure governs interactions between agents according to the rules of some coordination mechanism. For instance, consider an MAS whose coordination mechanism is an English auction (Parsons *et al.*, 2011). The coordination infrastructure will ensure that a message issued by a bidder reaches the auctioneer. Moreover, the coordination infrastructure will also ensure that the messages issued by bidders and auctioneers abide by the rules of the English auction protocol. Alternatively, consider that agents employ stigmergy to communicate by leaving pheromones in some environment. The coordination infrastructure will be responsible of the evaporation of the pheromones.

A further, fundamental responsibility of coordination infrastructures is *virtualisation*: to provide an interface to the external world, which basically amounts to providing the means to interact with: (i) sensors and actuators to operate on physical objects; and (ii) external services and applications (such as legacy systems or web services). With the aim of realising such main functionalities, a coordination infrastructure is computationally realised as a particular type of middleware for MAS.

Figure 1 (introduced and thoroughly discussed in Weyns *et al.*, 2009) provides a global picture of the middleware required to run an MAS. The architecture of this middleware is composed of the following layers:

- *Distributed and host infrastructure middleware*. Distributed middleware services (e.g. RMI, SOAP, etc.) are the basic infrastructure to build MAS.

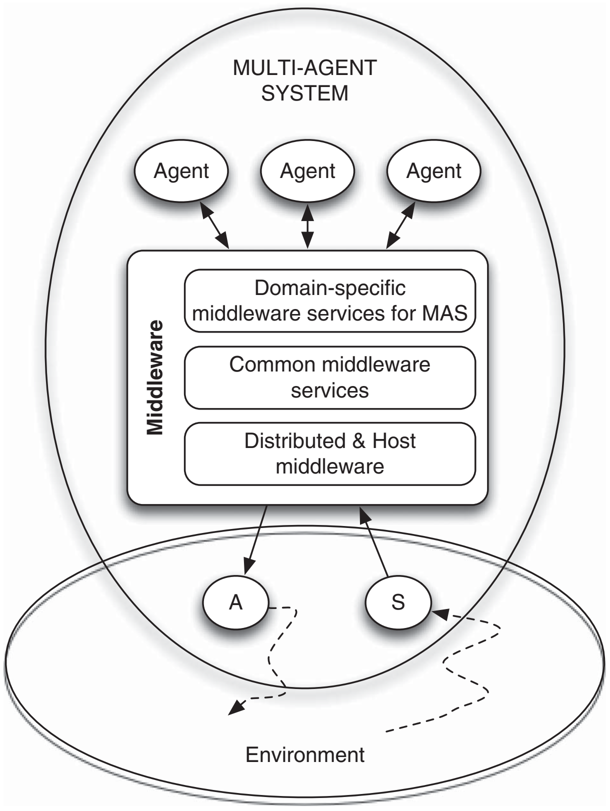

MULTI-AGENT SYSTEM
Agent
Agent
Agent
Middleware
Domain-specific middleware services for MAS
Common middleware services
Distributed & Host middleware
A
S
Environment

Figure 1 Middleware for a multi-agent system (A = actuator, S = sensor).

- *Common middleware services* such as security, persistency, transactions, etc.
- *Domain-specific middleware services*. This layer contains services to support agent interaction. Coordination infrastructures are part of this layer. They are mostly built on top of the distributed and host infrastructure middleware.

2.2. *Key requirements*

In what follows, we identify the design issues that an MAS coordination infrastructure must consider.

2.2.1. *How to enact interactions*

Interactions are at the heart of coordination. The first design choice of a coordination infrastructure engineer is to decide the types of agents that are allowed to interact by means of the infrastructure: human agents, software agents, or both. By supporting the interaction of both human and software agents, a coordination infrastructure can host the operation of hybrid MAS. On the one hand, allowing humans to participate in an MAS requires the building of interfaces that ease humans' interaction in MAS scenarios. On the other hand, supporting the participation of software agents requires the use of some agent communication language (ACL; such as e.g. FIPA (Labrou & Finin, 1997) or KQML (Labrou & Finin, 1997)) that agents can employ to shape their interactions as speech acts (Searle, 1969). Besides that, there is the matter of structuring interactions through some *coordination structure*. Interaction protocols, workflows, norms, organisations, and institutions are examples of coordination structures.

Notice that interactions within an MAS can also involve services. Thus, agents in an MAS can benefit from available web services. With this aim, a coordination infrastructure must support the interaction of agents and services, which amounts to providing agent-to-service interfaces. Furthermore, there is the issue of supporting the interaction of the MAS with the environment, namely with the external world where agents are situated.

To summarise, there are three types of interactions a coordination infrastructure might support: (i) *agent to agent*; (ii) *agent to service*; and (iii) *agent to environment*.

2.2.2. *How to support coordination decisions*

MAS research focuses on having self-interested agents interact so that some desired collective properties are achieved. This must occur despite changes in the agent population and changes in the environment wherein the MAS is situated. If an MAS intends to cope with such dynamic changes, it must be endowed with adaptive capabilities. This amounts to endowing the coordination infrastructure with the capability of deciding when and how to change the coordination mechanism mediating agent interactions when the collective goals are not achieved.

On the other hand, we can consider a similar problem for agents' decision making. Each agent must face the problem of deciding which actions to take to achieve her own goals while abiding by the rules of the coordination mechanism run by the MAS coordination infrastructure. A coordination infrastructure has to help an agent to achieve her goals by providing coordination decision support, namely by providing information or by recommending interactions, or even interaction plans, which assist agents in their decision making.

2.2.3. *How to support coordination design*

Coordination can be designed either off-line (before a coordination infrastructure starts operating), or on-line (at run-time). Off-line design requires that the engineer of an MAS application specifies the coordination rules of the application based on the coordination model employed by the infrastructure. For instance, if a coordination infrastructure employs Finite State Machines (FSMs) as the means of interacting, an application engineer must specify the FSMs required by its application. As an example, consider an electronic auction house to sell fish. The engineer would provide the infrastructure with an FSM specifying a Dutch auction.

By on-line design, we mean that agents themselves can agree on the terms of their interactions at run-time. Such capability poses a number of challenges to a coordination infrastructure, but fundamentally we identify two main issues. First, the infrastructure must provide matchmaking mechanisms that allow agents finding potential interaction partners. Second, the infrastructure must allow to enact coordination mechanisms agreed-upon by groups of agents at run-time. Following the example above, in the framework of an electronic auction house, agents might vote on which auction protocol to use, and the infrastructure would subsequently enact it.

2.2.4. How to engineer a coordination infrastructure

Although MAS are a particular type of distributed systems (Stone & Veloso, 2000), a coordination infrastructure must not be necessarily distributed. Therefore, it is up to the designer of the coordination infrastructure to decide whether the implementation of the functionalities satisfying the above-identified requirements is either centralised, distributed, or partially distributed. This choice is important because it largely affects scalability, security and fault tolerance.

Furthermore, a coordination infrastructure engineer must also choose whether the infrastructure remains architecturally neutral, hence allowing agents built according to any given architecture to participate, or otherwise enforce agents to run over some particular architecture (e.g. behaviour-based (Bellifemine *et al.*, 2001), or BDI (Rao & Georgeff, 1998)). Remaining architecturally neutral is a typical choice for coordination infrastructures hosting open MAS[1].

2.3. State-of-the-art

After analysing the coordination infrastructures in the literature, we can group them into three generations that we characterise next.

2.3.1. First-generation coordination infrastructures

First-generation coordination infrastructures were ad-hoc infrastructures developed to run particular MAS applications. Thus, each MAS application developed its own coordination infrastructure from scratch. A coordination infrastructure was not aimed at running more than a single MAS application. For instance, the marketplaces in Rodriguez-Aguilar *et al.* (1997), Wurman *et al.* (1998), and Sandholm (2002) were built each upon their own infrastructure.

2.3.2. Second-generation coordination infrastructures

Second-generation coordination infrastructures were spurred by the need for developing general-purpose infrastructures that could serve for a wide variety of MAS applications, hence reducing development time and cost. FIPA-compliant platforms played a major role to advance the state-of-the-art on general-purpose coordination infrastructures. The Foundation for Intelligent Physical Agents (FIPA[2]), a non-profit international organisation devoted to produce specifications for generic agent technologies, proposed a specification of the functionalities that any agent platform should provide (FIP, 2002). In particular, FIPA proposed that any agent platform has to provide the following services: (i) an agent management service (AMS) that handles agents' executions and allows finding other agents (i.e. white pages service); (ii) a reliable transport service among agents; and (iii) a directory service to register and discover agent services (i.e. yellow pages service). A requirement of the transport service is to support the interoperability with other compliant platforms to allow the communication between agents running in any of them. Notice that the AMS and the transport service can be regarded as basic coordination services required for the deployment of any MAS.

[1] 'An open system is one in which the structure of the system itself is capable of dynamically changing. The characteristics of such a system are that its components are not known in advance; can change over time; and can consist of highly heterogeneous agents implemented by different people, at different times, with different software tools and techniques' (Sycara, 1998).
[2] http://www.fipa.org.

After the release of the FIPA specification, several FIPA-compliant platforms were developed, such as JADE (Java Agent Development Framework; Bellifemine *et al.*, 2001), FIPA-OS (Poslad *et al.*, 2000), and Grasshopper (Baumer *et al.*, 1999). Notice that JADE has been the most widely used FIPA-compliant platform. Overall, FIPA-compliant coordination infrastructures help agents coordinate by offering mechanisms to discover agents and services, and a transport service transparent to agent physical locations. Regarding the latter issue, JADE puts significant effort on realising a distributed architecture transparently to software agents. Moreover, JADE has also addressed agent mobility (likewise Grasshopper (Baumer *et al.*, 1999)).

Overall, we observe that second-generation coordination infrastructures have particularly focused on providing interaction support for agent to agent interaction and engineering-distributed architectures.

2.3.3. *Third-generation coordination infrastructures*

Third-generation coordination infrastructures go beyond second-generation coordination infrastructures along several directions. First, this type of infrastructures turns their attention to social coordination models that offer a higher level of abstraction and further expressiveness than the interaction protocols exploited by first-generation infrastructures. Second, the infrastructures in this group become situated by supporting the interaction with services (fostered by the spread of service-oriented computing) and environments. Third, these infrastructures start incorporating intelligence to allow an MAS to dynamically adapt to unpredictable changes (fostered by the spread of autonomic computing (Brazier *et al.*, 2009) and research on self-organisation (Serugendo *et al.*, 2006)). Finally, the need for easing and speeding up development has led these infrastructures to produce a number of IDEs (Integrated Development Environments) that provide coordination design support.

In what follows we give more details on the contributions of this group of infrastructures to the requirements posed in Section 2.2.

(a) Interaction support: Regarding coordination models, several alternatives to interaction protocols have been proposed in the literature. JACK (Howden *et al.*, 2001) uses the notion of team: agent coordination is specified from the abstract viewpoint of a team of agents as a whole from a high-level perspective. WADE (Caire *et al.*, 2008), an extension of JADE, takes inspiration on business processes and employs *workflows* to specify agent coordination[3]. CArtAgO (Ricci *et al.*, 2011) employs the concept of *workspaces*, namely local contexts to which agents and artifacts might belong. By using coordination artifacts, coordination policies can be designed for workspaces. Each of these coordination infrastructures aims at supporting a different programming paradigm. Thus, JACK is intended to support *team-oriented programming* (Cohen & Levesque, 1991), WADE supports *agent-oriented programming* (Shoham, 1993), and CArtAgO is conceived to support *environment-oriented programming* (Weyns *et al.*, 2007).

Besides the contributions above, in the last few years coordination infrastructures have turned their attention to social coordination models. Thus, research on coordination models has grown around the concept of organisation. The notion of organisation was early introduced to computational systems within the field of Distributed Artificial Intelligence (DAI; Gasser *et al.*, 1987; Pattison *et al.*, 1987). Thus, early work in DAI identified organisational design as one of the main issues in order to cope with the complexity of designing DAI systems. More recently, a significant number of coordination infrastructures (Madkit (Gutknecht *et al.*, 2001), S-Moise+ (Hubner *et al.*, 2005), OR4MAS (Kitio *et al.*, 2007)[4] , AMELI (Esteva *et al.*, 2004), ALIVE (Vazquez-Salceda *et al.*, 2010)) have adopted organisations to provide a higher level of abstraction for agent coordination. Overall, despite slight differences between coordination infrastructures, an organisational model is composed of (i) a social structure that defines the roles participants can play in the organisation along with their relationships; (ii) a communication structure defining an agent

[3] In fact, WADE considers two application contexts: (i) WADE is used as a workflow engine and workflows implement processes that coordinate different systems (e.g. agents); and (ii) WADE is used as a agent-oriented development framework and workflows implement agent tasks.

[4] OR4MAS employs CArtAgO to provide a coordination infrastructure for the Moise + model (Hubner *et al.* 2002). Thus, while an organisation is specified following the Moise + model, CArtAgO is used to implement the coordination infrastructure.

communication language along with some ontology; (iii) a normative structure to define the regulations imposed by the organisation; and (iv) an interaction structure defining the interaction protocols that agents can employ within the organisation. Therefore, notice that interaction protocols are a typical component of organisations. Running an MAS as an organisation requires that a coordination infrastructure is capable of understanding the specification of an organisation model, can process agents' actions and make the state of the organisation evolve according to the specification. Unlike JACK, JADE, WADE, and CArtAgO, organisation-based coordination infrastructures are intended to facilitate organisation-oriented programming (Boissier & Sichman, 2004).

Organisation-oriented approaches argue that an organisation itself provides an environment for agent interaction, whereas environment-oriented approaches, such as CArtAgO, argue that the environment requires an explicit representation.

Service interaction has been handled by coordination infrastructures in several ways: via some service wrapping (through an artifact like CArtAgO or through an agent-like Madkit), via some generic service interface (e.g. AMELI (Arcos *et al.*, 2006)), or by allowing (web) service calls (e.g. JACK, WADE, ALIVE). But surely the most advanced service interaction is provided by ALIVE, since services are considered first-class citizens and, in fact, an organisation is employed to orchestrate services, namely to dynamically select, compose, and invoke services. In fact, the ALIVE framework was developed to support the engineering of service-oriented systems instead of MAS.

Although the above-mentioned coordination infrastructures have made headway in improving interaction support (agent to agent, agent to service, and agent to environment), definitely the kind of interaction that all coordination infrastructures have failed at supporting is human interaction. So far the support is limited to simple graphical interfaces that allow humans to interact as if they were software agents (in JADE, WADE, and AMELI) or artifacts (in CArtAgO). Only recent work on virtual electronic institutions (Bogdanovych, 2007; Trescak *et al.*, 2011), as extensions of AMELI, has started to consider anthropomorphic approaches to human interaction.

(b) Coordination decision support: In Section 2.2, we referred to providing support to coordination decisions as a key requirement for coordination infrastructures. There we distinguished between decision support for the infrastructure (to change its coordination mechanism) and decision support for the agents in the MAS run by the infrastructure (to help them achieve their goals).

Regarding the infrastructure, the adaptation of the coordination model to varying situations is an important topic due to the dynamic nature of MAS. This issue has not been sufficiently tackled by current research. Some exceptions are S-Moise+, where a special role Reorg is in charge of reorganising how tasks are assigned to agents, AMELI, extended in Arcos *et al.* (2008) with self-adaptation capabilities that allow to tune the parameters of the coordination model, and ALIVE. Nonetheless, there are some promising proposals to endow organisations with self-adaptation capabilities that can be eventually incorporated into coordination infrastructures. For instance, in Zhang *et al.* (2009) and Campos *et al.* (2011) different machine-learning techniques are proposed for this purpose. Alternatively, MACODO (Weyns *et al.*, 2010) offers an interesting perspective regarding the support for organisation adaptation. It provides a middleware for dynamic organisations that separates role-based agent behaviours from the management of organisation dynamics. Specifically, it considers laws for joining, leaving, merging, and splitting organisations that change the composition of overall organisations. In fact, the dynamics associated to organisations have attracted the attention of the research community both from structural and a normative points of view. By structural dynamics we mean changes in the agent's populations and their relationships (Horling *et al.*, 2001; Hubner *et al.*, 2004; Dignum *et al.*, 2005; Kota *et al.*, 2008). By normative dynamics, we refer to changes in regulations (Artikis *et al.*, 2009; Campos *et al.*, 2011). Nevertheless, many adaptation issues such as, for example, adaptation costs, still remain open.

Regarding the agent side, to the best of our knowledge no coordination infrastructure provides coordination decision support to participating agents.

(c) Coordination design support: Most coordination infrastructures require the off-line (at design time) specification of the coordination model in a centralised manner (namely there is a single designer of the coordination model). The only exceptions are ALIVE, which allows the on-line (at run-time) composition of web services, and MACODO. In order to support coordination design, some coordination

infrastructures count on an IDE that supports the specification of the MAS (e.g. WADE offers the WOLF IDE (Sacchi *et al.*, 2011), AMELI offers EIDE (Arcos *et al.*, 2005), and ALIVE offers OperettA (Aldewereld & Dignum, 2010)).

(d) Architecture of the coordination infrastructure: The majority of coordination infrastructures in this group are distributed (WADE, JACK, CArtAgO, Madkit, S-MOISE+, OR4MAS, AMELI, ALIVE, MACODO). There are more differences regarding the choice of agent architecture. Thus, we distinguish three main choices: architecturally neutral (Madkit, S-MOISE+, OR4MAS, AMELI), BDI-based (JACK, CArtAgO), own (JADE, WADE, ALIVE).

2.4. *Comparing coordination infrastructures*

From the discussion above in Section 2.3, we can conclude that coordination infrastructures have evolved from *ad hoc* infrastructures (first generation), to low-level infrastructures (second generation) mostly concerned with communication and distribution, to higher level infrastructures (third generation) that embrace social coordination models, embed intelligence to support adaptiveness, and are situated by providing interaction support to agents, services, and the environment.

Table 1 compares the features of the most salient coordination infrastructures based on the discussion in Section 2.3 while considering the key requirements posed in Section 2.2. The purpose of this comparison is to provide the means to identify open research challenges as we do in Section 2.5. With this aim, Table 1 considers the following dimensions:

1. Interaction suport
- *Coordination model* employed by the infrastructure (interaction protocols, workflows, norms, organisations, and institutions).
- *Human interaction*. Facilities offered to humans to interact with the coordination infrastructure.
- *Service interaction*. Capability of interacting with services.
- *Environment interaction*. Whether the infrastructure explicitly considers an environment and provides means to interact with.

2. Coordination decision support
- *Adaptation capabilities* of the coordination infrastructure to support the adaptation of the coordination mechanism at run-time.
- *Agent decision support* as decision support facilities offered to agents interacting via a coordination infrastructure.

3. Coordination design support
- *Design mode*. To distinguish whether coordination can be designed off-line, on-line (at run time), or both.
- *Engineering support*. Facilities provided to MAS engineers to design coordination.

4. Architecture
- *Agent architecture* supported by the infrastructure (e.g. JADE, JASON (Bordini *et al.*, 2005b), Jadex (Pokahr *et al.*, 2005)).
- *Infrastructure architecture* (e.g. centralised, distributed).

2.5. *Identifying open research challenges*

Table 1 compares the coordination infrastructures we have analysed in Section 2.3. By analysing Table 1 along with the state-of-the-art, we can identify the research challenges that have not been thoroughly addressed so far in the literature, and hence stand as future research opportunities. We group such research issues along four dimensions, each one corresponding to the research challenges analysed in Section 2.2.

How to enact interactions: Coordination infrastructures have largely focused on implementing functionalities to mediate interactions via interaction protocols, teams, workflows, organisations, or institutions. Software agents have been considered the main customers for such developments. However, helping humans interact within an MAS remains a rather unexplored, intricate matter. The lack of support for human interaction impedes the realisation of hybrid MAS that seamlessly integrate human and software agents.

Table 1 Comparing coordination infrastructures.

Requirement	Dimension	Coordination infrastructure	Feature
Interaction support	Coordination model	JADE	Interaction protocol
		WADE	Worflow
		JACK	Team
		CArtAgO	Workspace
		Madkit	Organisation (AGR model)
		S-MOISE+	Organisation
		OR4MAS	Organisation
		AMELI	Institution
		ALIVE	Organisation
		MACODO	Organisation
	Human interaction	JADE	Dummy agents
		WADE	Dummy agents
		JACK	—
		CArtAgO	GUI-based artifacts
		Madkit	—
		S-MOISE+	—
		OR4MAS	—
		AMELI	Dummy agents
		ALIVE	—
		MACODO	—
	Service interaction	JADE	—
		WADE	Web services
		JACK	Web services
		CArtAgO	Artifact-based interface
		Madkit	Agent-based interface
		S-MOISE+	—
		OR4MAS	Artifact-based interface
		AMELI	Generic interface
		ALIVE	Web services
		MACODO	—
	Environment awareness	JADE	—
		WADE	—
		JACK	—
		CArtAgO	Artifact-based
		Madkit	—
		S-MOISE+	—
		OR4MAS	Artifact-based
		AMELI	—
		ALIVE	—
		MACODO	Context-based
Coordination decision support	Adaptation	JADE	—
		WADE	—
		JACK	—
		CArtAgO	—
		Madkit	—
		S-MOISE+	Reorganisation
		OR4MAS	Reorganisation
		AMELI	Parameter tuning of coordination model
		ALIVE	Reorganisation
		MACODO	Reorganisation
	Agent decision support	JADE	—
		WADE	—
		JACK	—
		CArtAgO	—

Table 1 (Continued)

Requirement	Dimension	Coordination infrastructure	Feature
		Madkit	—
		S-MOISE+	—
		OR4MAS	—
		AMELI	—
		ALIVE	—
		MACODO	—
Coordination design support	Design mode	JADE	Off-line
		WADE	Off-line
		JACK	Off-line
		CArtAgO	Off-line
		Madkit	Off-line
		S-MOISE+	Off-line
		OR4MAS	Off-line
		AMELI	Off-line
		ALIVE	Off-line and on-line
		MACODO	Off-line and on-line
	Engineering support	JADE	API and monitoring tools
		WADE	WOLF IDE
		JACK	JACK IDE
		CArtAgO	—
		Madkit	API and run-time tools
		S-MOISE+	API
		OR4MAS	—
		AMELI	EIDE IDE
		ALIVE	Operetta IDE
		MACODO	—
Architecture	Agent architecture	JADE	Behaviour-based
		WADE	Behaviour-based
		JACK	BDI
		CArtAgO	JASON, Jadex
		Madkit	Neutral
		S-MOISE+	Neutral
		OR4MAS	Neutral
		AMELI	Neutral
		ALIVE	Alive
		MACODO	Neutral
	Infrastructure architecture	JADE	Distributed
		WADE	Distributed
		JACK	Distributed
		CArtAgO	Distributed
		Madkit	Distributed
		S-MOISE+	Distributed
		OR4MAS	Distributed
		AMELI	Distributed
		ALIVE	Distributed
		MACODO	Distributed

How to support coordination decisions: This requirement has been largely overlooked by research on coordination infrastructures with the exceptions identified in Section 2.3. Therefore, endowing a coordination infrastructure with adaptation capabilities and assisting agents in their interactions stand out as two open research questions. Regarding the former one, as we have pointed out in Section 2.3, the issue has already been identified in the literature and some significant contributions have already appeared. Regarding the latter one, although current MAS developments have taught us how arduous and intricate is to develop software agents that interact within an MAS, current research on coordination infrastructures has not considered how to assist agents in their decision making. Thus, providing coordination support to help agents reduce the scope of reasoning with the aim of achieving their goals is an open issue for future coordination infrastructures.

How to support coordination design: The vast majority of research on coordination infrastructures, with the exception of Aldewereld *et al.* (2010) and Vazquez-Salceda *et al.* (2010), has focused on the off-line design of coordination. Moreover, coordination design is typically programmed in a centralised manner by an engineer. Since open MAS are very dynamic and heterogenous systems, we take the stance that coordination mechanisms must be dynamically composed at run-time. Thus, along the lines of Weyns *et al.* (2009: 5), coordination design must move from programming to flexible composition. Coordination infrastructures are typically the result of the combination of different components: ontologies, protocols, decision procedures, agents, services, etc. In an open world, these components are created by independent engineers at different locations, are combined and recombined to build up more and more complex components, possibly by other engineers, and these complex components are shared again. Once finalised, the resulting coordination design can be enacted to fulfil the objectives of the MAS for which it was designed. Therefore, the on-line decentralised design (by flexible composition) and enactment of coordination infrastructures appears as a future research challenge. Finally, a further refinement of on-line coordination design has to do with agent involvement. Again, in an open world coordination infrastructures should consider how to allow agents to choose their own coordination mechanism, namely their own rules of interaction.

How to engineer a coordination infrastructure: As shown in Table 1, most coordination infrastructures have successfully explored distributed implementations, and yet there is still room for exploration. In particular, P2P systems offer a high degree of decentralisation and openness. Furthermore, there are additional benefits inherent to P2P systems that are potentially interesting for open MAS (e.g. self-organisation, resilience to faults and attacks, low barrier to deployment, etc.). All these features qualify P2P architectures as potential candidates to explore the deployment of coordination infrastructures in open scenarios.

In the rest of the paper, we try to analyse in more depth the open research challenges identified above and we also discuss some promising approaches in the literature to cope with such challenges. In general, we advocate that next generation coordination infrastructures must:

- become *socially aware*, by facilitating human interaction within an MAS;
- increase *decision support* to help agents achieve their own goals; and
- increase *openness* to support on-line, fully decentralised design and execution.

In what follows, Section 3 discusses how to achieve socially awareness, whereas Section 4 proposes means to assist agent decision making. Finally, Section 5 proposes how to engineer a coordination infrastructure in an open world to facilitate decentralised design and execution. In particular, the section investigates how to facilitate the distributed process of creation, combination, sharing, and execution of the components required to enact coordination infrastructures.

3 Challenge: human interaction

The design of hybrid MAS, where both software agents and humans interact, poses new challenges to coordination infrastructures. Humans present different characteristics and capabilities than software agents, hence they require different functionalities. For instance, interacting by exchanging messages in some agent communication language is appropriate for software agents, but not for human users.

Another key aspect when humans become active players in an MAS is how to represent the relevant information about the dynamics of the MAS. Information such as the coordination model, the system state, or the actions performed by other participants is essential to understand what is going on in the MAS or find out the valid actions an agent may perform. This information has to be effectively presented and continuously updated to human users.

Thus, to ‘open’ MAS to humans, coordination infrastructures must be extended by incorporating appropriate tools, services, and interfaces that specifically address human agents’ requirements, which are different than software agents’. Obviously web pages or 2D interfaces can be used to facilitate human participation, but more immersive environments such as 3D virtual worlds can provide a more effective interface to support human participation.

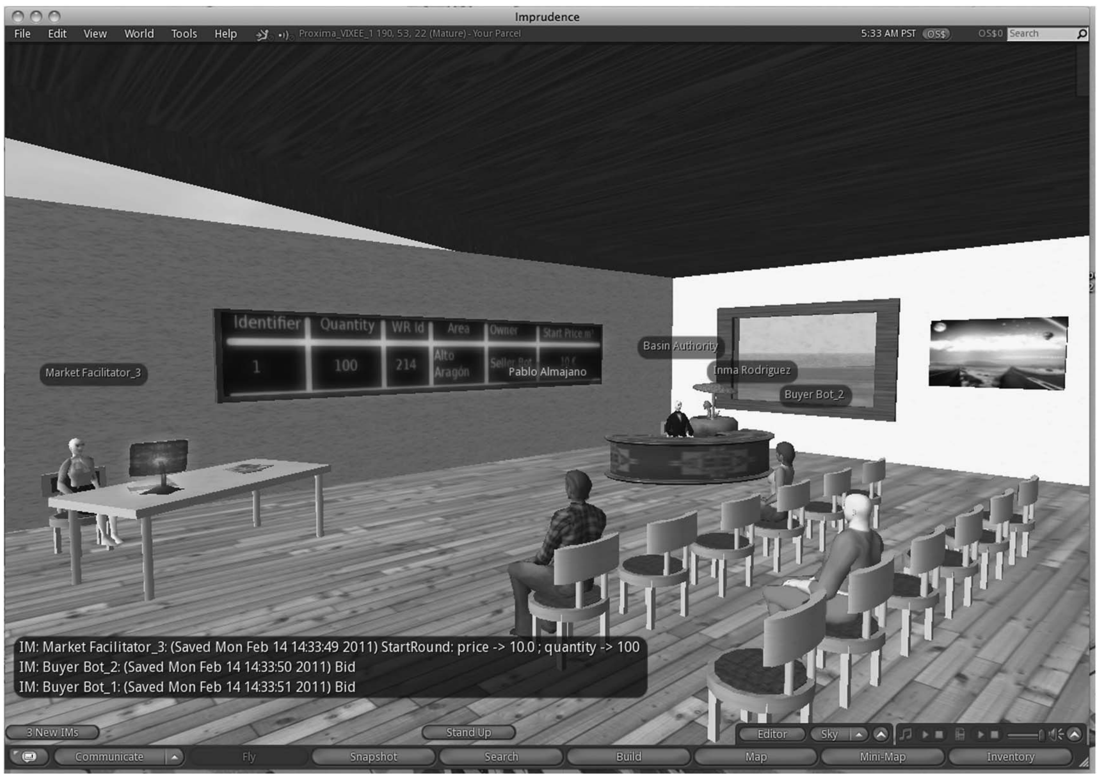
Virtual worlds technology has recently emerged in computing with enormous strength (Messinger *et al.*, 2009). A virtual world is an online immersive environment where, using 3D visualisation, humans participate represented as graphically embodied characters (avatars) and interact with others and the environment by using simple and intuitive control facilities. Because humans are social, the concept of virtual worlds is very appealing to mediate their remote interactions. Nowadays, there are millions of people connecting to virtual worlds every day. Such immersive and interactive environment provides many possibilities to represent the system state and the regulations defined by the coordination model. For instance, the other participants are represented also as avatars and their appearance can be used to display the role they are playing. We argue that 3D virtual worlds can be successfully used to incorporate humans into MAS. To illustrate this hypothesis, next we outline the work carried out to open electronic institutions to humans by using virtual worlds.

Virtual Institutions (Bogdanovych, 2007) is a concept that combines electronic institutions and 3D virtual worlds. The aim of Virtual Institutions is to design regulated environments where both human and software agents can participate. In this context, electronic institutions are used to define the coordination model that structure participants' interactions, while 3D virtual worlds facilitate human participation in the system. Many elements in an electronic institution specification show conceptual similarities with building blocks of virtual worlds: scenes can be visualised as rooms; connections between scenes can be visualised as doors; agents can be visualised as avatars; and the maximum number of participants in a scene can determine the size of its associated room. Thus, the virtual world representation of an electronic institution can be automatically generated from its specification. The most advanced proposal for such generation is the Virtual World Grammar (Trescak *et al.*, 2010), an extension of shape grammars (Stiny, 1980; Trescak *et al.*, 2012) for the automatic generation of a virtual world from an electronic institution specification. Hence, human users participate in the system by controlling an avatar in a virtual world, an automatically generated representation of the electronic institution.

The architecture of a virtual institution consists of three layers: the normative layer, the visual interaction layer, and the communication layer. The normative layer of a virtual institution consists of AMELI, the electronic institutions infrastructure. AMELI is in charge of keeping the system state and of enforcing the institutional rules. Software agents taking part in the system are directly connected to AMELI. The visual interaction layer provides the 3D interface that supports the participation of human users. Between them, the communication layer is in charge of causally connecting AMELI and the corresponding virtual world. The tasks of this layer are to inform AMELI about the actions performed by users through their avatars in the virtual world and to update the visualisation whenever the execution state changes. Hence, it guarantees consistency between the execution state stored by AMELI and the virtual world visualisation of the institution.

The first implemented infrastructure for the execution of virtual institutions used Adobe Atmosphere as a virtual world (Bogdanovych, 2007). The idea was further explored in the context of the Itchy Feet project where a prototype for the tourism domain was developed (Seidel, 2010) using the Torque game engine. Recently in Trescak *et al.* (2011), a new infrastructure for virtual institutions has been proposed, the so-called VIXEE. Among other features, VIXEE supports the connection to several virtual worlds and the dynamic update of the virtual world representation. For example, rooms are created or removed when activities at the institutional level start or finish. Notice that by providing a connection to several virtual worlds, users wearing different devices with different computation capabilities (e.g. smart phones) may participate in several of these virtual worlds at the same time.

Figure 2 displays a snapshot of an auction room of a virtual institution execution supported by VIXEE. The room recreates a real-life auction room, where each buyer sits in a room chair. Avatars representing software agents playing the buyer role are represented as avatars with blue skin, while avatars with green skin represent software agents controlling the auction execution. The panel on the wall represents the information of the current auction round. Its content is automatically updated as bids are made by buyers during an auction round, and as messages are sent by the auctioneer. Notice that users can easily perceive the other participants in the auction room and the role they play, as well as information about the current auction round.



```
IM: Market Facilitator_3: (Saved Mon Feb 14 14:33:49 2011) StartRound: price -> 10.0 ; quantity -> 100
IM: Buyer Bot_2: (Saved Mon Feb 14 14:33:50 2011) Bid
IM: Buyer Bot_1: (Saved Mon Feb 14 14:33:51 2011) Bid
```

Figure 2 Snapshot of a virtual institution execution.

4 Challenge: coordination support services

In this section, we focus on how to provide coordination support to agents participating in an MAS. As pointed out above, in Section 2, MAS coordination infrastructures have supported over time the execution of increasingly complex coordination models. Consequently, participating in an MAS is becoming a more complex task. At design time, it is not trivial for an agent to determine the best policies to achieve her goals. That would require to foresee the consequences of the agent's decisions within the MAS she participates in. For instance, a recent study (Sycara *et al.*, 2010) shows that when planning involves complex reasoning (e.g. in military environments), human users tend to lose track of the norms regulating interactions, thus resulting in plans with a significant number of norm violations.

We advocate that MAS coordination infrastructures should incorporate services with the aim of assisting participants to successfully achieve their goals, that is, services focusing on assisting coordination among participants. Failing to provide adequate assistance to users may lead to the failure of an MAS. This approach is also usual in human organisations that provide services and devote some resources to assist their users. Likewise we regard coordination assistance as a key requirement to help participants, either human or software agents, in MAS. A 3D interface such as the one described in Section 3 can be used to provide humans with assistance services.

We differentiate information-based services, for example, providing agents with the necessary information about the coordination model and the system state, from assistance-based services, for example, proposing plans to achieve a given goal. Our view is that assistance-based services still constitute an open challenge to the research community, since they represent a step beyond providing the information-based services offered by current coordination infrastructures. Along this line, Campos *et al.* (2009) propose the following taxonomy of assistance services:

- *Information-based* services providing agents with relevant information to successfully participate in the system. Basic information services should provide information about both the current coordination model and system state. Information can be provided pro-actively by the infrastructure, as for instance, when the coordination model is adapted, or after a request of an agent. Currently, there are some infrastructures that already provide information-based services. For instance, agents coordinated by S-Moise+ are informed whenever they acquire new obligations, while AMELI informs participants whenever an agent joins or leaves an interaction protocol.

- *Explanation-based* services describing the consequences of agents' actions. For instance, a service explaining why an agent is not allowed to perform certain action, or why an agent has acquired a new obligation. Explanation-based services complement information-based services providing valuable information that justifies the current state of the MAS and that helps agents adapt their decisions by analysing the causes of concurrent actions performed in the MAS.
- *Advice-based* services suggesting plans to participants to achieve their goals. The aim of these services is not to find optimal planning solutions, but instead to support an agent's plan by identifying and suggesting alternatives to weaknesses of the plan. For instance, a service can provide agents with the steps to follow to register an item in an electronic auction house. A more complex advice-based service is proposed in Oh *et al.* (2011) and Oh *et al.* (2011). These works propose an assistant service that helps the reasoning of human participants considering the norms of an MAS. The service recognises an agent's plans and reasons about the assistance she will require in the execution of such plans. The proposed architecture incorporates mechanisms to identify user intentions, normative reasoning, and planning integrating probabilistic plan reasoning with normative reasoning. The notion of proactive normative reasoning is proposed to predict the probability with which users will violate norms, allowing the system to take remedial actions before norms are actually violated.
- *Assessment-based* services allowing an agent to estimate the consequences of performing certain actions. Sometimes, it may be interesting to evaluate the consequences a given action before performing it. Assessment-based services can provide feedback such as the validity of some action given the current state of the MAS, the state that the MAS would reach if the action is executed, or the consequences for the agent (e.g. if the agent would acquire a new obligation after performing the action).

5 Challenge: decentralised architectures for design and execution of co-ordination infrastructures

Coordination infrastructures are typically the result of the combination of different components: ontologies, protocols, decision procedures, agents, services, etc. In an open world, these components are created by independent engineers at different locations, are combined and recombined to build up more and more complex components, possibly by other engineers, and these complex components are shared again. Once finalised, these infrastructures can be enacted to fulfil the objectives for which they were designed.

A big challenge is how to facilitate the distributed process of creation, combination, sharing, and execution of these components. We think that a distributed architecture seems the most appropriate approach, as it naturally matches the distributed character of the design process. Distributed systems are certainly more difficult to engineer than centralised systems. However, the number of advantages that they offer (robustness, scalability, security, or increased privacy) make them an attractive approach to build systems over networks like the Internet. The success of Grid and Cloud computing is based in part on this possibility of distributing the resources over a network and dynamically handling the changing needs of the users.

In particular, we think that P2P networks are a good candidate to implement the functionalities required for the construction of co-ordination infrastructures. The essence of P2P computing is that peers, when active, exchange services. In the context of a P2P network for the construction of co-ordination infrastructures, a peer could thus be requested by any other peer in the network to share components and execute services or agents. P2P systems offer many appealing features: a high degree of decentralisation, self-organisation, low barrier to deployment (compared with client-server systems), organic growth, resilience to faults and attacks, and abundance and diversity of resources (Rodrigues & Druschel, 2010).

These properties enable user community creation, facilitates the sharing of resources, and when created appropriately (i.e. by giving incentives to their usage) permits an explosion in usage.

A P2P node would then provide several key functionalities (i.e. services) for the distributed creation and execution of co-ordination infrastructures:

- *Communication.* These services would allow the node to exchange information with other known nodes in the network. In particular, they would allow for the nodes to exchange co-ordination components: specifications and agents' code. For agents and services being executed at the node these services would allow to communicate with agents and services being executed at other nodes in the network. This set of services implements the distributed execution of co-ordination infrastructures, in particular, they should include a Discovery Service to find content over the P2P network. The Discovery Service provides a lookup service and can be implemented by a DHT (Distributed hash table). This service is used to publish the components and services that the peers want to share.
- *Storage.* Every node could potentially store specifications and code (implementations of agents and services) that may be made available (published) to other nodes in the network. Each piece of code should be accompanied by a minimum description such as: its functionality, its authors, a certificate, its version, etc, so it can be found by the search services of other nodes. The agent implementation and this description could be bundled into for example, a zip file or a jar file. The descriptions should be provided either by the user of the node or imported from other nodes in the network. These storage services may include tracking for changes in the source node of the content. The storage services are in charge of publishing into the Discovery Service: (i) the shared co-ordination components specifications available for download; (ii) the shared agent implementations available for download; and (iii) the (type of) services and agents that the peer agrees to run.
- *Search.* A human user interacting with a node of the network may be specifying new components for which construction she may want to reuse existing components published in other nodes (e.g. protocols, ontologies). The search services allow to explore the P2P network for these components. Trust and reputation services might be included to rank the quality of the components and semantic services might be used to do approximate searches over the network.
- *Computing.* These services would sustain the execution of agents and services at a node. Services to launch and monitor agents are key for the distributed execution of co-ordination infrastructures. The launched agents would use the communication services of the node when they want to communicate with other agents executing either locally at this node or at other nodes of the network. Such a P2P network could also allow for the on-line co-ordination of agents, as they might have access to the complete set of P2P network resources at runtime and thus could search for potential co-ordination mechanisms, vote on which one to use and then enact it. This possibility would support requirement (3) in Section 2.2. Also, the distributed nature of a P2P network favours openness and scalability as required in Section 2.2.

A number of available middleware platforms could be used as the basis for node deployment. For example, Tapestry (Tap, n.d.) provides self-organising routing and an object location system. It helps recovering from failures with a mechanism that caches content. Chord (Stoica *et al.*, 2001) implements a purely decentralised P2P system where the number of known nodes is logarithmic on the size of the network. Search queries become binary searches over the network and thus have logarithmic complexity. New approaches are more resilient to attacks by implementing object replication and multi-path backups (e.g. Butterlfly (Datar, 2002)).

6 Conclusions

Coordination infrastructures play a central role in the engineering of MAS. They facilitate the deployment of the system and provide the services that agents need to coordinate. In this paper, we have reviewed the state-of-the-art on coordination infrastructures for MAS with the aim of identifying open research challenges that next generation coordination infrastructures must tackle. First, we analysed and compared how the contributions in the literature have fulfilled the requirements that the engineering of coordination infrastructures poses. Our analysis lead to a taxonomy of coordination infrastructures.

First-generation coordination infrastructures were *ad hoc infrastructures* developed to run particular MAS applications, and hence they were not aimed at being reused to run multiple MAS applications. Second-generation coordination infrastructures were spurred by the need for developing *general-purpose* infrastructures that could serve to develop a wide variety of MAS applications, hence reducing development time and cost. Nonetheless, we observe that these infrastructures particularly focused on providing interaction support for agent to agent interaction and engineering distributed architectures. Third-generation coordination infrastructures go beyond second-generation coordination infrastructures along several directions. First, this type of infrastructures turns their attention to *social* coordination models that offer a higher level of abstraction and further expressiveness than the interaction protocols exploited by first-generation infrastructures. Second, the infrastructures in this group become situated by supporting the interaction with services and environments. Third, these infrastructures start incorporating *intelligence* to allow an MAS to dynamically adapt to unpredictable changes.

Therefore, coordination infrastructures have evolved from *ad hoc* infrastructures (first generation), to low-level infrastructures (second generation) mostly concerned with communication and distribution, to higher level infrastructures (third generation) that embrace social coordination models, embed intelligence to support adaptiveness, and are situated. Furthermore, several conclusions stem from our analysis of the literature regarding open research challenges:

- Helping humans interact within an MAS remains a rather unexplored, intricate matter. The lack of support for human interaction impedes the realisation of hybrid MAS that seamlessly integrate human and software agents. Next generation coordination infrastructures must become *socially aware*, by facilitating human interaction within an MAS.
- Although current MAS developments have taught us how arduous and intricate is to develop software agents that interact within an MAS, current research on coordination infrastructures has not considered how to assist agents in their decision making. Thus, providing *decision support* to help agents reduce the scope of reasoning with the aim of achieving their goals is an open issue for next generation coordination infrastructures.
- The vast majority of research on coordination infrastructures, with the exception of Aldewereld *et al.* (2010) and Vazquez-Salceda *et al.* (2010), has focused on the off-line design of coordination in a centralised manner. Since open MAS are very dynamic and heterogenous systems, we take the stance that coordination mechanisms must be dynamically composed at run-time. Thus, along the lines of Weyns *et al.* (2009: 5), coordination design must move from programming to flexible composition. Therefore, the on-line decentralised design (by flexible composition) and enactment of coordination infrastructures appears as a future research challenge. Furthermore, coordination infrastructures should also consider how to allow agents to choose their own coordination mechanism, namely their own rules of interaction. To summarise, next generation coordination infrastructures must increase *openness* to support on-line, fully decentralised design and execution.

We have also identified some promising approaches in the literature, together with the research issues worth investigating, to cope with such challenges. First, we identified 3D Virtual Worlds as an appropriate technology to support human interaction, and hence achieve social awareness. Second, we identified a taxonomy of assistance services that include services beyond the information-based services currently offered coordination infrastructures. Implementing and incorporating such assistance services into next generation coordination infrastructures would significantly help agents achieve their goals. Third, we have identified P2P architectures as a good candidate to implement the functionalities required for the construction of co-ordination infrastructures that support on-line, decentralised design, and execution.

Acknowledgements

This work was funded by AT (CONSOLIDER CSD2007-0022), EVE (TIN2009-14702-C02-01, TIN2009-14702-C02-02), COR (TIN2012-38876-C02-01, TIN2012-38876-C02-02), MECER (201250E053) and the Generalitat of Catalunya (2009-SGR-1434).

References

Aldewereld, H. & Dignum, V. 2010. Operetta: organization-oriented development environment. In *Languages, Methodologies, and Development Tools for Multi-Agent Systems—Third International Workshop, LADS 2010, August 30–September 1, Revised Selected Papers,* Dastani, M., Seghrouchni, A. E. F., Hubner, J. & Leite, J. (eds). Lecture Notes in Computer Science **6822**, 1–18. Springer Berlin Heidelberg.

Aldewereld, H., Padget, J., Vasconcelos, W., Vazquez-Salceda, J., Sergeant, P. & Staikopoulos, A. 2010. Adaptable, organization-aware, service-oriented computing. *IEEE Intelligent Systems* **25**, 26–35.

Arcos, J. L., Esteva, M., Noriega, P., Rodriguez-Aguilar, J. A. & Sierra, C. 2005. Engineering open environments with electronic institutions. *Engineering Applications of Artificial Intelligence* **18**, 191–204.

Arcos, J. L., Noriega, P., Rodriguez-Aguilar, J. A. & Sierra, C. 2006. E4mas through electronic institutions. In *Environments for Multi-Agent Systems III, Third International Workshop, E4MAS 2006, Hakodate, Japan, May 8, Selected Revised and Invited Papers,* Dastani, M., Seghrouchni, A. E. F., Hubner, J. & Leite, J. (eds). Lecture Notes in Computer Science **4389**, 184–202. Springer Berlin Heidelberg.

Arcos, J. L., Rodriguez-Aguilar, J. A. & Rosell, B. 2008. Engineering Autonomic Electronic Institutions. *Engineering Environment-Mediated Multi-Agent Systems, Lecture Notes in Computer Science* **5049**, 76–87.

Artikis, A., Kaponis, D. & Pitt, J. 2009. Dynamic specifications of norm-governed systems. In *Multi-Agent Systems: Semantics and Dynamics of Organisational Models*, Dignum, V. (ed.). IGI Global, 460–479.

Baumer, C., Breugst, M., Choy, S. & Magedanz, T. 1999. Grasshoppera universal agent platform based on omg masif and fipa standards. In *First International Workshop on Mobile Agents for Telecommunication Applications (MATA99)*, 1–18.

Bellifemine, F., Poggi, A. & Rimassa, G. 2001. Developing multi-agent systems with JADE. In *Intelligent Agents VII Agent Theories Architectures and Languages*, Dastani, M., Seghrouchni, A. E. F., Hubner, J. & Leite, J. (eds). LNCS, **1986**, 42–47. Springer Berlin Heidelberg.

Bogdanovych, A. 2007. *Virtual Institutions*, PhD thesis, University of Technology.

Boissier, O. & Sichman, J. S. 2004. Organization oriented programming, OOP. AAMAS 2004 Tutorial.

Bordini, R. H., Dastani, M., Dix, J. & El Fallah Seghrouchni, A. (eds) 2005a. Multi-Agent Programming: Languages, Platforms and Applications, Volume 15 in the Multiagent systems, artificial societies, and simulated organizations series. Springer US.

Bordini, R. H., Hubner, J. F. & Vieira, R. 2005b. Jason and the golden fleece of agent-oriented programming. In *Multi-Agent Programming: Languages, Platforms and Applications*, Bordini, R. H., Dastani, M., Dix, J. & El Fallah Seghrouchni, A. (eds), 3–37. Springer-Verlag.

Brazier, F. M. T., Kephart, J. O., Parunak, H. V. D. & Huhns, M. N. 2009. Agents and service-oriented computing for autonomic computing: a research agenda. *IEEE Internet Computing* **13**, 82–87.

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F. & Mylopoulos, J. 2004. Tropos: an agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems* **8**, 203–236.

Caire, G., Gotta, D. & Banzi, M. 2008. Wade: a software platform to develop mission critical applications exploiting agents and workflows. In *Proceedings of the 7th international Joint Conference on Autonomous Agents and Multiagent Systems: Industrial Track, AAMAS '08*, 29–36.

Campos, J., Esteva, M., Lopez-Sanchez, M., Morales, J. & Salamo, M. 2011. Organisational adaptation of multi-agent systems in a peer-to-peer scenario. *Computing* **91**, 169–215.

Campos, J., Lopez-Sanchez, M. & Esteva, M. 2009. Coordination support in mas. In *8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 09)*, 1301–1302.

Cohen, P. R. & Levesque, H. J. 1991. Teamwork. *Nous* **25**, 487–512.

Datar, M. 2002. Butterflies and peer-to-peer networks. In *Algorithms ESA*, Mohring, R., Raman, R. (eds). LNCS, **2461**, 201–222. Springer Berlin/Heidelberg.

Dignum, V., Dignum, F., Furtado, V., Melo, A. & Sonenberg, L. 2005. Towards a simulation tool for evaluating dynamic reorganization of agents societies. *Proceedings of Workshop on Socially Inspired Computing, AISB Convention* **230**, 153–162.

Dyke Parunak, H., Brueckner, S. & Sauter, J. 2005. Digital pheromones for coordination of unmanned vehicles. In *Environments for Multi-Agent Systems*, Weyns D., Dyke Parunak H. & Michel F. (eds), LNCS, **3374**, 246–263. Springer.

Esteva, M., Rosell, B., Rodriguez-Aguilar, J. A. & Arcos, J. L. 2004. Ameli: an agent-based middleware for electronic institutions. *International Joint Conference on Autonomous Agents and Multiagent Systems* **1**, 236–243.

FIP 2002. FIPA Abstract Architecture Specification. `http://www.fipa.org/specs/fipa00001/`.

Gasser, L., Bragamza, C. & Herman, N. 1987. MACE: a flexible testbed for distributed AI research. In *Distributed Artifical Intelligence*, Huhns M. (ed.) Pitmanand Morgan Kaufmann, 119–152.

Gutknecht, O., Ferber, J. & Michel, F. 2001. Integrating tools and infrastructures for generic multi-agent systems. In *Proceedings of the Fifth International Conference on Autonomous Agents (Agents'01)*, 441–448.

Horling, B., Benyo, B. & Lesser, V. 2001. Using self-diagnosis to adapt organizational structures. In *Proceedings of the 5th International Conference on Autonomous Agents*, 529–536.

Howden, N., Ronnquist, R., Hodgson, A. & Lucas, A. 2001. Jack intelligent agents—summary of an agent infrastructure. In *Proceedings of the 5th International Conference on Autonomous Agents (AGENTS'01).*

Hubner, J., Sichman, J. & Boissier, O. 2002. A model for the structural, functional, and deontic specification of organizations in multiagent systems. *Advances in Artificial Intelligence* **2507**, 439–448.

Hubner, J. F., Sichman, J. S. & Boissier, O. 2004. Using the Moise+ for a cooperative framework of mas reorganisation, LNAI. *Proceedings of the 17th Brazilian Symposium on Artificial Intelligence (SBIA'04)*, **3171**, 506–515. Springer.

Hubner, J. F., Sichman, J. S. & Boissier, O. 2005. S-MOISE+: a middleware for developing organised multi-agent systems. In *Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems*, Boissier, O., Padget, J., Dignum, V., Lindemann, G., Matson, E., Ossowski, S., Sichman, J. S. & Vazquez-Salceda, J. (eds). LNCS, **3913**, 64–78. Springer.

Jennings, N. R., Sycara, K. & Wooldridge, M. 1998. A roadmap of agent research and development. *Autonomous Agents and Multi-agent Systems* **1**, 275–306.

Kitio, R., Boissier, O., Hubner, J. F. & Ricci, A. 2007. Organisational artifacts and agents for open multi-agent organisations: "giving the power back to the agents". In *COIN*, Sichman J. S., Padget J. A., Ossowski S. & Noriega P. (eds). LNCS, **4870**, 171–186. Springer.

Kota, R., Gibbins, N. & Jennings, N. 2008. Decentralised structural adaptation in agent organisations. In *AAMAS Workshop Organised Adaptation in MAS*, 54–71.

Labrou, Y. & Finin, T. 1997. A proposal for a new KQML specification. Technical Report TR CS-97-03, Computer Science and Electrical Engineering Department, University of Maryland.

Messinger, P. R., Stroulia, E., Lyons, K., Bone, M., Niu, R. H., Smirnov, K. & Perelgut, S. 2009. Virtual worlds—past, present, and future: new directions in social computing. *Decision Support Systems* **47**, 204–228.

Oh, J., Meneguzzi, F. & Sycara, K. 2011. Probabilistic plan recognition for intelligent information agents. *Proceedings of ICAART* **2**, 281–287.

Oh, J., Meneguzzi, F., Sycara, K. & Norman, T. 2011. Prognostic agent assistance for norm-compliant coalition planning. In *The Second International Workshop on Infrastructures and Tools for multiagent systems*, 126–140.

Parsons, S., Rodriguez-Aguilar, J. A. & Klein, M. 2011. Auctions and bidding: a guide for computer scientists. *ACM Computing Surveys* **43**, 10.

Pattison, H. E., Corkill, D. D. & Lesser, V. R. 1987. Instantiating Descriptions of Organizational Structures. In *Distributed Artificial Intelligence, Research Notes in Artificial Intelligence*, Huhns M. N. (ed.) **I**. Pitman Publishers, 59–96.

Platon, E., Sabouret, N. & Honiden, S. 2006. Environmental support for tag interactions. In *E4MAS*, Weyns D., Parunak H. V. D. & Michel F. (eds). LNCS, **4389**, 106–123. Springer.

Pokahr, A., Braubach, L. & Lamersdorf, W. 2005. Jadex: a bdi reasoning engine. In Bordini, R. H., Dastani, M., Dix, J., and El Fallah Seghrouchni, A., 149–174.

Poslad, S., Buckle, P. & Hadingham, R. 2000. The fipa-os agent platform: open source for open standards. In *Proceedings of the 5th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents*, 355–368.

Rao, A. S. & Georgeff, M. P. 1998. Modeling rational agents with a bdi-architecture. In *Readings in agents*, Huhns M. N. & Singh M. P. (eds). Morgan Kaufmann Publishers Inc., 317–328.

Ricci, A., Piunti, M. & Viroli, M. 2011. Environment programming in multi-agent systems–an artifact-based perspective. *Autonomous Agents and Multi-Agent Systems* **23**, 158–192.

Rodrigues, R. & Druschel, P. 2010. Peer-to-peer systems. *Communications of the ACM* **53**, 72–82.

Rodriguez-Aguilar, J. A., Noriega, P., Sierra, C. & Padget, J. 1997. Fm96.5 a java-based electronic auction house. In *Second International Conference on The Practical Application of Intelligent Agents and Multi-Agent Technology: PAAM97*, 207–224.

Sacchi, G., Marando, A., Quarantotto, E. & Caire, G. 2011. Wade tutorial. Defining agents as workflows, WOLF. Tutorial.

Sandholm, T. 2002. Emediator: a next generation electronic commerce server. *Computational Intelligence* **18**, 656–676.

Searle, J. R. 1969. *Speech Acts*. Cambridge University Press.

Seidel, I. 2010. *Engineering 3D Virtual World Applications Design, Realization and Evaluation ofa 3D e-Tourism Environment*. PhD thesis, Technischen Universitat Wien Fakultat fur Informatik.

Serugendo, G. D. M., Gleizes, M. P. & Karageorgos, A. 2006. Self-organisation and emergence in mas: an overview. *Informatica (Slovenia)* **30**, 45–54.

Shoham, Y. 1993. Agent-oriented programming. *Artificial Intelligence* **60**, 51–92.

Stiny, G. 1980. Introduction to shape and shape grammars. *Environment and Planning B* **7**, 343–351.

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F. & Balakrishnan, H. 2001. Chord: a scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM '01 Conference*.

Stone, P. & Veloso, M. 2000. Multiagent systems: a survey from a machine learning perspective. *Autonomous Robots* **8**, 345–383.

Sycara, K. P. 1998. Multiagent systems. *AI Magazine* **19**, 79–92.

Sycara, K., Norman, T., Giampapa, J., Kollingbaum, M., Burnett, C., Masato, D., McCallum, M. & Strub, M. 2010. Agent support for policy-driven collaborative mission planning. *The Computer Journal* **53**, 528–540.

Tap n.d.. Tapestry `http://tapestry.apache.org`.

Trescak, T., Esteva, M. & Rodriguez, I. 2010. A virtual world grammar for automatic generation of virtual worlds. *The Visual Computer* **26**, 521–531.

Trescak, T., Esteva, M. & Rodriguez, I. 2011. Vixee an innovative communication infrastructure for virtual institutions. In *Proceedings of The 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 10),* 1131–1132.

Trescak, T., Esteva, M. & Rodriguez, I. 2012. A shape grammar interpreter for rectilinear forms. *Journal of Computer-Aided Design* **44**, 657–670. Elsevier.

Vazquez-Salceda, J., Vasconcelos, W. W., Padget, J., Dignum, F., Clarke, S. & Palau Roig, M. 2010. Alive: an agent-based framework for dynamic and robust service-oriented applications. In *Proceedings of 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, 1637–1638.

Weyns, D., Haesevoets, R., Helleboogh, A., Holvoet, T. & Joosen, W. 2010. The macodo middleware for context-driven dynamic agent organizations. *ACM Transactions on Autonomous Adaptative Systems* **5**, 3:1–3:28.

Weyns, D., Helleboogh, A., Holvoet, T. & Schumacher, M. 2009. The agent environment in multi-agent systems: a middleware perspective. *Multiagent and Grid Systems* **5**, 93–108.

Weyns, D., Omicini, A. & Odell, J. 2007. Environment as a first class abstraction in multiagent systems. *Autonomous Agents and Multi-Agent Systems* **14**, 5–30.

Wooldridge, M. 2002. *An introduction to multiagent systems*, Wiley.

Wurman, P. R., Wellman, M. P. & Walsh, W. E. 1998. The michigan internet auctionbot: a configurable auction server for human and software agents. In *Proceedings of the second international conference on Autonomous agents*, Sycara, K. P. & Wooldridge, M. (eds). *AGENTS '98*, 301–308. ACM.

Zambonelli, F., Jennings, N. R. & Wooldridge, M. 2003. Developing multiagent systems: the gaia methodology. *ACM Transactions on Software Engineering and Methodology* **12**, 317–370.

Zhang, C., Abdallah, S. & Lesser, V. 2009. Integrating organizational control into multi-agent learning. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, 757–764.