An Interaction-oriented Agent Framework for Open Environments

Matteo Baldoni[1], Cristina Baroglio[1], Federico Bergenti[4], Elisa Marengo[1], Viviana Mascardi[3], Viviana Patti[1], Alessandro Ricci[2], and Andrea Santi[2]

[1] Università degli Studi di Torino,
{baldoni,baroglio,emarengo,patti}@di.unito.it
[2] Università degli Studi di Bologna,
{a.ricci,a.santi}@unibo.it
[3] Università degli Studi di Genova, {mascardi}@disi.unige.it
[4] Università degli Studi di Parma
{federico.bergenti}@unipr.it

Abstract. The aim of the work is to develop formal models of interaction and of the related support infrastructures, that overcome the limits of the current approaches. We propose to represent explicitly not only the agents but also the computational environment in terms of rules, conventions, resources, tools, and services, that are functional to the coordination and cooperation of the agents. These models will enable the verification of the interaction in the MAS, thanks to the introduction of a novel social semantics of interaction based on commitments and on an explicit account of the regulative rules.

Keywords: Commitment-based protocols, High-level environment models, Direct and mediated communication, Agent-oriented infrastructure

1 Introduction

The growing pervasiveness of computer networks and of Internet is an important catalyst pushing towards the realization of *business-to-business*, *cross-business solutions* or, more generally, of *open environment systems*. Interaction, coordination, and communication acquire in this context a special relevance since they allow the involved groups, often made of heterogeneous and antecedently existing entities, to integrate their capabilities, to interact according to some agreed contracts, to share best practices and agreements, to cooperatively exploit resources and to facilitate the identification and the development of new products. Multi-Agent Systems (MASs) seem to be the most proper abstraction for developing cross-business systems, since they share with them the same characteristics (autonomy, heterogeneity) and the same issues (interaction, coordination, communication). These issues received a lot of attention by the research community working on agent models and platforms but, in our opinion, no proposal tackles them in a seamless and integrated way.

The first limit of existing agent frameworks and platforms concerns the *forms of allowed interactions*. Most of them, such as [8, 9], only supply the means for

realizing agent interaction through *direct communication* (message exchange). This feature is common to all those approaches which foresee *agents* as the only available abstraction, thereby leading to message exchange as the only natural way agents have to interact. However, think of a business interaction between two partners. Once the client has paid for some item into the paypal account of the merchant, it should not be necessary that the client also sends a message to the merchant about the payment. Indeed, it is sufficient for the merchant to check his/her account to be informed about the transaction. The interaction already occurred.

In general, indirect communication fosters the collaboration and the coordination inside open systems, in that it allows anonymous, non-specialized interfaces that invite participation of new knowledge sources [21]. Indirect communication can be realized by means of persistent observable state changes, as it is for instance done by stigmergic approaches. In the literature there are models that allow the MAS designer to cope with a wider range of communication kinds by explicitly providing abstractions that support also the realization of *indirect forms* of interaction. In particular, the Agents & Artifacts model (A&A) [44, 28, 38] provides the explicit abstractions of *environment* and *artifact*, that are entities which can be acted upon, observed, perceived, notified, and so forth. Unfortunately, the A&A *misses a semantics* of communication, which would instead be required to achieve the seamless and coherent integration of the interaction, coordination, communication issues that we look forward. Other frameworks lack a satisfactory *semantics of interaction*. For instance, many of them, e.g. [8, 9], adopt a *mentalistic* semantics which does not allow the agents, that are involved in the interaction, to verify that their partners respect the agreements. This, however, is a crucial aspect when one models business relationships.

In this respect, one interesting possibility would be to use an approach based on a *social* and *observational* semantics, like the commitment-based approach [30]. In this context, the agents' own behaviors remain private but agents agree on the meaning of a set of social actions: since interactions are observable and their semantics is shared, each agent is able to draw conclusions concerning the behavior of their partners as well as of the system as a whole. The advantage is that it becomes possible to detect violations to the agreed interaction and identify responsibilities. Proposals of this kind, that overcome the limitations of a pure mentalistic approach, seem particularly suitable to give to the A&A meta-model the semantics of communication that it still misses. Moreover, an interaction-oriented framework must supply the means for representing laws, rules, habits, which have strong implications on how agents can interact. These can be modeled by means of temporal regulations, along the line of [6, 22].

This paper proposes a new agent-programming framework, Mercurio, that supports interaction-oriented computing and is currently under development. The paper is organized as follows. Section 2 describes the proposal. Section 3 discusses how the proposal advances the state of art. Final remarks in Section 4 conclude the paper.

2 The Mercurio framework

Open systems are made of heterogeneously designed and pre-existing parties that are assembled with some aim, which none of them can pursue alone. In order to allow for a fruitful cooperation, the interaction that each agent carries on with the others, in the context of the assembled system, must respect some rules. In other words, the regulation of interaction is a decisive factor. The proposals that can be found in the literature, concerning the formation of and the interaction within decentralized structures, are still incomplete. For instance, electronic institutions [15, 2] regulate interaction, tackle open environments, and their semantics allows the verification of properties but they only cope with direct communication protocols, based on speech acts. On the other hand, most of the models and architectures for "environments", which also allow indirect communications, prefigure simple/reactive agent models without defining semantics that are comparable to the ones for ACL. This lack hinders agents to reason about their partners' actions and to detect possible violations to the agreed behaviors [25]. In general, no proposal can yet capture all the requirements posed by interaction-oriented computation in open environments. Here, in fact, it is necessary to coordinate autonomous and heterogeneous agents and it is not possible to assume mutual trust among them. It is necessary to have an unambiguous semantics allowing the verification of interaction properties both before the interaction takes place [35] as well as during the interaction [1], preserving at the same time the privacy of the implemented policies.

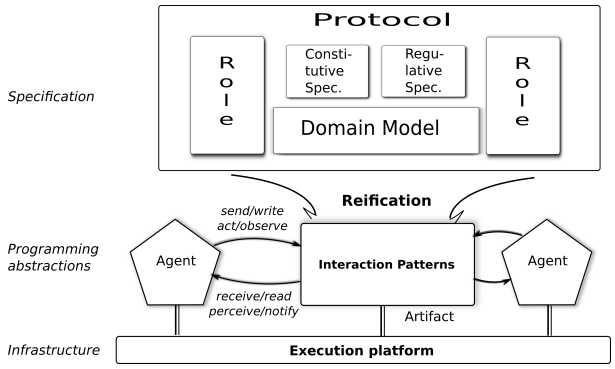

Fig. 1. The Mercurio architecture.

Figure 1 draws the overall picture of the proposal. We distinguish three levels: the *specification* level, the level of the *programming abstractions*, and the *infrastructure*. As we will see, this setting enables forms of verification that encompass both global interaction properties and specific agent properties, such as interoperability and conformance [4].

2.1 Specification Level

The specification level allows the designer to shape the interactions that will characterize the system. Open systems involve autonomous partners with heterogeneous software designs and implementations; hence, the need of identifying high-level abstractions that allow modeling them in a natural way. In order to minimize the effort needed to define proper interfaces and to minimize the altering of internal implementations, Telang and Singh [42] propose that such abstractions should capture the contractual relationships among the partners, which are well-known to the business analysts and motivate the specification of the processes, and identify in commitment-based approaches [41, 46] the appropriate features for performing this task. These, in fact, allow capturing the business intent of the system, leaving aside implementative issues. Other existing approaches (e.g. BPEL, WS-CDL) rely on the specification of control and business flows, imposing unnecessarily restrictive orderings of the interactions but supply the means for capturing neither the business relationships nor the business intent. This limitation is highlighted also by authors like Pesic and van der Aalst [31, 26], who, in particular, show the advantages of adopting a declarative rather than a procedural representation.

By relying on an *observational semantics*, commitment-based approaches can cope both with direct forms of communication (by supporting the implementation of communicative acts), and with forms of interaction, that are mediated by the environment (by supporting the implementation of non-communicative acts, having a social meaning). Agents can not only send and receive messages but they can also act upon or perceive the social state.

Traditional commitment-based approaches, nevertheless, do not suit well those situations where the evolution of the social state is constrained by conventions, laws, and the like, because they do not supply the means for specifying legal *patterns of interaction*. This kind of constraints characterizes, however, many practical situations. Recent proposals, like [6, 22], solve the problem by enriching commitment protocols with temporal regulations. In particular, [6] proposes a decoupled approach that separates a constitutive and a regulative specification [40]. A clear separation brings about many advantages, mostly as a consequence of the obtained modularity: easier re-use of actions, easier customization, easier composition. Roughly speaking, constitutive rules, by identifying certain behaviors as foundational of a certain type of activity, create that activity. They do so by specifying the semantics of actions. Regulative rules contingently constrain a previously constituted activity. In other words, they rule the "flow of activity", by capturing some important characteristics of how things should be carried on in *specific contexts* of interaction [12].

In [6] the constitutive specification defines the meaning of actions based on their effects on the social state, the regulative specification reinforces the regulative nature of commitment by adding a set of behavioral rules, given in terms of *temporal constraints* among commitments (and literals). These constraints define schemes on how commitments must be satisfied, regulating the evolution of the social state independently from the executed actions: the constitutive speci-

fication defines *which* engagement an agent has to satisfy, whereas the regulative specification defines *how* it must achieve what promised. Since interactions are observable and their semantics is shared, each agent is able to draw conclusions concerning the behavior of the partners or concerning the system as a whole. The specification level of Mercurio relies on the representation described in [6] for defining the set of legal interactions.

2.2 Programming Abstractions Level

This level realizes at a programming language level the abstractions defined above. This is done by incorporating interaction protocols based on commitments, patterns of interaction, forms of direct and indirect communication and coordination between agents (such as stigmergic coordination) inside the programmable environments envisaged by the A&A meta-model [44, 28, 38]. The resulting programmable environments will provide flexible *communication channels* that are specifically suitable for open systems. *Agents*, on the other hand, are the abstraction used to capture the interacting partners.

The notion of environment has always played a key role in the context of MAS; recently, it started to be considered as a first-class abstraction useful for the design and the engineering of MAS [44]. A&A follows this perspective, being a meta-model rooted upon Activity Theory and Computer Support Cooperative Work that defines the main abstractions for modeling a MAS, and in particular for modeling the environment in which a MAS is situated. A&A promotes a vision of an endogenous environment, that is a sort of software/computational environment, part of the MAS, that encapsulates the set of tools and resources useful/required by agents during the execution of their activities. A&A introduces the notion of artifact as the fundamental abstraction used for modeling the resources and the tools that populate the MAS environment. In particular, the fact of relying on computational environments provides features that are important from a Software Engineering point of view:

abstraction **-** the main concepts used to define application environments, i.e. artifacts and workspaces, are first-class entities in the agents world, and the interaction with agents is built around the agent-based concepts of action and perception (use and observation);

modularity and encapsulation **-** it provides an explicit way to modularize the environment, since artifacts are components representing units of functionality, encapsulating a partially-observable state and operations;

extensibility and adaptation **-** it provides a direct support for environment extensibility and adaptation, since artifacts can be dynamically constructed (instantiated), disposed, replaced, and adapted by agents;

reusability **-** it promotes the definition of types of artifact that can be reused, such as in the case of coordination artifacts empowering agent interaction and coordination, blackboards and synchronizers.

2.3 Infrastructure Level

In the state of the art numerous applications of the endogenous environments, i.e. environments used as a computational support for the agents' activities, have been explored, including coordination artifacts [29], artifacts used for realizing argumentation by means of proper coordination mechanisms [27], artifacts used for realizing stigmergic coordination mechanisms [36], organizational artifacts [19, 33]. Our starting point is given by works that focus on the integration of agent-oriented programming languages with environments [37] and by the CArtAgO framework [39]. The CArtAgO framework provides the basis for the engineering of MAS environments on the base of: (i) a proper computational model and (ii) a programming model for the design and the development of the environments on the base of the A&A meta-model.

Even if CArtAgO can be considered a framework sufficiently mature for the concrete developing of software/computational MAS environments it can not be considered "complete" yet. Indeed at this moment the state of the art and in particular the CArtAgO framework are still lacking: (i) a reference standard on the environment side comparable to the existing standards in the context of the agents direct communications (FIPA ACL), (ii) the definition of a rigorous and formal semantics, in particular related to the artifact abstraction, (iii) an integration with the current communication approaches (FIPA ACL, KQML, etc.), and finally (iv) the support of semantic models and ontologies.

The infrastructure level will be developed by taking as reference JaCaMo http://jacamo.sourceforge.net, which is a platform for *multi-agent programming*, that successfully integrates BDI agents (developed in Jason [10]), artifact-based environments (CArtAgO) and organisations (specified by MOISE [20]).

2.4 Mediation as a foundation for interaction

One of the tasks, usually assigned to the environment, is that of *medium* of communication [44, 28]. In our opinion, a programmable environment could play the important role of *arbitrator*. In other words, besides supplying the functionalities, needed for communicating, it can also be entrusted to check that the interaction is evolving in obedience to the rules as well as to reify the social state. The key is to develop artifacts that: (i) implement interaction specifications, and in particular the social expectations (given as commitments) entailed by them, (ii) monitor on-going interactions in order to detect possible incorrect executions and manage them. This perspective allows interaction, that is mediated by programmable environments, to be declined so as to capture different features and structures that are currently studied by as different researches as those concerning organizational theory [24, 7, 19, 20], normative MAS [47], and e-institutions [15, 2].

Figure 2 sketches an e-institution in the setting that we are proposing [23]. It is composed of three artifacts: one for the social state, one acting as a catalog of social actions, and the last one encoding the constraints upon the interaction. We adopt the definition of artifact in [44, 28, 38]: so, each artifact provides a set

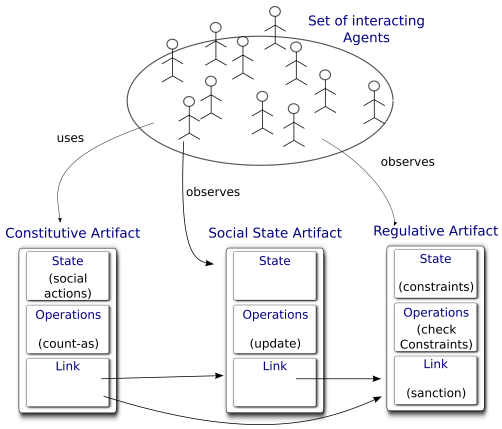

Fig. 2. E-institution representation by means of agents and artifacts.

of *operations*, a set of *observable properties* and a set of *links*. Using an artifact involves two aspects: (1) being able to execute operations, that are listed in its usage interface, and (2) being able to perceive the artifact observable information. From an agent's standpoint, artifact operations represent the actions, that are provided by the environment. Moreover, artifacts can be linked together so as to enable one artifact to trigger the execution of operations over another artifact. The *social state artifact* contains the commitments and the facts as envisioned by commitment-based specification. It is updated after each execution of a social action. Agents can observe its contents. The *constitutive artifact* contains the set of the institutional actions together with their semantics, given in terms of effects on the social state. It must be observable by the agents. The link to the social state represents the fact that the execution of an institutional action triggers the update of the social state. The *regulative artifact* contains the constraints on the evolution of the social state. Violations can be detected automatically when the state is updated by checking whether some constraint is unattended. The check is triggered by the social state itself which, in turn, triggers the regulative artifact. The Mercurio framework accommodates equally well the solution where checks are performed by an arbiter agent. We propose the one based on artifacts because, as observed in [17], agents are autonomous and, therefore,they are free to choose whether to sanction a violation or not. Artifacts guarantee that violations are always detected.

Notice that the modularity entailed by the separation of the constitutive and of the regulative specifications, which characterizes the model, allows also for dinamically changing the agreed rules of the interaction, i.e. the regulative specification, at run-time, along the line of [3]. Indeed, it would be sufficient to include appropriate meta-actions to the constitutive artifact, whose effect is to change some of the constraints defined in the regulative artifact.

Organizations can be realized in a similar way. The focus, here, is on the structure of a certain reality, in terms of roles and groups. Each role will have an artifact associated with it, implementing the actions that the role player can execute on the social state. Organizations can be used to regiment the execution of the protocol [18]. More details in [23].

3 Discussion

We think that Mercurio will give significant contributions to industrial applicative contexts; in particular, to those companies working on software development in large, distributed systems and in service-oriented architectures. Among the most interesting examples are the integration and the cooperation of e-Government applications (services) spread over the nation. In this context, the aim is to verify the adherence of bureaucratic procedures, of the public administration, to the current laws (e.g. http://www.ict4law.org). Another interesting application regards (Web) services. Some of fundamental aspects promoted by the SOA model, such as autonomy and decoupling, are addressed in a natural way by the agent-oriented paradigm. The development and analysis of service-oriented systems can benefit from the increased level of abstraction offered by agents, by reducing the gap between the modeling, design, development, and implementation phases. In this context, it is necessary to deploy complex interactions having those characteristics of flexibility that agents are able to guarantee.

Nowadays, the development of MASs is based on two kinds of tools: agent platforms and BDI (or variations) development environments. The former, e.g. JADE and FIPA-OS provide only a transport layer and some basic services. Moreover, they lack support for semantic interoperability because they do not take into account the semantics of the ACL they adopt. The available BDI development environments, such as Jadex [11] and 2APL [14], support only syntactic interoperability because they do not integrate the semantics of the adopted ACL. Our proposal is compatible with the programming of BDI agents, in that the way agents are realized is totally transparent to the interaction media supplied by Mercurio. As such, Mercurio allows the interaction of *any kind* of agent. At the same time, by directly implementing the interaction specifications into an "intelligent" medium, the social semantics of the provided interactive actions is *guaranteed*; moreover, the medium supplies additional features, in particular concerning the verification of interactions.

Although our proposal is more general than e-institutions, it is interesting to comment some work carried on in this context. The current proposals for electronic institutions do not supply yet all of the solutions that we need: either they do not account for indirect forms of communication or they lack mechanisms for allowing the a priori verification of global properties of the interaction. As [16, 43] witness, there is an emerging need of defining a more abstract notion of action, which is not limited to direct speech acts, whose use is not always natural. For instance, for voting in the human world, people often raise their hands rather than saying the name corresponding to their choice. If the environment were

represented explicitly it would be possible to use a wider range of instrumental actions, that can be perceived by the other agents through the environment that acts as a medium.

Moreover, for what concerns the abstract architecture, e-institutions, e.g. Ameli [15], place a middleware composed of governors and staff agents between participating agents and an agent communication infrastructure. The notion of environment is dialogical: it is not something agents can sense and act upon but a conceptual one that agents, playing within the institution, can interact with by means of norms and laws, based on specific ontologies, social structures, and language conventions. Agents communicate with each other by means of speech acts and, behind the scene, the middleware mediates such communication. Finally, there are two significant differences between artifacts and e-institutions [2]: (i) e-institutions are tailored to a particular, though large, family of applications while artifacts are more generic; (ii) e-institutions are a well established and proven technology that includes a formal foundation, and advanced engineering and tool support, while for artifacts, these features are still in a preliminary phase. Mercurio gives to artifacts the formal foundation in terms of commitments and interaction patterns; engineering tools are currently being developed.

For what concerns organizations, instead, there are some attempts to integrate them with artifacts, e.g. ORA4MAS [19]. Following the A&A perspective, artifacts are concrete bricks used to structure the agents' world: part of this world is represented by the organizational infrastructure, part by artifacts introduced by specific MAS applications, including entities/services belonging to the external environment. In [19] the organizational infrastructure is based on $Moise^{+}$, which allows both for the enforcement and the regimentation of the rules of the organization. This is done by defining a set of conditions to be achieved and the roles that are permitted or obliged to perform them. The limit of this approach is that it cannot capture contexts in which regulations are, more generally, norms because norms cannot be restricted to achievement goals.

Finally, for what concerns commitment-based approaches, the main characteristic of the Mercurio proposal stands in the realization of the commitment stores as artifacts/environments which are capable to monitor the interaction, taking into account also the regulative specification. Such kind of artifact is a first-class object of the model, and is available to the designer. These features are advantageous w.r.t. approaches like [13], where these elements reside in the middleware, and are therefore shielded from the agents and from the designer. A negative consequence of the lack of appropriate programming abstraction is the difficulty to verify whether the system corresponds to the specification.

4 Conclusion

The Mercurio framework, presented in this paper, represents the core of a research project proposal, that is currently waiting for approval. The formal model of Mercurio relies on the commitment-based approach, presented in [6, 5]. The proposals coming from Mercurio conjugate the flexibility and openness features

that are typical of MAS with the needs of modularity and compositionality that are typical of design and development methodologies. The adoption of commitment protocols makes it easier and more natural to represent (inter)actions that are not limited to communicative acts but that include interactions mediated by the environment, namely actions upon the environment and the detection of variations of the environment.

For what concerns the infrastructure, a first result is the definition of environments based on A&A and on CArtAgO, that implement the formal models and the interaction protocols mentioned above. A large number of environments, described in the literature and supporting communication and coordination, have been stated considering purely reactive architectures. In Mercurio we formulate environment models that allow goal/task-oriented agents (those that integrate pro-activities and re-activities) the participation to MAS. Among the specific results related to this, we foresee an advancement of the state of the art with respect to the definition and the exploitation of forms of stigmergic coordination [36] in the context of intelligent agent systems. A further contribution regards the flexible use of artifact-based environments by intelligent agents, and consequently the reasoning techniques that such agents may adopt to take advantage of these environments. First steps in this direction, with respect to agents with BDI architectures, have been described in [34, 32].

The Mercurio project aims at putting forward a proposal for a language that uses the FIPA ACL standard as a reference but integrates forms of interactions, that are enabled and mediated by the environment, with direct forms of communication. This will lead to an explicit representation of environments as first-class entities (in particular endogenous environments based on artifacts) and of the related model of actions/perceptions. As future work, we plan to implement a prototype of the reference infrastructural model. The prototype will be developed upon and integrate existing open-source technologies, among which: JADE, the reference FIPA platform, CArtAgO, the reference platform and technology for the programming and execution of environments, as well as agent-oriented programming languages such as Jason and 2APL.

References

1. M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. Verifiable Agent Interaction in Abductive Logic Programming: The SCIFF Framework. *ACM Trans. Comput. Log.*, 9(4), 2008.
2. J. L. Arcos, P. Noriega, J. A. Rodríguez-Aguilar, and C. Sierra. E4MAS Through Electronic Institutions. In Weyns et al. [45], pages 184–202.
3. A. Artikis. A Formal Specification of Dynamic Protocols for Open Agent Systems. *CoRR*, abs/1005.4815, 2010.
4. M. Baldoni, C. Baroglio, A. K. Chopra, N. Desai, V. Patti, and M. P. Singh. Choice, Interoperability, and Conformance in Interaction Protocols and Service Choreographies. In *Proc. of AAMAS*, pages 843–850. 2009.
5. M. Baldoni, C. Baroglio, and E. Marengo. Behavior-oriented Commitment-based Protocols. In *Proc. of ECAI*, pages 137–142, 2010.

6. M. Baldoni, C. Baroglio, E. Marengo, and V. Patti. Constitutive and Regulative Specifications of Commitment Protocols: a Decoupled Approach. *ACM TIST, Special Issue on Agent Communication*, 2011.
7. M. Baldoni, G. Boella, and L. van der Torre. Bridging Agent Theory and Object Orientation: Agent-like Communication among Objects. In *Post-Proc. of ProMAS 2006*, LNAI 4411, pages 149–164. Springer, 2007.
8. F. Bellifemine, F. Bergenti, G. Caire, and A. Poggi. JADE - A Java Agent Development Framework. In *Multi-Agent Progr.: Lang., Plat. and Appl.*, vol. 15 of *MAS, Art. Soc., and Sim. Org.*, pages 125–147. Springer, 2005.
9. F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa. JADE: A Software Framework for Developing Multi-agent Applications. Lessons learned. *Information & Software Technology*, 50(1-2):10–21, 2008.
10. R. H. Bordini, J. F. Hübner, and R. Vieira. Jason and the Golden Fleece of Agent-Oriented Programming. In In *Multi-Agent Progr.: Lang., Plat. and Appl.*, vol. 15 of *MAS, Art. Soc., and Sim. Org.*, pages 3–37. Springer, 2005.
11. L. Braubach, A. Pokahr, and W. Lamersdorf. Jadex: A BDI Agent System Combining Middleware and Reasoning. In *Software Agent-Based Applications, Platforms and Development Kits*. Birkhauser Book, 2005.
12. C. Cherry. Regulative Rules and Constitutive Rules. *The Philosophical Quarterly*, 23(93):301–315, 1973.
13. A. K. Chopra and M. P. Singh. An Architecture for Multiagent Systems: An Approach Based on Commitments. In *Proc. of the ProMAS*, 2009.
14. M. Dastani. 2APL: a Practical Agent Programming Language. *Autonomous Agents and Multi-Agent Systems*, 16(3):214–248, 2008.
15. M. Esteva, B. Rosell, J. A. Rodríguez-Aguilar, and J. L. Arcos. AMELI: An Agent-Based Middleware for Electronic Institutions. In *AAMAS*, pages 236–243. 2004.
16. N. Fornara, F. Viganò, and M. Colombetti. Agent Communication and Artificial Institutions. *JAAMAS*, 14(2):121–142, 2007.
17. N. Fornara, F. Viganò, M. Verdicchio, and M. Colombetti. Artificial Institutions: a Model of Institutional Reality for Open Multiagent Systems. *Artif. Intell. Law*, 16(1):89–105, 2008.
18. J. F. Hübner, O. Boissier, and R. H. Bordini. From Organisation Specification to Normative Programming in Multi-agent Organisations. In *CLIMA XI*, LNAI 6245, pages 117–134. Springer, 2010.
19. J. F. Hubner, O. Boissier, R. Kitio, and A. Ricci. Instrumenting Multi-agent Organisations with Organisational Artifacts and Agents: "Giving the Organisational Power Back to the Agents". *Proc. of AAMAS*, 20, 2009.
20. J. F. Hübner, J. S. Sichman, and O. Boissier. Developing Organised Multiagent Systems Using the MOISE. *IJAOSE*, 1(3/4):370–395, 2007.
21. D. Keil and D. Goldin. Modeling Indirect Interaction in Open Computational Systems. In *Proc. of TAPOCS*, pages 355–360. IEEE Press, 2003.
22. E. Marengo, M. Baldoni, C. Baroglio, A. K. Chopra, V. Patti, and M. P. Singh. Commitments with Regulations: Reasoning about Safety and Control in REGULA. In *Proc. of AAMAS*, 2011.
23. E. Marengo. Designing and Programming Commitment-based Service-oriented Architectures on top of Agent and Environment Technologies. Technical Report RT 129/2010, Dip. di Informatica, Univ. di Torino, 2010.
24. C. Masolo, L. Vieu, E. Bottazzi, C. Catenacci, R. Ferrario, A. Gangemi, and N. Guarino. Social Roles and their Descriptions. In *Proc. of KR2004*, pages 267–277, 2004. AAAI Press.

25. P. McBurney and S. Parsons. Games That Agents Play: A Formal Framework for Dialogues between Autonomous Agents. *JLLI*, 11(3):315–334, 2002.
26. M. Montali, M. Pesic, W. M. P. van der Aalst, F. Chesani, P. Mello, and S. Storari. Declarative Specification and Verification of Service Choreographies. *ACM TWEB*, 4(1), 2010.
27. E. Oliva, P. McBurney, and A. Omicini. Co-argumentation Artifact for Agent Societies. In *Proc. of ArgMAS 2007*, LNCS 4946, pages 31–46, 2007. Springer.
28. A. Omicini, A. Ricci, and M. Viroli. Artifacts in the A&A Meta-model for Multi-agent Systems. *JAAMAS*, 17(3):432–456, 2008.
29. A. Omicini, A. Ricci, M. Viroli, C. Castelfranchi, and L. Tummolini. Coordination Artifacts: Environment-Based Coordination for Intelligent Agents. In *Proc. of AAMAS*, pages 286–293, 2004.
30. Singh M. P. An Ontology for Commitments in Multiagent Systems. *Artif. Intell. Law*, 7(1):97–113, 1999.
31. Maja Pesic and Wil M. P. van der Aalst. A Declarative Approach for Flexible Business Processes Management. In *Proc. of BPM*, pages 169–180, 2006.
32. M. Piunti and A. Ricci. Cognitive Use of Artifacts: Exploiting Relevant Information Residing in MAS Environments. In *Proc. of KRAMAS*, LNCS 5605, pages 114–129, 2008. Springer.
33. M. Piunti, A. Ricci, O. Boissier, and J. F Hübner. Embodying Organisations in Multi-agent Work Environments. In *Proc. of IAT*, pages 511–518. 2009.
34. M. Piunti, A. Ricci, L. Braubach, and A. Pokahr. Goal-Directed Interactions in Artifact-Based MAS: Jadex Agents Playing in CARTAGO Environments. In *Proc. of IAT*, pages 207–213. IEEE, 2008.
35. S. K. Rajamani and J. Rehof. Conformance Checking for Models of Asynchronous Message Passing Software. In *Proc. of CAV*, pages 166–179. 2002.
36. A. Ricci, A. Omicini, M. Viroli, L. Gardelli, and E. Oliva. Cognitive Stigmergy: Towards a Framework based on Agents and Artifacts. In [45], pages 124–140.
37. A. Ricci, M. Piunti, D. L. Acay, R. H. Bordini, J. F. Hübner, and M. Dastani. Integrating Heterogeneous Agent Programming Platforms within Artifact-based Environments. In *Proc. of AAMAS (1)*, pages 225–232. 2008.
38. A. Ricci, M. Piunti, and M. Viroli. Environment Programming in MAS – An Artifact-Based Perspective. *JAAMAS*.
39. A. Ricci, M. Piunti, M. Viroli, and A. Omicini. Environment Programming in CArtAgO. In *Multi-Agent Prog. II: Lang., Plat. and Appl.*, 2009.
40. J.R. Searle. *The Construction of Social Reality*. Free Press, New York, 1995.
41. M. P. Singh. A Social Semantics for Agent Communication Languages. In *Issues in Agent Communication*, LNCS 1916, pages 31–45. Springer, 2000.
42. P. R. Telang and M. P. Singh. Abstracting Business Modeling Patterns from RosettaNet. In *SOC: Agents, Semantics, and Engineering*, 2010.
43. M. Verdicchio and M. Colombetti. Communication Languages for Multiagent Systems. *Comp. Intel.*, 25(2):136–159, 2009.
44. D. Weyns, A. Omicini, and J. Odell. Environment as a First Class Abstraction in Multiagent Systems. *JAAMAS*, 14(1):5–30, 2007.
45. D. Weyns, H. Van Dyke Parunak, and F. Michel, editors. *Environments for Multi-Agent Systems III*, LNCS 4389. Springer, 2007.
46. P. Yolum and M. P. Singh. Commitment Machines. In *Proc. of ATAL*, pages 235–247, 2001.
47. F. Zambonelli, N. R. Jennings, and M. Wooldridge. Developing Multiagent Systems: The Gaia Methodology. *ACM Trans. Softw. Eng. Methodol.*, 12(3):317–370, 2003.