Artif Intell Rev
DOI 10.1007/s10462-012-9390-7

Semantic technologies for open interaction systems

Nicoletta Fornara · Charalampos Tampitsikas

Abstract Open interaction systems play a crucial role in agreement technologies because they are software devised for enabling autonomous agents (software or human) to interact, negotiate, collaborate, and coordinate their activities in order to establish agreements and manage their execution. Following the approach proposed by the recent literature on agent environments those open distributed systems can be efficiently and effectively modeled as a set of correlated physical and institutional spaces of interaction where objects and agents are situated. In our view in distributed open systems, spaces are fundamental for modeling the fact that events, actions, and social concepts (like norms and institutional objects) should be perceivable only by the agents situated in the spaces where they happen or where they are situated. Institutional spaces are also crucial for their active functional role of keeping track of the state of the interaction, and for monitoring and enforcing norms. Given that in an open distributed and dynamic system it is fundamental to be able to create and destroy spaces of interaction at run-time, in this paper we propose to create them using Artificial Institutions (AIs) specified at design time. This dynamic creation is a complex task that deserves to be studied in all details. For doing that, in this paper, we will first define the various components of AIs and spaces using Semantic Web Technologies, then we will describe the mechanisms for using AIs specification for realizing spaces of interaction. We will exemplify this process by formalizing the components of the auction Artificial Institution and of the spaces created for running concrete auctions.

Keywords Open interaction systems · Artificial institutions · Institutional spaces · Semantic web technologies · Norms · OWL

N. Fornara (✉) · C. Tampitsikas
Università della Svizzera italiana, via G. Buffi 13, 6900 Lugano, Switzerland
e-mail: nicoletta.fornara@usi.ch

C. Tampitsikas
University of Applied Sciences Western Switzerland, 3960 Sierre, Switzerland
e-mail: charalampos.tampitsikas@usi.ch

Published online: 24 January 2013

1 Introduction

Open interaction systems play a crucial role in agreement technologies because they are software devised for enabling autonomous agents (software or human) to interact, negotiate, collaborate, and coordinate their activities in order to establish agreements for achieving certain goals. These interactions may be finalized to the definition of contracts or agreements among multiple parties and to their execution, monitoring, and enforcement. The most important aspect of this type of systems is the lack of knowledge, in advance, of what agents may participate in one of their enactment, therefore no assumption can be made on the internal architecture of the participating agents or on their willing to satisfy the norms and the rules that regulate the interaction (d'Inverno et al. 2012). Open interaction systems are crucial for the design, development, and deployment of applications in different fields, like e-commerce, e-government, supply-chain, management of virtual enterprise, and collaborative-resource sharing systems.

Open interaction systems are *dynamic distributed event based systems* having the following fundamental components:

– A *state* that evolves due to the *events* that happens and the *actions* (viewed as events with an actor) performed by the interacting agents. Events and actions are described by means of their *preconditions*, which need to be satisfied for the successful performance of the events or actions, and their *effects* on the state of the system. Important events are due to the *elapsing of time* or to the change of the value of some properties. Crucial actions are *communicative acts* performed by the agents to interact and negotiate.
– Given that no assumption can be made on the expected behaviour of the interacting agents, *norms* are a fundamental part of open systems. They are used to express obligations, prohibitions, permissions that regulate the interaction of the agents. Norms can also be used to express institutional powers, which are specific institutional conditions for the successful performance of institutional actions. At design time norms are expressed in an abstract form in terms of *roles* and they may contain some un-specified parameters that become defined only at run-time during specific interactions among specific agents.
– Given that the interacting agents and the open interaction system itself are software that are running on *different platforms*, it is required to define standard mechanisms and rules for the agents for: (i) *perceiving* the state of the system, and the events and actions that happen in the system; and (ii) for *acting* within the system. Moreover due to performance, security, privacy, and relevance reasons in a distributed system *limited observability* of events and actions and the contextual relevance of norms have to be taken into account (Okuyama et al. 2008).

Following the approach proposed by the recent literature on agent environments (Weyns et al. 2007; Ricci et al. 2011) those open distributed systems can be efficiently and effectively modeled as a set of correlated physical and *institutional spaces* of interaction where *objects* and *agents* are situated (Tampitsikas et al. 2012). In our view in distributed open systems, spaces are fundamental for modeling the fact that events, actions, and social concepts (like norms and institutional objects) should be perceivable only by the agents situated in the spaces where they happen. Institutional spaces are also crucial for their active functional role of keeping track of the state of the interaction, and for monitoring and enforcing norms.

This abstract concept of *space* presents some similarities with the notion of *container* introduced in the GOLEM environment framework (Bromuri and Stathis 2009), the notions of *workspace* introduced in the CArtAgO environment framework (Ricci et al. 2009), and the concept of *scene* used in the formalization of Electronic Institutions (Esteva et al. 2001;

d'Inverno et al. 2012). But spaces in our approach are not just the containers of agents and objects (or artifacts as proposed in the CArtAgO framework) modeling the locality of the application domain but in addition they are the main enforcers of the norms and regulators of the open interaction systems' evolution.

Given that in an open distributed and dynamic system it is fundamental to be able to create and destroy spaces of interaction at run-time, in this paper we propose to create them using Artificial Institutions (AIs) specified at design time. In particular starting from our past studies on AI (Fornara et al. 2007; Fornara and Colombetti 2009b) and on the formalization of obligations using Semantic Web Technologies (Fornara and Colombetti 2010; Fornara 2011; Fornara et al. 2012), in this paper we will first define the various components of AIs using Semantic Web Technologies (Hitzler et al. 2009). Then we will describe the mechanisms for using AIs specification for concretely realizing at run-time spaces of interaction formalized using Semantic Web Technologies. Those institutional spaces will represent the multi-agent environment where the agents will be situated. In order to be able to manage the effects and the perception of social and institutional interactions among agents, and in particular the performance of institutional actions (Fornara et al. 2007), the functionalities and services encapsulated in existing models of agent environments should be extended coherently.

We will exemplify the proposed approach by formalizing the components of the auction AI and by describing how to use that AI for dynamically creating the spaces required for running concrete auctions. In this paper we will also propose to use this approach for concretely realizing a prototype for running auctions where the environment component is obtained by extending the functionalities of an existing environment framework: the GOLEM (Bromuri and Stathis 2009) environment. In particular GOLEM will be extended and interfaced with the Semantic Web representation of AIs and spaces by implementing a suitable synchronization component.

This paper is organized as follows. In Sect. 2 we introduce the notion of Artificial Institution (AI), of space of interaction, and their connections. In Sect. 3 the OWL model of AI, of spaces of interaction, and the mechanisms for using AIs for dynamically creating at run-time spaces of interaction are presented. In this section we will also discuss the need to define hierarchies of AIs (Colombetti et al. 2002) and hierarchies of the spaces realized using such AIs. In Sect. 4 the architecture of the prototype that we are implementing for testing the proposed model is briefly described and compared to related approaches.

2 Spaces of interaction and artificial institutions

In our view, open interaction systems for autonomous heterogenous agents can be modeled using AIs, and enacted as a set of physical and institutional *spaces of interaction* (Tampitsikas et al. 2012). Spaces are introduced to model the fact that interactions usually take place in a limited physical and/or institutional place, for example in a classroom, in a meeting room, during a run of an auction, or inside a team created for solving a specific problem.

The notion of space is fundamental in the specification of open systems to model *limited observability*. A space, in fact, represents for the interacting agents the *boundaries* for the effects and for the perception of the events and actions that happen in a space, which indeed may be perceived only by the agents inside that space. A space has also functional responsibilities, that is, it is in charge of *mediating* the events and the actions that happen inside the space. This means that the space has to register the fact that an event or action has happened, and it has to notify it to the agents in the space that are registered for its template. In this paper we will extend this notion of space in order to be able to formalize the components required for representing and managing the *institutional aspects of the interaction*.

An environment for multi-agent systems (MAS) is composed by multiple *spaces* where *objects* and *agents* are situated. In an agent environment *objects* are one of the building blocks. They are used to represent the various non-autonomous components of the system, like *physical entities* external with respect to the system (like databases or external files and web services), offering an abstraction, for the agents, that hides the low level details. Objects are fundamental also for representing *institutional entities* that are manipulated by the agents during their institutional interaction. Institutional entities have one or more *institutional attribute*, and their value can be changed thanks to the performance of *institutional actions* (Fornara et al. 2008) and thanks to the common acceptance of the semantics of those actions from the agents belonging to the space where the action is performed and the object, on which it is performed, is situated. For example a run of an auction can be opened or closed by suitable institutional actions that have the effect to change the state of the run of the auction, similarly the attributes of an agreement or of a contract holding between different agents can be set with specific values. Physical objects can be considered institutional objects when they get institutional attributes during the dynamic evolution of the state of the environment.

In order to manage the performance of *institutional actions* and the fact that the interactions are regulated by *norms* (that are used to express obligations, prohibitions, and permissions to perform certain type of actions) it is necessary to extend the functionalities of spaces to concretely realize the mechanisms for:

- Keeping track of the interactions among agents and computing the state of spaces on the basis of the events, actions, and institutional actions performed by the agents and on the basis of their pre-conditions and effects. For example in the Dutch auction the agent playing the role of auctioneer can only lower the current ask price.
- Checking that the agent that performs an institutional action has the institutional power for doing such an action and that all the other preconditions for the performance of the action are satisfied. For example an agent that has not the institutional power of declaring open the auction can attempt to do its effects will not change the state of the state of the interaction where the action is performed, i.e. the action is void;
- *Monitoring* the interactions, that is represent in the most suitable way the actions and events that happen in the system in the state of the interaction, then check if they are compliant with a given set of norms used to express obligations, prohibitions, and permissions related to type of actions;
- *Enforcing* the norms for example by applying sanctions (Fornara and Colombetti 2009a) that change the reputation values of the agents.

We assume that in the environment used for creating an open interaction system there is always a *root space* that contains all the physical laws of the system, this is also the space where the agents need to register for starting to interact in a given open system. The agents situated in such a root space need to realize complex interactions with the other agents, and in particular they need to be able to dynamically create and destroy at run-time spaces of interaction. Think for example to a market-place where the spaces for running specific type of auctions or for negotiating different types of contracts are continuously created and terminated. Given that defining all the rules, norms, and the context of interactions at run-time is a complex task, in this paper we propose to create such spaces of interaction using pre-defined pattern of interaction, defined at design time, which are modeled using the notion of *Artificial Institution* (AI) (Fornara et al. 2007, 2008; Fornara and Colombetti 2009b). This approach, from the software engineering point of view, has also the significant advantage of making it possible to use many times existing specifications of AIs. For example the AI specification of a given type of auction, the Dutch Auction can be used for realizing different run of this type of auction in an open electronic market-place. As we will propose in next sections, the process of re-using existing AI is encouraged and made easier if standard well-known technologies, like the Semantic Web Technologies, are used for their formalization.

The creation at run-time of an institutional space by using AIs defined at design time is a complex task that deserves to be studied in all details. For doing that, in this paper, we will first define the various application independent components of AIs using Semantic Web Technologies, in particular OWL 2 DL[1] (Hitzler et al. 2009). The model of AI that we propose in this paper is inspired from the model presented in Fornara and Colombetti (2009b) where AI are formalized using Event Calculus. Obviously given that in this proposal we adopt other formal languages we will need to change some parts of the model and to extend it to take into account its connections with the environment components, that is with spaces and objects. For example an advantage of this new model is that the content of norms are not any more specific actions but a classes of possible actions. Then in a second step we will study and describe the mechanisms for using the specification of a generic AI for realizing and executing spaces of interaction. Finally we will explain how we plan to use this model based on OWL AIs and spaces for the realization of a first prototype of a market-place. From the architectural point of view an open interaction system is a particular type of *distributed event-based system*, which is in charge of the complex task of distributing the perception and notification of actions and events to the participating agents. Therefore for the architecture of our prototype we decided to adopt and extend an already existing environment framework: the GOLEM framework (Bromuri and Stathis 2009).

There are many advantages in using Semantic Web Languages (Antoniou and Harmelen 2008; Hitzler et al. 2009) for the specification and realization of an open interaction systems with respect to the adoption of other formal languages, like for example the Z state-based specification language based on set theory used in d'Inverno et al. (2012) for the specification of Electronic Institutions. The first advantage is that Semantic Web languages are international standards, and therefore it is possible to realize systems by reusing existing ontologies (for example the Time Ontology) and the ontologies proposed in this work may be easily re-used in other systems. Second it is possible to use some of the good existing tools and libraries for editing ontologies and accessing them from programs, that is by directly using them for the implementation of open systems. Moreover given that OWL 2 DL is a decidable fragment of FOL there are several reasoners available[2] for reasoning on OWL specifications and deducing knowledge. Finally given that OWL 2 DL ontologies coming from different sources can be easily merged by taking the union of their axioms (or using ontology alignment mechanisms when the different ontologies are not immediately compatible) it is possible for agents to enter in different open systems and reason on their future actions, even if those interaction systems have different set of norms, rules, and description of the context of the enabled interactions.

3 Formal specification of AIs and spaces using OWL

In this section we formalize, using Semantic Web Technologies, the classes, properties, individuals, axioms, and assertions required for the specification of Artificial Institutions at design time. We also propose a formal specification of the concepts required for the definition and

[1] http://www.w3.org/TR/owl2-syntax/.

[2] W3C list of reasoners, editors, development environments, APIs: http://www.w3.org/2007/OWL/wiki/Implementations.

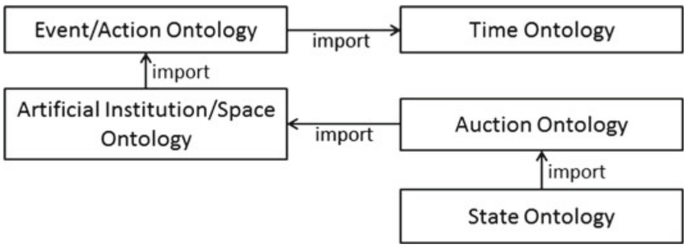

Fig. 1 Import relationship among the proposed OWL 2 DL ontologies

dynamic evolution of institutional spaces at run-time and the process for dynamically creating them using AI specifications. The concepts introduced will be exemplified by formalizing some components of the *Dutch auction AI* and of the spaces that it is possible to generate from this AI. The constructs used for defining AIs and spaces will be mainly defined in a set of OWL 2 DL ontologies (see "Appendix A" for a brief introduction to the main components of an OWL ontology) that represent different components of the proposed model.

Those ontologies, made available on-line,[3] have been created using Protégé ontology editor[4] and checked (in order to determine whether or not the ontology is consistent, identify subsumption relationships between classes, and many other things) by using the HermiT[5] OWL 2 DL reasoner. In presenting their formalization we use capital initials for classes, and lower case initials for properties and individuals. We assume that all individuals introduced in the ABox of the ontologies are asserted being different individuals.

The process of actually creating and destroying spaces of interaction at run-time, is concretely realized by a program that accesses, using OWL-API,[6] the content of the ontology where the model of the relevant AI is stored and that manipulates the content of the *State Ontology*, used for representing the state of the existing spaces of interaction, by adding a new individual that represents the new space. We decided to use a program and OWL libraries for creating new individuals in the ontology because it is impossible to write an OWL 2 DL axiom for doing that.

In order to be able to use already existing ontologies, like for example the W3C OWL *Time Ontology*,[7] and for making the ontologies proposed in this paper usable in other applications, we decide to create the following different ontologies for the formalization of our model: an application independent ontology for describing events and actions, the *Event/Action Ontology*; an application independent ontology where the concepts required for describing artificial institutions and spaces are formalized, the *Artificial Institution/Space Ontology*; an ontology for describing different type of auctions using the proposed model of AI, the *Auction Ontology*, and an ontology used for describing at run-time the state of the spaces of interaction dynamically created, the *State Ontology*. Those ontologies are connected by an "import" relationship as depicted in Fig. 1.

The *Time Ontology* is used to represent *instants of time*, *intervals*, and relationships among them. The *Event/Action Ontology*, introduced in one our previous work (Fornara 2011), is used to represent events and actions (with the classes Event and Action) and it imports

[3] http://www.people.lu.unisi.ch/ontolgy/ontologies.zip.

[4] http://protege.stanford.edu/.

[5] http://hermit-reasoner.com/.

[6] http://owlapi.sourceforge.net/.

[7] http://www.w3.org/TR/owl-time/.

the *Time Ontoloy*. An event is connected to the instant of time when it happens by the property $\mathsf{atTime}: \mathsf{Eventuality} \rightarrow \mathsf{TemporalEntity}$ where $\mathsf{Event} \sqsubseteq \mathsf{Eventuality}$ and the class Eventuality is used to generalize the notion of event and fluent (a state of affair that can change in time). Every event has an atTime value as expressed by the following axiom: $\mathsf{Event} \equiv \exists \mathsf{atTime.Instant}$. Actions are particular type of events having an actor: $\mathsf{Action} \sqsubseteq \mathsf{Event}$; $\mathsf{Action} \equiv \exists \mathsf{hasActor.Agent}$. The class $\mathsf{ChangeEvent} \sqsubseteq \mathsf{Event}$ represents the events due to the change of the value of a property. The class $\mathsf{TimeEvent} \sqsubseteq \mathsf{Event}$ represents a special type of events whose characteristic is simply of being associated to an instant of time. They are useful for expressing deadline in obligations specification. In order to be able to represent that an instant of time is elapsed we introduce the class $\mathsf{Elapsed} \sqsubseteq \mathsf{Instant}$. If events/actions are associated to an elapsed instant of time, they actually happened, otherwise they are simply a description of those events/actions. The assignment of an instant of time to the class Elapsed is realized by a the software in charge of representing or simulating the time evolution of the system, this with the goal of being able to monitor the behavior of the agents.

3.1 Institutional actions

Institutional actions (Fornara et al. 2007) (represented with the class $\mathsf{InstAction} \sqsubseteq \mathsf{Action}$) are a special type of actions whose effects change the value of institutional properties. For example the action of opening or closing an auction, creating a space of interaction, or assigning a role to an agent. An *institutional action* is successfully performed if and only if the actor of the action has the *institutional power* to perform such an action and if other application dependent contextual conditions are satisfied (Searle 1995), otherwise the effects of the action are empty. In our previous institutional models (Fornara et al. 2007; Fornara and Colombetti 2009b) institutional actions were performed by means of declarative communicative acts. This approach has the problem that the receiver of those acts should be the set of all agents for which the institutional action is relevant. Computing this set of agents is a complex task for one agent situated in a distributed open system with limited observability of the state of the interaction. On the contrary in the model presented in Tampitsikas et al. (2012) and in this paper, agents may *attempt* to perform institutional actions in a specific space of the environment. Then the environment is in charge of checking if the preconditions of the institutional action (at least the ones related to institutional power) are satisfied. If the preconditions are satisfied the action happens and its effects are represented in the state of the space where the action is performed and become perceivable, at least, by those agents situated in that space and for which the action is relevant (the agents registered to a template of the institutional action).

3.2 Artificial institutions and institutional spaces

Artificial institutions are defined at design-time and at run-time they can be used for creating one or more institutional spaces. An *Artificial Institution* (AI) is characterized by: (i) a set of concepts and properties introduced by the specific AI; for example in an auction AI it is necessary to represent the products, the reservation price, the various ask-prices, the value of the bids, the maximum duration of the auction, and so on; (ii) a set of actions available for the agents and defined by the AI; (iii) a set of roles, which are labels defined in a given AI for abstracting from the specific agents that will take part at run-time to an interaction; (iv) a set of norms for expressing at design time the obligations, permissions, prohibitions, and institutional powers of the agents that will play a given role.

A specific *institutional space* is characterized by: (i) the AI or AIs used to create the space; if more than one AI is used for creating a space, the problem of aligning different AIs models and checking the consistency of the specification obtained by using different AIs arises; (ii) the set of sub-institutional spaces dynamically created inside a given space; (iii) the list of agents that are inside the space at a given instant of time; (iv) the list of roles defined in the space that is generated from the list of roles of the AIs used to create the space; (v) the list of objects (i.e. the products sold in the auction, concrete obligations and institutional powers) that the agents can manipulate in the space.

For representing those concepts we create the *Artificial Institution/Space Ontology* that defines the ArtificialInst and InstSpace classes and the following property used to connect a space with the AI used for its realization:

isRealizationOf: InstSpace $\rightarrow$ ArtificialInst.

For exemplifying those concepts we create the *Auction Ontology* used for defining an artificial institution that can be used for realizing generic auctions. In this ontology we create the individual auction that belongs to the ArtificialInst class. In those ontologies agents are represented as individuals that belong to the Agent class. The isIn: Agent $\rightarrow$ InstSpace property is used to connect an agent with the spaces where it is situated.

We assume that at run-time every interaction system has a root-space that belongs to the PhysicalSpace class. PhysicalSpace and InstSpace are both subclass of the Space class. Every new space that should be created is a sub-space of an existing space. For representing this relation among spaces we define the sub-space: Space $\rightarrow$ Space property.

Institutional actions for creating new spaces can be represented in the ontology as individuals belonging to the CreateSpace $\sqsubseteq$ InstAction class. This action has various parameters, some of them are independent from the type of the AI used for creating the space, they are: (i) the actor (a general property of all type of actions) represented with the hasActor: Action $\rightarrow$ Agent property; (ii) the name of the space where the action is performed, which should be specified for every type of action and it is represented with the performedIn: Action$\rightarrow$ Space; Functional(performedIn) property; (iii) the name of the new space that should be created; and (iv) the name of the AI that should be used for creating the space. Some other parameters of the create space action are strictly related to the AI used for creating the space, for example if the auction AI is used, required values are the date when the auction will start and the reservation price. The generic create space action performed by agent Robert in the root-space at instant2 by using the auction AI can be represented in the *State Ontology* with the following assertions:

CreateSpace(act01); hasActor(act01,Robert); performedIn(act01,root-space);
newSpace(act01,run01); usedAI(act01,auction); Instant(instant2);
atTime(act01,instant2);

The effects of this action are represented by the following assertions:

InstSpace(run01); sub-space(run01, root-space);
isRealizationOf(run01,auction);

If the action is successful all those assertions are added to the *ABox* of the *State Ontology* by the synchronization component described in Sect. 4. An agent situated in a space can enter in all its sub-spaces and can be contemporarily in two or more institutional spaces. The rules that regulate the action of entering in the various sub-spaces are defined in the external space.

3.3 Hierarchy of artificial institutions and spaces

The auction artificial institution defines the concepts and properties that are common to every type of auction, like for example the ask-price, the reservation price, the action of bidding, and the roles of auctioneer, participant, and winner. Specific type of auctions, like the *English auction* or the *Dutch auction* defines further properties, roles, and norms, specific for that type of auction. For example they have different rules for determining the winner of one run of the auction. The winner of the English auction is the participant who did the highest bid once the run of the auction is closed. The winner of the Dutch auction is the first participant who accepts the current ask price declared by the auctioneer. In the *Auction Ontology* those different types of auctions can be modeled with the following individuals belonging to the ArtificialInst class: ArtificialInst(eng-auction), ArtificialInst(dutch-auction). Those different types of auctions creates a *hierarchy* of AIs that is crucial for the re-usability of AI models. Given that these AIs are represented in the ontology as individuals (not as classes) this hierarchy should be explicitly represented by introducing the following transitive property:

specializes: ArtificialInst $\rightarrow$ ArtificialInst; Tra(specializes);
specializes(eng-auction, auction); specializes(dutch-auction, auction).

This hierarchy of AIs influences the mechanisms by which the properties, the roles and the norms of an AI are inherited by more specific type of AIs, as we will discuss in the following sub-sections. The specialize relation between AIs, like the one previously introduced, creates a hierarchy between the classes of the spaces created using those AIs as asserted by the following axiom:

isRealizationOf $\circ$ specializes $\sqsubseteq$ isRealizationOf.

As consequence of this axiom the class of the spaces that realizes a specific AI, like for instance the eng-auction AI, is sub-class of the class of spaces that realizes the more generic AI, like the auction AI, as expressed in the following axiom. This is an important aspect, because the properties having as domain the more generic class of spaces can be used also for the more specific class of spaces. For example the property that associates to an auction space the price that the winner has to pay for the auctioned product is a generic property defined for every type of auction. A consequence of this axiom is also that a space that realizes a given AI realizes also all its more generic AIs. For example if the space run01-dutch-auction realizes the dutch-auction AI it realizes also the auction AI.

3.4 Roles

An AI may define different roles, for example in the auction AI we have the roles of auctioneer and participant, in a company we have the roles of boss and employee. In our model a *role* is a label defined in a given AI. We introduce the class RoleName to represent the set of possible role labels and the following property that associates a role label to the AI where it is defined:

isRoleOf: RoleName $\rightarrow$ ArtificialInst.

For example auctioneer is a role defined by the auction AI as stated by the following assertions: RoleName(auctioneer); isRoleOf(auctioneer, auction);

At runtime agents situated in a given space may play the roles defined in the AIs used to create such a space. For example agent Robert may play the role auctioneer in the

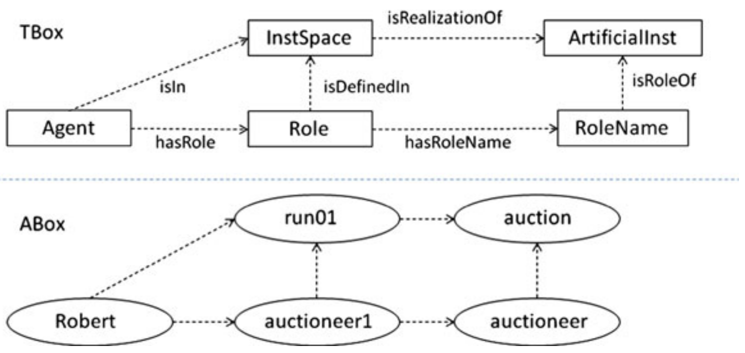

Fig. 2 Classes, properties, and individuals related to the notion of role

institutional space run01 that realizes the auction AI. This is a ternary predicates that cannot be expressed as a single property in OWL[8]: the fact that Robert belongs to the institutional space run01 is not enough to know that he is playing the role of auctioneer in this space. This because Robert can belong also to another institutional space, for example run02 that is a realization of the same AI used to create run01, but where Robert is not playing the role of auctioneer. This is a very common problem in OWL ontologies, to solve it, similarly to what is proposed in the W3C Organization Ontology[9] (where the Membership class is defined, but where there is not the idea of defining reusable AI models), we introduce the Role class used to collect the roles defined in a specific institutional space. These roles are connected with their corresponding role names in the AIs and with the space where are defined by the following properties:

hasRoleName: Role $\rightarrow$ RoleName; Functional(hasRoleName);
isDefinedIn: Role $\rightarrow$ InstSpace; Functional(isDefinedIn);

When a new space is created it is necessary to add to the Role class the individuals used for representing its roles. For example when the space run01 is created using the auction AI, the role auctioneer01 has to be created in the space and it has to be connected to the role auctioneer defined in the auction AI by means of the following assertions:

Role(auctioneer01); isDefined(auctioneer01,run01);
hasRoleName(auctioneer01, auctioneer).

The property: hasRole: Agent $\rightarrow$ Role allows to represent the fact an agent plays a given role. For example when agent Robert is in the space run01 and starts to play the role auctioneer01 we have to add to the ontology the assertion hasRole(Robert,auctioneer01).All those classes, properties, and individuals related to the notion of role are represented for more clarity in Fig. 2.

The institutional actions for assigning a role to an agent are represented with the class AssignRole $\sqsubseteq$ InstAction. These actions have as parameter the *actor* (like all type of actions), the *agent* that will play the new role, and the *role*. For this type of actions, the *space* where the action is performed is univocally determined by the name of the role. When an action

[8] See http://www.w3.org/TR/swbp-n-aryRelations/ for a complete explanation of how defining n-ary relations on the Semantic Web.

[9] http://www.w3.org/TR/vocab-org/.

of this type is successfully performed it is registered in the OWL ontology, and its specific parameters are represented by means of the following properties:

hasAssignedAgent: AssignRole $\rightarrow$ Agent
hasAssignedRole: AssignRole $\rightarrow$ Role.

Similarly the DismissRole $\sqsubseteq$ InstAction action is used to dismiss an agent from a given role. In general the more specific type of AI, for instance the dutch-auction AI, inherits from the more generic AI, for example the auction AI, the list of its roles. This is expressed by the following axiom:

$$\text{isRoleOf} \circ \text{specializes}^{-} \sqsubseteq \text{isRoleOf}.$$

3.5 Norms

Norms in MAS have the following main characteristics (da Silva Figueiredo et al. 2010): (i) they are used to define at design time the *obligations*, *prohibitions*, *permissions*, and *institutional powers*, and they are all defined in terms of roles; (ii) they regulate the performance of *actions* and they are *active* during a period of time that can be expressed through *activation* and *deactivation events*. When they express obligations a deadline should be specified; (iii) norms specify *sanctions* for norms violations, *rewards* for norm fulfillment, or *sanctions* for the attempt to perform an institutional action without having the right institutional power or when specific preconditions are not satisfied.

A norm is an individual that belongs to the Norm class, which is the domain of a set of properties. First we define the properties for connecting the norm to the AI where it is defined, for specifying its type, and for expressing its debtor:

isNormOf: Norm $\rightarrow$ ArtificialInst;
hasNormDebtor: Norm $\rightarrow$ RoleName;
hasNormType: Norm $\rightarrow$ {obl,perm,prohib,power}

Like for roles, in general the more specific type of AI inherits from the more generic AI the list of its norms. This is expressed by the following axiom:

$$\text{isNormOf} \circ \text{specializes}^{-} \sqsubseteq \text{isNormOf}.$$

Norms have activation and deactivation events that are represented using classes of events, and they have a content that is a class of actions whose performance is regulated by the norm. The advantage of expressing them using classes instead of using specific individuals, is that the debtor agent at run-time will be able to exploit its autonomy and the possibility to perform automated reasoning on OWL ontologies for planning its action and choosing the best one among a set of actions that will fulfill the obligation. If the norm is an obligation it should have also a duration that will be used for computing the deadline within which the obliged action has to be performed. Given that a norm is an individual and its activation, content, and deactivation components are classes, we need to use OWL 2 *punning process*[10]

[10] http://www.w3.org/TR/owl2-new-features/#F12:_Punning.

Class $\leftrightarrow$ Individual (which allows to use the same term for both a class and an individual) for connecting a norm to its components, this by using the following properties:

hasNormActivation: Norm $\rightarrow$ Event;
hasNormContent: Norm $\rightarrow$ Action;
hasNormEnd: Norm $\rightarrow$ Event;
hasDuration: Norm $\rightarrow$ Integer.

At design time we have less information about the value of certain norm properties with respect to the information available at run-time. For example, the actual agents that will take part to the interaction on which the norms should be applied and the value of some parameters (i.e. the current ask-price of an auction) will become known only at run-time, when a space for running the auction is created. Every norm, defined at design time, will generate at run-time many specific obligations, prohibitions, permissions, and institutional powers, one for every agent that will start to play the role of debtor of a norm. We will model this aspect by having norms defined at design time and associated to specific AIs, which generate at run-time different types of *objects* belonging to specific institutional spaces. For modeling those concepts in the *Artificial Institution/Space Ontology* we create the Object class, which contains the classes Obligation, Prohibition, Permission, and InstPower. An object is connected to its space by means of the property belongsTo: Object $\rightarrow$ Space; Functional(belongsTo). In the definition of some components of norms we will use the special individual InstSpace(new-space) for being able to refer at design time to the specific space that will be generated by the AI where the norm belongs. When a norm generates a specific object in a specific space such a special individual has to be substituted with the individual used for representing the specific space.

In case the norm represents an obligation, following the approach presented in Fornara (2011), we will represent the specific obligations generated by the norm as individuals belonging to the class Obligation $\sqsubseteq$ Event. A specific *obligation* is treated as an event because it has associated the instant of time when it is created. This is fundamental for writing the axioms for deducing the state of obligations where we need to check that the event that activates it and the action that fulfills it are subsequent to its creation. In a similar way we model *prohibitions* that are used to express that an action belonging to its content should not be performed, obviously the axioms for deducing their fulfillment or violation are different with respect to the axioms of obligations. Specific *institutional power* objects can be created when an agent starts to play a given role and they may become active when some conditions are satisfied. Differently from obligations and prohibitions institutional powers are never fulfilled or violated but they are used by the environment component for mediating the attempts to perform institutional actions.

3.5.1 Example: the winner norm

In the specification of various types of auctions there is a norm that obliges the agent playing the role of auctioneer to assign the role of winner to a participant with certain characteristics. In the dutch-auction AI the winner is the agent that accepts the current auctioneer's ask-price. In the English auction the winner is the agent that did the highest bid during the run of the auction. This norm for the dutch-auction AI is formalized with the following assertions:

Norm(norm-winner-du);
isNormOf(norm-winner-du,dutch-auction);
hasNormType(norm-winner-du, obl);
hasNormDebtor(norm-winner-du,auctioneer);

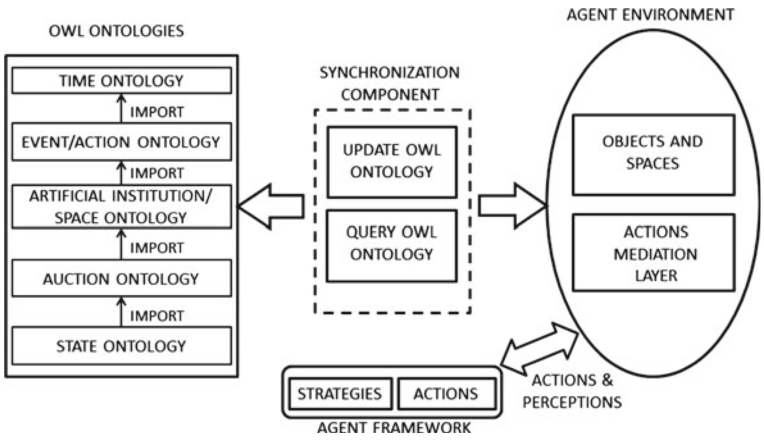

Fig. 3 Proposed Architecture

The event that activates the obligation represented by this norm is the action of accepting the current ask-price performed by one of the participants. This is an application dependent institutional action that only an agent playing the role participant in one specific run generated by the dutch-auction AI has the institutional power to perform. The effects of this action are to create an obligation, for the accepting agent, to pay to the auction house the value of the ask-price. The class of institutional actions used for accepting the ask-price of a run of this type of auction is represented with the class $\mathsf{AcceptAskPrice} \sqsubseteq \mathsf{InstAction}$. The activation event of the norm-winner-du is a class defined as follows:

$$\mathsf{StartEvent\text{-}winner\text{-}du} \equiv \mathsf{AcceptAskPrice} \sqcap \mathsf{performedIn} \ni \mathsf{new\text{-}space} \sqcap$$
$$\exists \mathsf{hasActor}.(\exists \mathsf{hasRole}.(\mathsf{hasRoleName} \ni \mathsf{participant} \sqcap \mathsf{isDefinedIn} \ni \mathsf{new\text{-}space}));$$
$$\mathsf{hasNormActivation(norm\text{-}winner\text{-}du,\ StartEvent\text{-}winner\text{-}du)};$$

The content class of norm-winner is defined as:

$$\mathsf{Content\text{-}winner\text{-}du} \equiv \mathsf{AssignRole} \sqcap \mathsf{hasAssignedRole}.(\mathsf{hasRoleName} \ni \mathsf{winner}) \sqcap$$
$$\exists \mathsf{hasAssignedAgent}.(\exists \mathsf{hasActor}^{-}.(\mathsf{AcceptAskPrice} \sqcap \mathsf{performedIn}. \ni \mathsf{new\text{-}space}));$$
$$\mathsf{hasNormContent(norm\text{-}winner\text{-}du,\ Content\text{-}winner\text{-}du)};$$

At run-time the debtor of this norm becomes known as soon as an agent starts to play in a specific space a role having as role name auctioneer. When this happens a specific obligation, used to model a specific realization of the norm norm-winner-du in the specific space, has to be created. The start event class and the content class of the new obligation are obtained substituting the individual new-space with the real name of the space. In order to express the fact that the auctioneer has n instant of time for declaring the winner, we assert hasNormDuration(norm-winner-du,n). At run-time this value will be used to set the interval of the generated obligations (see Fornara 2011 for more details on the *Obligation Ontology*).

4 Proposed architecture and related works

We are currently using the presented OWL model of artificial institutions and spaces for realizing a first prototype of a market-place. The architecture of such a prototype consists of four main building blocks (depicted in Fig. 3): (i) the *OWL 2 DL ontologies* used to represent AIs and spaces at design time and also at run time; (ii) the *agent environment* whose core is based on *GOLEM* platform (Bromuri and Stathis 2009) as we better explain below; (iii) the synchronization component among them; (iv) an *agent framework* for the development of cognitive agents able to plan their strategies, to perform actions on the environment and to perceive the state of the environment. GOLEM is an agent environment that can be used to create multi-agent applications where cognitive agents may interact. In our previous works Tampitsikas et al. (2012, 2011) we have extended the GOLEM platform in order to specify declaratively the agent environment and some institutional components, like norms, as a logic-based theory concretely realized using PROLOG. Such a GOLEM platform is used in our prototype for realizing inside the agent environment the spaces of interaction for the participating agents.

In the architecture of our prototype we decided to formalize Artificial Institutions and the spaces generated at run-time from those AIs in OWL 2 DL ontologies. This for having the advantages previously explained in Sect. 2, and in particular for the possibility to clearly specify our model in a W3C standard language for which many tools are available, for the advantage of using existing OWL ontologies and making our ontology re-usable in other applications, and for the decidability of the formal language. We decided also to use the GOLEM platform realized in PROLOG, a logic programming language, and in Object Event Calculus (OEC) (Kesim and Sergot 1996) for the realization of the first-class environment that the agents perceive and interact with. We did this choice because the OEC is suitable to represent the evolution in time of complex structures by means of events. In fact it determines the state of an object by assigning values to its attributes. Based on this property, it deals with the evolution of an object over time, parameterizing its attributes with the time at which these attributes hold their various values. Moreover PROLOG supports negation as failure a type of non-monotonic reasoning that in our framework is important for deducing the state of norms on the bases of the events that happen in the system.

As a consequence of those choices we need to introduce in our architecture a synchronization component (implemented in Java using OWL-API[11]) in charge of: (i) updating the OWL 2 DL ontologies with actions and events that happen in the environment and with the effects of agents' interactions; (ii) querying the OWL 2 DL ontologies on behalf of the agent environment for getting information contained in the ontologies, like for instance when an institutional action is performed for getting the list of institutional powers of the actor of the action. When information from OWL ontologies is retrieved into the agent environment, it is represented as first-class objects and spaces that can be perceived and manipulated by the software agents situated in a specific space.

To the best of our knowledge there are not other works where Semantic Web Technologies are used for modeling AI specifications and where AIs specifications are used at run-time for dynamically creating spaces of interactions situated in agent environments. There exist however a few interesting proposals that are partially correlated to our work and worth mentioning.

Sensoy et al. proposed the OWL-POLAR framework (Sensoy et al. 2012) whose focus is more on OWL policy representation and reasoning than on complete artificial institutions specification and on their use at run-time. Despite this fact, the OWL-POLAR is related to our work since the policies specified by the authors are equivalent to the norms in our AI definition. According to the authors, policies should be interpreted at run-time and for this reason they define a policy/norm language capable of: (i) clearly representing the different

[11] http://owlapi.sourceforge.net/.

types of policies in terms of related activities; (ii) reasoning on proper policy application; (iii) reasoning on policy validity and potential policies conflict. While the semantic representation of policies in OWL-POLAR shares common logical foundations with the norm representation in AIs, it differs on the fact that norms are situated in an agent environment. Norms in our approach are first-class objects situated in and limited by spaces. Spaces constitute the mechanism for monitoring the state of norms as well transforming their status. In advance, our system is event-based, meaning that the norms' states transition preconditions are not related to the normative state of the system in general but on classes of agent produced events.

Finally, the OWL-POLAR framework embeds strategies for resolving conflicts between policies. The main concept of anticipating conflicting policies lies in transforming the problem into an ontology consistency checking task. On the contrary, in AIs, norm interdependencies are realized and solved in the fist-class representation of the system and not in the OWL description of the MAS (Tampitsikas et al. 2012) making easier their anticipation. GOLEM queries the OWL 2 DL ontologies, depicts the state of the system, updates the environment and it is the environment as an active entity that automatically detects norm conflicts.

Another relevant related work is the paper (Zarafin et al. 2012) where the importance of integrating Semantic Web technologies and Multi-Agent Systems is discussed and a first draft of an OWL ontology of the MOISE organizational specification model is presented.

Finally another work related to our research line, but from a different perspective, is the JaCaMo platform, recently developed by Boissier et al. (2011). A JaCaMo multi-agent interaction system consists of three basic components:

- an agent organization modeled and programmed in the $\mathcal{M}$OISE Organization Management infrastructure (Hübner et al. 2002);
- a BDI agent programming framework realized with the Jason[12] a platform for the development of multi-agent systems that incorporates an agent oriented programming language;
- a distributed agent environment realized with CArtAgO, a framework and infrastructure for environment programming and execution in multi-agent systems (Ricci et al. 2009).

JaCaMo integrates these three frameworks by defining in particular a semantic link among concepts of the different dimensions (agent, environment and organization) at the meta-model and programming levels, in order to obtain a uniform programming model aimed at simplifying the combination of those dimensions when programming multi-agent systems. From the modeling point of view our approach differs from the approaches based on organization specification because our focus is on artificial institutions and spaces, that is, on the definition of the context and the conventions of the interaction and on the possibility to re-use such specifications for the realization of different open interaction systems.

Acknowledgments We thank Marco Colombetti, Michael Ignaz Schumacher, and Stefano Bromuri for the fruitful discussions on the model of AIs and spaces. Nicoletta Fornara's research is supported by Hasler Foundation project nr. 11115-KG and by Swiss State Secretariat for Education and Research (SER) project nr. C08.0114 within the COST Action IC0801 Agreement Technologies. Charalampos Tampitsikas's research is supported by the SER project "Open Interaction Frameworks, Towards A Governing Environment" within the COST Action IC0801 Agreement Technologies.

[12] http://jason.sourceforge.net/wp/.

Appendix A: OWL 2 DL

OWL 2 DL is a practical realization of a Description Logic known as $\mathcal{SROIQ}(\mathcal{D})$. It allows one to define *classes*, *properties*, and *individuals*. An OWL ontology consists of: a set of class axioms that specify logical relationships between classes, which constitutes the *Terminological Box* (*TBox*); a set of property axioms to specify logical relationships between properties, which constitutes a *Role Box* (*RBox*); and a collection of assertions that describe individuals, which constitutes an *Assertion Box* (*ABox*). Classes are formal descriptions of sets of objects (taken from a nonempty universe), and individuals can be regarded as names of objects of the universe. A class is either a *basic class* (i.e., an atomic class name) or a *complex class* build through a number of available *constructors*. Properties can be either *object properties*, which represent binary relations between objects of the universe, or *data properties*, which represent binary relationships between objects and data values (taken from XML Schema datatypes).

Through *class axioms* one may specify that subclass ($\sqsubseteq$) or equivalence ($\equiv$) relationships hold between certain classes, and that certain classes are disjoint. In particular, class axioms allow one to specify the domain and range of a property p (p: $\mathsf{A} \rightarrow \mathsf{B}$ where class A is the domain and class B is the range), and that a property is functional or inverse functional. *Property axioms* allow one to specify that a given property (or chain of subproperties) is a subproperty of another property, that two properties are equivalent, or that a property is reflexive, irreflexive, symmetric, asymmetric, or transitive. Finally, *assertions* allow one to specify that an individual a belongs to a class C: $\mathsf{C(a)}$, that an individual a is (or is not) related to another individual b through an object property R: $\mathsf{R(a,b)}$, that an individual is (or is not) related to a data value through a data property, or that two individuals are equal or different.

Complex classes can be specified by using Boolean operations on classes: $\mathsf{C} \sqcup \mathsf{D}$ is the union of classes, $\mathsf{C} \sqcap \mathsf{D}$ is the intersection of classes, and $\neg\, \mathsf{C}$ is the complement of class C. Classes can be specified also through *property restrictions*: (i) $\exists\, \mathsf{R.C}$ denotes the set of all objects that are related through property R to some objects belonging to class C, at least one; if we want to specify to how many objects an object is related we should write: $\leq \mathsf{nR}$, $\geq \mathsf{nR}$, $=\mathsf{nR}$ where n is any natural number; (ii) $\forall\, \mathsf{R.C}$ denotes the set of all objects that are related through R only to objects belonging to class C; (iii) $\mathsf{R} \ni \mathsf{a}$ denotes the set of all objects that are related to a through R.

References

Antoniou G, Harmelen Fv (2008) A semantic web primer (Cooperative Information Systems), 2nd edn. The MIT Press, Cambridge

Boissier O, Bordini RH, Hübner J, Ricci A, Santi A (2011) Multi-agent oriented programming with JaCaMo. Sci Comput Program (in press). Available online 29 October 2011

Bromuri S, Stathis K (2009) Distributed agent environments in the ambient Event Calculus. In: Proceedings of the third ACM international conference on distributed event-based systems, DEBS '09. ACM, New York, NY, USA, pp 1–12

Colombetti M, Fornara N Verdicchio M (2002) The role of institutions in multiagent systems. In: Proceedings of the workshop on knowledge based and reasoning agents, VIII Convegno AI* IA, vol 2002

da Silva Figueiredo K, Torres da Silva V, de Oliveira Braga C (2010) Modeling norms in multi-agent systems with NormML. In: De Vos M, Fornara N, Pitt JV, Vouros G (eds) COIN 2010 international workshops, COIN@AAMAS 2010, Toronto, Canada, May 2010, COIN@MALLOW 2010, Lyon, France, August 2010, Revised selected papers, vol 6541 of LNCS. Springer, pp 39–57

d'Inverno M, Luck M, Noriega P, Rodriguez-Aguilar JA, Sierra C (2012) Communicating open systems. Artif Intell 186:38–94

Esteva M, Rodr Ãguez-Aguilar JA, Sierra C, Garcia P, Arcos J (2001) On the formal specification of electronic institutions. In: Dignum F, Sierra C (eds) Agent mediated electronic commerce, vol 1991 of Lecture Notes in Computer Science, Springer, Berlin, pp 126–147

Fornara N (2011) Semantic agent systems: foundations and applications volume 344 of studies in computational intelligence chapter 2. In: Elçi A, Orgun MA (eds) Specifying and monitoring obligations in open multiagent systems using semantic web technology. Springer, Berlin, pp 25–46

Fornara N, Colombetti M (2009a) Specifying and enforcing norms in artificial institutions. In: Baldoni M, Son T, van Riemsdijk M, Winikoff M (eds) Declarative agent languages and technologies VI, vol 5397 of Lecture Notes in Computer Science. Springer, Berlin, pp 1–17

Fornara N, Colombetti M (2009b) Specifying artificial institutions in the Event Calculus. In: Dignum V (ed) Handbook of research on multi-agent systems: semantics and dynamics of organizational models, information science reference, chapter XIV. IGI Global, pp 335–366

Fornara N, Colombetti M (2010) Representation and monitoring of commitments and norms using OWL. AI Commun 23(4):341–356

Fornara N, Okouya D, Colombetti M (2012) Using OWL 2 DL for expressing ACL content and semantics. In: Cossentino M, Kaisers M, Tuyls K. Weiss G (eds) Systems multi-agent, 9th European workshop, EUMAS 2011, Maastricht, The Netherlands, November 14–15, (2011) Revised Selected Papers, vol 7541 of Lecture Notes in Computer Science, Springer, Berlin, pp 97–113

Fornara N, Viganò F, Colombetti M (2007) Agent communication and artificial institutions. Autonom Agents Multi Agent Syst 14(2):121–142

Fornara N, Viganò F, Verdicchio M, Colombetti M (2008) Artificial institutions: a model of institutional reality for open multiagent systems. Artif Intell Law 16(1):89–105

Hitzler P, Krötzsch M, Rudolph S (2009) Foundations of semantic web technologies. Chapman & Hall/CRC, London

Hübner JF, Sichman JS, Boissier O (2002) A model for the structural, functional, and deontic specification of organizations in multiagent systems. In: Proceedings of the 16th Brazilian symposium on artificial intelligence: advances in artificial intelligence, SBIA '02. Springer, London, UK, pp 118–128

Kesim FN, Sergot M (1996) A logic programming framework for modeling temporal objects. IEEE Trans Knowl Data Eng 8(5):724–741

Okuyama FY, Bordini RH, da Rocha Costa AC (2008) A distributed normative infrastructure for situated multi-agent organisations. In Proceedings of AAMAS '08, vol 3. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, pp 1501–1504

Ricci A, Piunti M, Viroli M (2011) Environment programming in multi-agent systems: an artifact-based perspective. Auton Agents Multi Agent Syst 23(2):158–192

Ricci A, Piunti M, Viroli M, Omicini A (2009) Environment programming in CArtAgO. In: Bordini RH, Dix J, Fallah-Seghrouchni AE (eds) Multi-agent programming: languages, platforms and applications, vol 2. Springer, Berlin, pp 259–288

Searle JR (1995) The construction of social reality. Free Press, New York

Sensoy M, Norman TJ, Vasconcelos WW, Sycara K (2012) OWL-POLAR: a framework for semantic policy representation and reasoning. Web Semant Sci Serv Agents WWW 12–13:148–160

Tampitsikas C, Bromuri S, Fornara N, Schumacher MI (2012) Interdependent artificial institutions in agent environments. Appl Artif Intell 26(4):398–427

Tampitsikas C, Bromuri S, Schumacher MI (2011) MANET: a model for first-class electronic institutions. In: Cranefield S, van Riemsdijk M, Vazquez-Salceda J, Noriega P (eds) Coordination, organizations, instiutions, and norms in agent system VII volume 7254 of LNCS page to appear. Springer, Berlin

Weyns D, Omicini A, Odell J (2007) Environment as a first class abstraction in multiagent systems. Auton Agents Multi Agent Syst 14(1):5–30

Zarafin AM, Zimmermann A, Boissier O (2012) Integrating semantic web technologies and multi-agent systems: a semantic description of multi-agent organizations. In: Ossowski S, Toni F, Vouros G (eds) Proceedings of the first international conference on agreement technologies, vol 918. Dubrovnik, Croatia, pp 296–297, October 15–16, 2012. CEUR workshop proceedings