Using OWL Artificial Institutions for dynamically creating Open Spaces of Interaction*

Nicoletta Fornara[1]**, Charalampos Tampitsikas[1,2]***

[1] Università della Svizzera italiana,
via G. Buffi 13, 6900 Lugano, Switzerland
{nicoletta.fornara, charalampos.tampitsikas}@usi.ch,
[2] University of Applied Sciences Western Switzerland, Institute of Business Information Systems, 3960 Sierre, Switzerland

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. In our view those systems can be efficiently and effectively modeled as a set of physical and institutional spaces of interaction. In a distributed open system, 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. Spaces are also crucial for their functional role of keeping track of the state of the interaction, and for monitoring and enforcing norms. Given that 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 using Semantic Web Technologies. Then we will describe the mechanisms for concretely using AIs specification for concretely realizes spaces of interaction. We will exemplify this process by formalizing the components of the auction AI and of the spaces required for running concrete auctions.

1 Introduction

Open interaction systems play a crucial role in agreement technologies because they are software devised for enabling autonomous agents (software or human)

* AT2012, 15-16 October 2012, Dubrovnik, Croatia. Copyright held by the author(s).
** Supported by the Hasler Foundation project nr. 11115-KG and by the SER project nr. C08.0114 within the COST Action IC0801 Agreement Technologies.
*** Supported by the Swiss State Secretariat for Education and Research (SER) within the COST action IC0801 "Agreement Technologies", project title "Open Interaction Frameworks, Towards A Governing Environment".

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/agreements among multiple parties and to their execution, monitoring, and enforcement. The most important aspect of this type of systems is that it is not known in advance 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 [4]. Open interaction systems are crucial for the design, development, and deploy 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 describes 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 behavior of the interacting agents, *norms* are a fundamental part of open systems. They are used to express obligations, prohibitions, permissions, and institutional powers, that regulate the interaction of the agents. At design time norms are expressed using *roles*.
- 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 a contextual relevance of norms has to be taken into account [13].

In our past works we have proposed to use Artificial Institutions (AIs) [8] for the design of open interaction systems and recently we started to study how to formalize some components of AIs using Semantic Web Technologies [9, 6, 10]. In parallel we started to study how to integrate the model of AIs with the notion of *environment* [**?**,14] and in particular with the notion of *institutional space* of interaction [18]. Extending existing notions of environment is crucial for being able to manage the effects and the perception of social and institutional interactions among agents, and in particular the performance of institutional actions [11]. A crucial advantage of using the notion of agent environment in modelling open systems is the availability of environment frameworks like GOLEM [1] and CArtAgO [15] that can become fundamental components in the concrete realization of prototypes of open interaction systems.

This paper is organized as follows. In Section 2 we introduce the notion of Artificial Institution (AI), of space of interaction, and their connections. In Section 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 [2] and hierarchies of the spaces realized using such AIs. Finally in Section 4 the architecture of the prototype that we are implementing for testing the proposed model, with the final goal of realizing a complete marketplace, is briefly described.

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* [18]. 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. An environment for multi-agent systems (MAS) is composed by multiple *spaces* where *objects* and *agents* are situated. This abstract concept of space presents some interesting similarities with the notion workspace introduced in some environments studies [1] and of scene used in Electronic Institutions [4].

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 roles, 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*. 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* whose value can be changed thanks to the performance of *institutional actions* and thanks to the common acceptance of the meaning of those actions from the agents belonging to the space where the objects are situated. For example a run of an auction that can be opened and closed, or an agreement/contract between agents whose attributes can be filled 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 (used to express obligations, prohibitions, permissions) it is necessary to extend the functionalities of spaces. An institutional space that contains institutional objects has 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 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 it but the effects of the action will not change the state of the state where it is performed;
- *monitoring* the interactions and check if they are compliant with a given set of norms used to express obligations, prohibitions, and permissions;
- *enforcing* the norms for example by applying sanctions [7] 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) [11, 8]. This approach, from the software engineering point of view, has the enormous advantage of making it possible to use many times existing specifications of AIs, for example the AI specification of a given type of auction for realizing an 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[3] [12]. The model of AI that we propose in this paper is inspired from the model presented in [8] where AI are formalized using Event Calculus. Obviously given

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

that in this proposal we adopt other formal languages we will need to change some parts of the model (for example an advantage of this model is that the content of norms are classes of possible actions instead of being a specific fluent) and to extend it to take into account its connections with the environment components, that is with spaces and objects. Then as a second step we will study and describe the mechanisms for concretely using the specification of a generic AI for concretely realize and execute 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 [1].

There are many advantages in using Semantic Web Languages for the specification and realization of an open interaction systems with respect to the adoption of other formal languages as proposed in other approaches [4]. 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 programming and editing ontologies. Moreover given that OWL 2 DL is a decidable fragment of FOL there are several reasoners available[4] for reasoning on OWL specifications. 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 interact with other agents by using different open systems, even if they have different set of norms, rules, and context of interaction.

3 Formal specification of AIs and spaces using OWL

In this section we formalize, using Semantic Web Technologies, the concepts, properties, and axioms required for the specification of artificial institutions at design time. We also propose a formal specification of the concepts required for the definition and dynamic evolution of institutional spaces at run-time and the process for dynamically creating them using AI specifications. Those concepts, properties, and axioms used for defining AIs and spaces will be mainly defined in the TBox (Terminological Box) of a set of OWL 2 DL ontologies introduced for representing different components of the proposed model and created using Protégé ontology editor[5]. The process for actually creating and destroying spaces at run-time, is concretely realized by querying the content of the ontology where

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

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

the model of AIs is stored and by manipulating the content of the ontology (the *State Ontology*) used for representing the state of the existing spaces of interaction. Given that it is impossible to add individuals to ontologies using OWL 2 DL axioms this process is implemented with a program that uses OWL libraries, like OWL-API, for accessing OWL 2 DL ontologies. The concepts introduced will be exemplified by formalizing some of the components of the Dutch auction AI and of the space generated from this AI.

In order to be able to use already existing ontologies, like for example the W3C OWL *Time Ontology*[6], and in order to make 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 (*Event/Action Ontology*); an application independent ontology where the concepts required for describing artificial institutions and spaces are formalized (*Artificial Institution/Space Ontology*); an ontology for describing different type of auctions using the proposed model of AI (*Auction Ontology*), and an ontology used for describing at run-time the state of the spaces of interaction dynamically created (*State Ontology*). Those ontologies are connected by an "import" relationship as depicted in Figure 1.

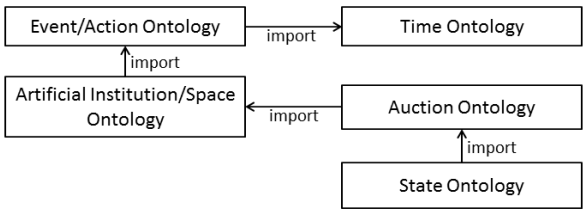

Fig. 1. Import relationship among the proposed OWL 2 DL ontologies.

The *Time Ontology* is used to represent *instants of time*, *intervals*, and relationships among them. The *Event/Action Ontology* [6] is used to represent events and actions (with the classes Event and Action) and it imports the *Time Ontoloy*. An event is connected to the instant of time when it happens by the atTime: Eventuality $\rightarrow$ TemporalEntity property. Every event has an atTime value: Event $\equiv \exists$atTime.Instant. Actions are particular type of events having an actor: Action $\sqsubseteq$ Event; Action$\equiv \exists$hasActor.Agent. The class ChangeEvent $\sqsubseteq$ Event represents the events due to the change of the value of a property. The class TimeEvent $\sqsubseteq$ 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 Elapsed $\sqsubseteq$ Instant. If events/actions are associated to an elapsed instant of time, they are actually happened, otherwise they are

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

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 interaction, this with the goal of being able to monitor the behaviour of the agents.

3.1 Institutional actions

Institutional actions [11] (represented with the class InstAction $\sqsubseteq$ 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 [16], otherwise the effects of the action are empty. In our previous institutional models [11, 8] 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. Differently in the model presented in [18] 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 properties 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; Fun(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 Section 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-acution, auction); specializes(dutch-acution, 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. This hierarchy of AIs is reflected in an equivalent hierarchy of the classes created by the spaces that realize a given AI. This is expressed in the following axioms:

isRealizationOf$\ni$eng-auction $\sqsubseteq$ isRealizationOf$\ni$auction;

isRealizationOf$\ni$dutch-auction $\sqsubseteq$ isRealizationOf$\ni$auction

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 these axioms is 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 institutional space run01 that realizes the auction AI. This is a ternary predicates that cannot be expressed in OWL: 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 belongs 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[7] (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 → RoleName; Fun(hasRoleName);
isDefinedIn: Role → InstSpace; Fun(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 → Role allows to represent the fact an agent plays a give 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 Figure 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 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 → Agent and hasAssignedRole: AssignRole → 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: isRoleOf $\circ$ specializes$^-$ $\sqsubseteq$ isRoleOf.

3.5 Norms

Norms in MAS have the following main characteristics [3]: (i) they are used to define at design time the *obligations*, *prohibitions*, *permissions*, and *institutional*

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

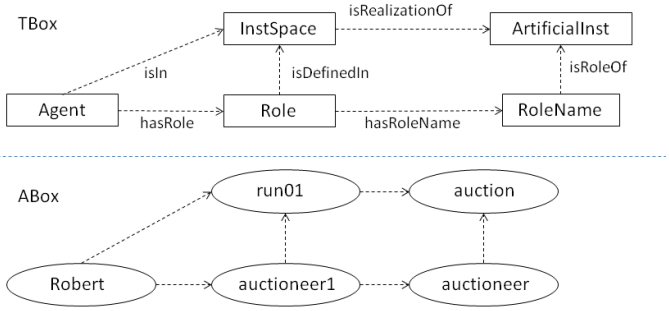

Fig. 2. Classes, properties, and individuals related to the notion of role.

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:

isNormOf $\circ$ specializes^{-} $\sqsubseteq$ 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 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. 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*[8] 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.

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

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; Fun(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 [6], 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 start 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.

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);

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 AcceptAskPrice $\sqsubseteq$ InstAction. The activation event of the norm-winner-du is a class defined as follows:

StartEvent-winner-du $\equiv$ AcceptAskPrice $\sqcap$ performedIn$\ni$new-space $\sqcap$
$\exists$hasActor.($\exists$hasRole.(hasRoleName$\ni$participant $\sqcap$ isDefinedIn$\ni$new-space));

hasNormActivation(norm-winner-du, StartEvent-winner-du);

The content class of norm-winner is defined as:

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

hasNormContent(norm-winner-du, Content-winner-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 [6] for more details on the *Obligation Ontology*).

4 Architecture of the prototype

We plan to use the presented OWL model of artificial institutions and spaces for realizing a first prototype of a market-place. The architecture of this prototype consists of three main building blocks (depicted in Figure 3): (i) the *OWL 2 DL ontologies* used to represent AIs and spaces; (ii) the *agent environment* whose core is based on *GOLEM* platform [1]; (iii) the synchronization component among them.

GOLEM is an agent environment that can be used to create multi-agent applications where cognitive agents may interact. We have extended GOLEM in order to specify declaratively the agent environment as a logic-based theory (First-Order Logic based on Prolog) [18]. GOLEM is used in our prototype for realizing inside the agent environment the spaces of interaction for the participating agents. Given that, for the advantages previously explained, we decided to formalize AIs and spaces using OWL ontologies, in the architecture of our prototype we need to introduce a synchronization component (implemented in Java using OWL-API[9]) in charge of: (i) updating the OWL 2 DL ontologies with

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

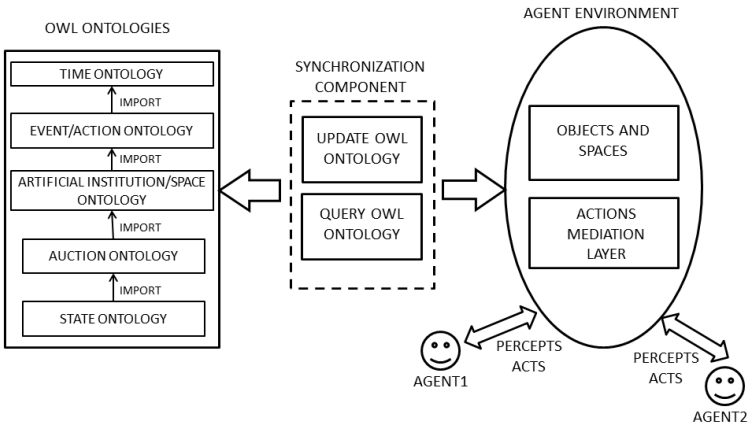

Fig. 3. Architecture of the prototype.

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. When information from OWL ontologies is retrieved into the agent environment, its are 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. An interesting proposal that is correlated to this work is the OWL-POLAR framework [17] whose focus is more on OWL policy representation and reasoning than on complete artificial institutions specification and use at run-time.

Acknowledgments We thank Marco Colombetti, Michael Ignaz Schumacher, and Stefano Bromuri for the fruitful discussions on the model of AIs and spaces.

References

1. S. Bromuri and K. Stathis. Distributed agent environments in the ambient event calculus. In *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, DEBS '09, pages 1–12, New York, NY, USA, 2009. ACM.
2. M. Colombetti, N. Fornara, and M. Verdicchio. The role of institutions in multiagent systems. In *Proceedings of the Workshop on Knowledge based and reasoning agents, VIII Convegno AI* IA*, volume 2002, 2002.
3. K. da Silva Figueiredo, V. T. da Silva, and C. de O. Braga. Modeling Norms in Multi-agent Systems with NormML. In M. D. Vos, N. Fornara, J. V. Pitt, and G. A. Vouros, editors, *COIN 2010 International Workshops, COIN@AAMAS 2010,*

Toronto, Canada, May 2010, COIN@MALLOW 2010, Lyon, France, August 2010, Revised Selected Papers., volume 6541 of *LNCS*, pages 39–57. Springer, 2010.
4. M. d'Inverno, M. Luck, P. Noriega, J. A. Rodriguez-Aguilar, and C. Sierra. Communicating open systems. *Artificial Intelligence*, 186(0):38–94, 2012.
5. C. A. Ellis, S. J. Gibbs, and G. Rein. Groupware: some issues and experiences. *Commun. ACM*, 34(1):39–58, Jan. 1991.
6. N. Fornara. Specifying and Monitoring Obligations in Open Multiagent Systems using Semantic Web Technology. In A. Elçi, M. T. Kone, and M. A. Orgun, editors, *Semantic Agent Systems: Foundations and Applications*, volume 344 of *Studies in Computational Intelligence*, chapter 2, pages 25–46. Springer-Verlag, 2011.
7. N. Fornara and M. Colombetti. Specifying and Enforcing Norms in Artificial Institutions. In M. Baldoni, T. Son, M. van Riemsdijk, and M. Winikoff, editors, *Declarative Agent Languages and Technologies VI*, volume 5397 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin / Heidelberg, 2009.
8. N. Fornara and M. Colombetti. Specifying Artificial Institutions in the Event Calculus. In V. Dignum, editor, *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*, Information science reference, chapter XIV, pages 335–366. IGI Global, 2009.
9. N. Fornara and M. Colombetti. Representation and monitoring of commitments and norms using OWL. *AI Commun.*, 23(4):341–356, 2010.
10. N. Fornara, D. Okouya, and M. Colombetti. Using OWL 2 DL for expressing ACL Content and Semantics. In M. Cossentino, M. Kaisers, K. Tuyls, and G. Weiss, editors, *EUMAS 2011 Selected and Revised papers*, LNCS, page to appear, Berlin, Heidelberg, 2012. Springer-Verlag.
11. N. Fornara, F. Viganò, and M. Colombetti. Agent communication and artificial institutions. *Autonomous Agents and Multi-Agent Systems*, 14(2):121–142, 2007.
12. P. Hitzler, M. Krötzsch, and S. Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009.
13. F. Y. Okuyama, R. H. Bordini, and A. C. da Rocha Costa. A distributed normative infrastructure for situated multi-agent organisations. In *Proceedings of AAMAS '08 - Volume 3*, pages 1501–1504, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.
14. A. Ricci, M. Piunti, and M. Viroli. Environment programming in multi-agent systems: an artifact-based perspective. *Autonomous Agents and Multi-Agent Systems*, 23(2):158–192, Sept. 2011.
15. A. Ricci, M. Piunti, M. Viroli, and A. Omicini. Environment Programming in CArtAgO. In R. H. Bordini, M. Dastani, J. Dix, and A. E. Fallah-Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms and Applications*, volume 2, pages 259–288. Springer, 2009.
16. J. R. Searle. *The construction of social reality*. Free Press, New York, 1995.
17. M. Sensoy, T. J. Norman, W. W. Vasconcelos, and K. Sycara. Owl-polar: A framework for semantic policy representation and reasoning. *Web Semant.*, 12-13:148–160, Apr. 2012.
18. C. Tampitsikas, S. Bromuri, N. Fornara, and M. I. Schumacher. Interdependent Artificial Institutions In Agent Environments. *Applied Artificial Intelligence*, 26(4):398–427, 2012.
19. D. Weyns, A. Omicini, and J. Odell. Environment as a first class abstraction in multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 14(1):5–30, 2007.