An Infrastructure for the Design and Development of Open Interaction Systems

Daniel Okouya[1], Nicoletta Fornara[1], and Marco Colombetti[1,2]

[1] Università della Svizzera Italiana,
via G. Buffi 13, 6900 Lugano, Switzerland
{daniel.okouya,nicoletta.fornara,marco.colombetti}@usi.ch
[2] Politecnico di Milano,
piazza Leonardo da Vinci 32, 20135 Milano, Italy
marco.colombetti@polimi.it

Abstract. We propose an infrastructure for the design and development of Open Interaction Systems (OISs), based on solutions from the areas of Service Oriented Architecture, Semantic Technologies and Normative Multiagent Systems, in particular the OCeAN metamodel of Artificial Institution. OISs are open to diverse types of participants (i.e., software agents), and enable them to interact with each other to achieve their objectives. To do so the participants are allowed to interact in compliance with previously agreed-upon regulations provided by the system and on the basis of the semantics of the communicative acts performed, both of which are enforced by the system. The infrastructure we propose involves four layers: (i) the Messaging Layer, which enables observable ACL message exchanges between heterogeneous participants while respecting ownership boundaries; (ii) the Core Service Layer, which enables the participants with performing observable non-communicative actions relevant to the ongoing application (iii) the Bridging Layer, in charge of interpreting the participants' actions in a form suitable for regulation; and (iv) the Regulation Layer, which holds the regulations and enforces them with respect to the participants' activities.

Keywords: Open Interaction Systems, Artificial Institution, Ontologies, Normative Systems, Agent Communication

1 Introduction

Open Interaction Systems (OISs) are distributed systems which diverse types of participants (i.e., software agents) can freely join with the goal of interacting with each other to achieve their objectives. To do so the participants are allowed to interact by exchanging messages with rigorously defined syntax and semantics, in compliance with previously agreed-upon norms provided by the system; both the norms and the communication language are enforced by the system.

In our past work we have proposed the OCeAN metamodel [12] for the specification of OISs. In this paper we describe an infrastructure, currently under development, for the actual implementation of such systems. In designing this infrastructure we aim at guaranteeing openness and interoperability, while keeping as close as possible to those technologies that are sufficiently mature and stable, and are already adopted by a large industrial community. Among such technologies we include standard Service Oriented technologies [4] and Semantic Web technologies [14].

The infrastructure we propose involves four layers: (i) the *Messaging Layer*, which enables heterogeneous participants to interact with each other through communicative actions while respecting ownership boundaries; (ii) the *Core Service Layer*, which allows the participants to exploit the support services offered by the OIS to perform non-communicative actions; (iii) the *Bridging Layer*, in charge of interpreting the participants' actions in a form suitable for regulation; and (iv), the *Regulation Layer*, which holds the norms regulating the interactions and enforces them relative to the participants' actions. More specifically:

- the Messaging Layer provides a messaging protocol based on standard technologies (HTTP, SOAP, WSDL) and uses Web Service technologies for the transfer of messages between participants, by prescribing the use of a specific message transfer service exposed via WSDL; messages realize communicative or institutional acts and comply with OCeAN-ACL [10], an Agent Communication Language strongly based on Semantic Web technologies, and on OWL 2 DL in particular;
- the Core Service Layer makes certain complementary services available to the participants (e.g., an OIS realizing an e-marketplace may offer services related to payment, product delivery, and so on) to perform observable non-communicative actions relevant to the ongoing application;
- the Bridging Layer interprets the participants' communicative and non-communicative actions in a form suitable for regulation. Coherently with the OCeAN metamodel, such acts either result into commitments (like in the case of acts of informing, requesting, etc.) or are regarded as attempts to perform institutional actions relying on suitable count-as rules;
- finally, the Regulation Layer realizes a normative context (again according to the OCeAN metamodel), that is, a set of artificial institutions specifying the institutional actions that can be performed and the set of norms that have to be followed.

In this paper we provide a detailed specification of all layers and describe the implementation, currently under development, of an infrastructure oriented to the implementation of an open e-marketplace. The paper is organized as follows. In Section 2 we describe the functionalities pertaining to the Messaging Layer and how we implement them by exploiting standard Web Service technology. In Section 3 we briefly sketch how the core services offered by the OIS can be actually realized, considering an e-marketplaces as an example. In Section 4 we describe the functionalities pertaining to the Regulation Layer and how we implement them by exploiting Semantic Web technologies, and OWL ontologies in particular. In Section 5 we explain how relevant events taking place at either the Massaging or the Core Service Layer are made available to the Regulation Layer. In Section 6 we review some related works. Finally in Section 7 we draw some conclusions and briefly describe our plans for future work.

2 The Messaging Layer

In an OIS, a large part of the participants' interaction is carried out through the exchange of suitable messages. Therefore the bottom layer of our infrastructure provides the means to enable heterogeneous participants to interact with each other by exchanging messages in a fully interoperable fashion. In addition, it does so in such a way that it ensures the observability of these interactions, to the purpose of regulation.

To this end our infrastructure integrates principles from Service Oriented Architecture (SOA) and from Multiagent Systems (MAS). First, a message transfer approach is prescribed that is neutral to the internals of the participants, and leverages standard technologies to facilitate widespread adoption. This is in contrast with approaches based on some of the most well known ready-to-use messaging technologies like JMS[3], RMI[4], and CORBA [17], which bind either to a particular programing language [13] or to a programing language paradigm [17]. Such approaches do not fully decouple the end point implementation from the messages, thus limiting interoperability [18, 2]. Our architecture, following SOA's principles of loose coupling, solely prescribes a message format together with its transfer protocol, both of them strictly decoupled from the end point implementation, while insisting as much as possible on the adoption of standard technologies [5, 4, 21].

Next, we add to the architectural prescriptions, the combination of the SOA concept of a message, as comprised of carrying and content information, with the MAS idea of a powerful and flexible Agent Communication Language (ACL). More precisely, we take the content part of a SOA message to represent the various components of a suitably designed ACL. Thus, together with the neutral messaging protocol delineated above, the participants are enabled to interact through the performance of communicative acts, in a totally interoperable fashion.

Finally, to enable the observability of the communicative acts performed by the participants to the purpose of regulation, this layer further mandates a *Communication Channel* (CC) to mediate message exchanges between participants. More precisely, to communicate with other registered participants, a registered participant shall send its messages to the address of the CC, with the name of the desired participants as the recipients. The CC receives the message which, if approved by the regulative process of the infrastructure, is then delivered to the intended participants. If the message is not approved, the CC rejects it and sends a suitable explanation to the sender.

[3] `http://docs.oracle.com/javaee/6/tutorial/doc/bncdr.html`
[4] `http://docs.oracle.com/javase/7/docs/technotes/guides/rmi/index.html`

These architectural requirements are met in the infrastructure as follows. In the first place, the infrastructure provides for a messaging protocol based on standard neutral technologies: HTTP, SOAP[5], and WSDL[6]. In other words, Web Service technology is adopted for message transfer between participants, by specifying a message transfer service, exposed via WSDL, in which HTTP is used for the transport of messages and SOAP for their structure. Our choice is motivated by the fact that this technology represents a standard approach for making available over the network functionalities that are triggered or delivered by exchanging messages.

In the second place, the infrastructure then specifies the body of the SOAP messages as messages of our ACL[7]. From the syntactic point of view, the ACL we propose is very close to KQML[8] and FIPA ACL[9], from which however it substantially departs as far as semantics is concerned (see Section 5). As with FIPA standards, our ACL come with a separate Content Language (CL). Our CL is defined as an OWL Ontology, the *Content Language Ontology*[10]. It plays a role similar to FIPA-RDF.

Thus, realizing the first two requirements, we define a WSDL file with only one service, which is the delivery of an ACL message, carried in the body of a SOAP message. The WSDL contract represents any form of message that can be exchanged between entities of our OIS, with the requirement that the massage contains the address to reply to according to the same contract. Communication between participants is only allowed through the use of this service; consequently, all participants are required to be equipped with a suitable communication module, composed of: (i), a *listening-point*, that is, a web-service provider exposing a message delivery service defined according to our WSDL contract; and (ii), a *talking-point*, that is, a web-service client that requests the delivery of a message in conformance with that contract [9].

A crucial advantage of this approach is the provision of a messaging protocol in the form of a WSDL contract, which is both human readable and machine processable. Such a contract can be easily handled with the support of runtime frameworks coming along with Web Service technology, such as Apache CXF [1, 15]. We use CXF to automatically generate the core of the communication module of the participating component of our infrastructure; hence anyone can easily generate the necessary facilities to handle the transmission of messages abiding to the exposed messaging protocol and adapt it to their need, in order to participate in the OIS.

Finally, to deal with message transfer the infrastructure provides an implementation of the CC as a Java component, developed with CXF as exposed above.

[5] `http://www.w3.org/TR/soap`
[6] `http://www.w3.org/TR/wsdl`
[7] `http://www.people.lu.unisi.ch/okouyad/AclOverSoapHttpMP.wsdl`
[8] `http://www.csee.umbc.edu/csee/research/kqml/`
[9] `http://www.fipa.org/repository/aclspecs.html`

3 The Core Service Layer

As we have already remarked, in OISs, a large part of the participants' interaction is carried out by exchanging suitable messages; under the circumstances of our Infrastructure, as a result of its messaging layer, it can be further stated that it is mainly by performing communicative acts, that is, by exchanging ACL messages. However, most types of applications will also require the executions of actions that are not strictly speaking communicative. We identify these actions as *non-communicative acts* and classify them into two categories. First, non-communicative acts that concern the interaction between the participants and certain components of the infrastructure, designed to provide support to the participants' activities; as we shall see, these non-communicative acts are typically *application-independent.* Second, non-communicative acts occurring between the participants that concern certain *application-specific* interactions.

More specifically, on the one hand, some of the *application-independent* non-communicative actions are intended to support the enforcement of ownership-boundaries between participants, enabling them to connect with each other without introducing dependencies. To this purpose, the infrastructure provides for a *Registry* component, within which the participants can be listed or unlisted by performing actions such as *registering or deregistering* their identities. Although the registration and deregistration processes do presuppose the performance of certain communicative actions (more precisely, of the request to be registered or deregistered), the actions of registering or deregistering a participant are not themselves communicative. Rather, they are non-communicative actions made available to the participants by the infrastructure, through the provision of services that may be invoked using communicative actions (requests).

On the other hand, some of the application-specific activities, that is, some of the activities that are carried out between the participants, may also require more than the sole performance of communicative actions. That is, the nature of the interactions may demand the performance of *application-specific* non-communicative actions, which, as in the case of communicative acts (i.e., the other actions occurring between the participants), must also be made observable to the infrastructure. For example, in an e-marketplace system, when engaging in a purchasing activity, after settling a contract with communicative acts, the buyer may be required to carry out a payment, while the seller may be required to deliver a product. These are both non-communicative acts inherent to the purchasing activity, and as such must also be visible to the infrastructure.

Thus, the objective of this Layer is to equip the infrastructure so that: (i), it enables the participants with performing all the infrastructure-specific *non-communicative actions* belonging to the direct interactions between the participants and the infrastructure itself; and (ii), it can observe the performance of the application-specific *non-communicative actions* inherent to some of the activities occurring between the participants.

To this end, first, as suggested above, for those non-communicative actions that are *application-independant* (i.e., infrastructure-specific) at present our infrastructure provides a Registry component, implemented in Java, to serve as a

White Pages Service. It provides among others, for the *registering* and *deregistering* actions. As this component is endowed with ACL-processing capabilities, participants can request its services using ACL messages.

Next, for the *non-communicative* actions that are application-specific, in unison with the approach used for the communicative acts (i.e., that the observation of the actions occurring between the participants goes through the mediation of their performance), the infrastructure also proposes to mediate them. In this respect, however, the core service layer proceeds differently from the messaging layer. Indeed, the different communicative actions that can be performed by the participants are the same across applications; thus, the observation process necessary to handle them is also application-independent, and therefore can be achieved by a generic component: the Communication Channel. In contrast, non-communicative acts occurring between participants are typically application-dependent: their presence, what they achieve, and how they achieve it, always depend on the application being realized. Indeed, on one hand, unlike communicative acts that ought to be always available to the participants, the presence of those non-communicative actions is application-specific; for instance, the availability of a delivery action would be irrelevant to an application that does not deal with delivery, such as an e-market for computational services. On the other hand, when present, the performance of those non-communicative acts can substantially vary depending on the requirements of the applications in which they are performed: as illustrated by the case of a payment, while one application may require a system like PayPal, another one may require a direct bank-to-bank transfer or a cheque payment, which would require to go through different steps and to supply different information. Another important difference is that, unlike communicative actions, non-communicative actions can also vary in nature, that is, they can be electronic, physical or involve both aspects.

Hence, to mediate them, the infrastructure must proceed carefully taking into account their fundamental application-oriented characteristics, as well as their nature that can involve any combination of physical and electronic aspects. To achieve this, the architecture prescribes that the Core Service Layer provides for the incorporation of *observable* application-specific components, offering to the participants specific services of mediation for those application-specific, non-communicative actions. These components must be such that they seemingly interoperate with the participants for the invocation of the actions that they mediate, whose performances must be observable.

To that purpose, on the one hand, this layer specifies the interfaces of the mediating components, so that the relevant parts of the infrastructure can take into account the performance of the non-communicative actions they are in charge of. On the other hand, it prescribes the characteristics that the components must posses so that their services can be seemingly consumed. In support of that latter point, the layer mandates the use of communicative acts to invoke their mediation services. That is, while the *message-transfer mediation service* of the messaging Layer is invoked using a SOAP message (as a typical web-service), theses services are invoked using an ACL message, that is, the content of the soap message. It brings the advantage of providing a unique flexible protocol for the invocation of these services, independently of their nature and level of complexity. This implies that those mediating components must be able to process certain ACL messages.

Meanwhile, it is important to mention that as part of the service they provide, our infrastructure does not require that mediating components directly perform the non-communicative actions they supply: indeed they may do so, or guarantee their performance by some external systems, or simply acknowledge their external realization as informed by a set participants who have agreed to use an external service for their interaction. In this regard, the layer classifies theses services into two distinct categories: internal services and external services. In the former case, the service is internally managed by the component itself; this means that when directly asked by a participant, the component takes charge of the execution of the activities involved in the service. In the latter case, which represent a very decentralized approach providing more freedom to the participants, the execution is guaranteed by the participants themselves, which then inform the infrastructure of the results. Here mediation plays the role of a neutral authority that acknowledges the realization of services taking place out of its direct control, according to the specific rules governing the application.

4 The Regulation Layer

Once heterogeneous participants, possibly belonging to different owners, can interact with each other as exposed above, it is necessary that they get provided with some form of harnessing framework defining norms that regulate their interactions. This is particularly important as it allows the participants to have reasonable expectations with respect to the interactions they engage in order to achieve their objectives. Moreover given that we target systems as e-marketplaces, taking in account the sensitive nature of their activities, the architecture prescribes the realization of a neutral *third-party component* in charge of analyzing the participants' interactions (by using the information received by the Bridging Layer as described in Section 5), with the aim of monitoring the evolution of the state of the interaction and specifying and enforcing the norms of the regulating context.

In order to realize all these functionalities we introduce in the proposed architecture the *Regulation Layer*. It is based on the OCeAN meta-model [11], in which regulating contexts are defined as *artificial institutions* that provide a high-level representation of a specific set of institutional actions together with the norms that govern them, and of the institutional objects that need to be observed to monitor the evolution of the state of the interactions. For every specific application, such institutions are operationalized by grounding them in the current domain [12, 7].

The Regulation Layer must possess a formal representation of the state of the interaction suitable to carry out automatic reasoning. In particular this representation has to include specifications of: (i), the regulating context in force;

(ii), the types of events and actions the application is dealing with; (iii), the application-dependent and application-independent knowledge defining the relevant objects and their states during the interaction; and (iv), the instances of the institutional actions and events that happen in the system. Reasoning will then allow the system to monitor the evolution of the state of the interaction, detecting in particular norms fulfillment and violation.

Our infrastructure meets these requirements in the following way. We define our regulating context as an OCeAN artificial institution. The first regulating context we have operationalized so far is the *Commitment Institution*, which regulates agent interactions in terms of the commitments they make to each other by the performance of communicative acts [6, 7]. This is an application-independent foundational institution, used in the definition of more specific application-dependent institutions (like for example the institutions formalizing different types of auctions). This Commitment Institution specifies commitments as institutional objects, together with their life-cycle rules and the institutional actions that allow an agent to create, cancel, or otherwise manipulate them. This enables us to monitor the state of an interaction in term of the evolution of the commitments that the participants make to each other. Application-dependent regulating contexts (like for example those relevant to e-commerce) are also represented as OCeAN institutions.

In our infrastructure, institutions as well as domain knowledge (e.g., knowledge about the products that are exchanged in the e-market) are represented as ontologies specified in OWL 2 DL [14], the standard language for defining ontologies in the Semantic Web. Also the state of an interaction is represented in an OWL ontology, that we call the *Interaction Ontology*, which is continually updated while the interaction proceeds (see Section 5). More precisely the *Interaction Ontology* contains a representation of the institutional objects defined by the institutions in force, along with when required, the institutional actions that create and manipulate them, altogether with the basic actions and objects the institution may operate with. To serve this purpose, the *Interaction Ontology* imports:

- an OWL upper ontology specifying common application-independent concepts like the notion of agent, action, event, and object;
- the *SWRL Temporal Ontology*[10] for representing instants and intervals of time;
- the OWL ontologies used for representing the relevant artificial institutions e.g. the Commitment Institution Ontology[11];
- the *Domain Ontology* used for representing relevant domain knowledge.

Some of these ontologies are described in details in [10]. The ontology imports are realized according to an architecture [10] that we have crafted specifically to avoid conflicts and duplications of those application-independent concepts (like agent, action, temporal interval, etc.) on which several ontologies overlap.

[10] `http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTemporalOntology`
[11] `http://www.people.lu.unisi.ch/okouyad/CommitmentOntology.owl`

Using OWL 2 DL reasoning, our representation makes it possible to monitor the state of the interactions according to the rules of the context. Thus, equipped with it, in compliance with the prescriptions of the architecture which require a neutral third-party component to enact this functionality, our infrastructure provides for a regulation component which plays the role of interaction manager, in charge of monitoring regulations and requesting their enforcement when necessary. To this purpose, the regulation component relies on the Pellet OWL 2 reasoner[12] that it uses in conjunction with the OWL-API[13]: every time a relevant event happens (such as the elapsing of a pertinent instant of time, or the realization of an institutional or non-institutional action or events), a suitable assertion is added to the ABox of the *Interaction Ontology* and the reasoning process is triggered.

Implementing such a task by OWL 2 DL reasoning is not straightforward. First, as participants interactions have to be monitored over time, it is necessary to carry out some kind of temporal reasoning. For instance, if a participant has a commitment towards another to realize a given action before a deadline, in order to deduce that the after the deadline the commitment is fulfilled or violated it is necessary to deduce that the deadline has elapsed. This cannot be specified by OWL axioms alone; therefore, SWRL[14] rules containing temporal built-ins have been added to perform suitable temporal inferences. Such rules exploit the Time Ontology developed by the Protege group [16], which provides a time representation format that is suitable for calculation, is aligned with the current XSD standards, and defines a rich set of temporal builts-ins that can be used to extend our OWL ontologies with SWRL rules. However, given that these built-ins are not SWRL standards, they are not natively supported by reasoning engines; as the Protege group has provided an implementation for reasoning with these built-ins only with the Jess rule engine, we have developed our implementation for extending the reasoning capabilities of Pellet reasoner by using the Pellet custom built-ins definition mechanism.

Representing the evolution of the state of interactions (including for example the new commitments that the participants bring about) by means of a continuous update of the *Interaction Ontology* at run-time [8], is a delicate task because it may introduce inconsistencies. More specificaly, in our formalization of the *Commitment Institution ontology*[11] presented in [6], in which we refer to it as the *Obligation ontology*, we specify that an action-commitment (i.e., a commitment to perform an action, namely, an obligation), has an associated temporal interval, within which the action must be executed. Determining this interval can involve several steps depending on the properties inherited by the the commitment at its creation. In certain situations such as when the action-commitment is conditional, it only becomes activated if a specific triggering event or action takes place; when this activation occurs, the beginning and the end instant of time of the interval associated to the action-commitment have

[12] `http://clarkparsia.com/pellet/`

[13] `http://owlapi.sourceforge.net/`

[14] `http://www.w3.org/Submission/SWRL/`

to be set. For example, if the exchange of a message commits a participant to deliver a product within two days, on condition that the receiver of the product performs a payment, then the action-commitment will be created as soon as the message is exchanged, but will only be activated when the payment takes place. At activation time the interval will be determined as follows: (i), its beginning is set at the time instant of the activation; and (ii), its end is set at the beginning plus two days. All this can be expressed by a suitable SWRL rule. However, if several actions belonging to the activation class of the obligation take place, the SWRL rule will be activated several times and the interval of the obligation will be represented incorrectly. This problem cannot be solved inside the OWL ontology, even by the use of additional SWRL rules; therefore we regulate the activation of the relevant SWRL rule with an external Java program that using the OWL-API, checks that an interval that is already set is not further changed. In short, some reasoning and calculations have to be made outside of the reasoner, in order to properly manage the Interaction Ontology.

5 The Bridging Layer

To regulate interactions it is necessary to capture the participants' actions and other relevant events that take place in the system, and to represent them in a form that suits the abstraction level at which regulation operates. This is the purpose of the Bridging Layer. For this, it prescribes a bridging component which equipped with the definition of the institutions in force that are shared with the regulation component, operates as detailed in the following.

First, all events (inclusive of the participants' actions) that are relevant for regulation must be observed by the Bridging component. These events take place either at the Messaging Layer or at the Core Service Layer. As far as the former is concerned, the relevant events consist in exchanges of ACL messages, which are made available for observation by the CC (Communication Channel) component of the Messaging Layer. To the purpose of regulation, it is therefore crucial that all message exchanges between participants take place through the CC provided by the infrastructure. As we have already remarked, however, message exchanges are not the only events that need regulation. Among these also certain non-communicative events are included, like for example the actions of payment or delivery of products. These events are made available by the Core Service Layer.

Subsequently, the observed events have to be represented in a form that is suitable for regulation. In particular, given that the Regulation Layer relies on artificial institutions, representing an observed concrete event in a form suitable for regulation involves producing a representation that is compatible with the specification of the artificial institution.

In the OCeAN metamodel, artificial institutions deal with two types of events, that we respectively call *basic* and *institutional events*. An institutional event Y is an event that is brought about by the performance of another, lower level event X, thanks to suitable *counts-as* rules, provided that certain enabling conditions C hold. For example, an artificial institution may specify that a certain type of message sent by a suitably empowered agent A will count as an institutional action of opening an auction. Contrastingly, basic events are events that can be directly produced by a participant, without the need of realizing it through the performance of another, lower level event. For example, performing the concrete action of sending a message to another participant is represented in the institution as a basic event of message exchange.

Therefore, transforming an observed concrete event in a form suitable for regulation requires producing a representation of either a basic or institutional event. In the Regulation Layer, both artificial institutions and the concrete domains over which they operate are specified as OWL ontologies. Thus the infrastructure transforms the observed concrete event into OWL individuals that belong to classes of events pertaining either to the institution ontologies or to the concrete domain ontologies. More accurately, as institutional events are grounded on basic events, this transformation process consists of: (i), creating an OWL individual representing the basic event; and (ii), optionally creating an OWL individual representing the institutional event, if this is required by a count-as relationship defined in the institution in force.

Coherently with the OCeAN metamodel, we provide a set of application-independent *counts-as* links between message exchanges (considered as basic events) and the creation of suitable commitments (considered as institutional events): these rules are part of the Commitment Institutions and specify the application-independent component of the semantics of OCeAN-ACL. More specifically, according to the OCeAN-ACL semantics, the exchange of commissive messages (like promising) and directive messages (like requesting) are interpreted in the Commitment Institution as institutional actions that create *action commitments* [20], that is, commitments to perform the action described in the content part of the message. Commitments of this type can be considered as equivalent to *obligations*; for example, if agent A *promises* to agent B to pay a given sum of money M for a given product P, the communicative act will be interpreted as a create-obligation institutional action, that is, an attempt to create an obligation of agent A to pay M euros to B for product P. When the Bridging Layer delivers this institutional action to the Regulation Layer, the Interaction Ontology will be updated with a new institutional object of type Obligation, with A as the debtor, B as the creditor, and the payment of M euros for P as the content. Thereafter, the obligation will be monitored for its fulfillment, violation or cancellation as part of the process of interaction monitoring carried out by the Regulation Layer. Requests are treated in a similar way, except that they involve one more step; more precisely, a request is interpreted as the attempt to create an *action precommitment* (or *preobligation*), which in turn leads to an attempt to create an obligation for the receiver, if the receiver accepts the request (i.e., the preobligation).

Assertive communicative acts (like *informing*) are conceptually different from commissives and directives, because they introduce *propositional commitments*[20], which cannot be interpreted as ordinary obligations. For example, if agent A informs agent B that the product delivered is damaged, this commits A to the truth of what is said (i.e., that the product is indeed damaged), but does not obligate A to perform any predefined action. We have not yet worked out a representation of propositional commitments for our infrastructure: this issue is therefore deferred to future works.

Finally, there is another type of communicative acts, which following the terminology of Searle's Speech Act Theory [19] we call *declarations*; examples are declaring that an auction is open, or that a specific agent is the winner of an auction's run. Declarations are carried out by exchanging suitable ACL messages, with declaration as the performative, and a content that represents the institutional action being performed. Coherently with the OCeAN metamodel, such messages are interpreted within an artificial institution through a *counts-as* rule, which generate the declared institutional action provided that certain conditions hold. Typically, a condition for the successful performance of a declaration is that the actor of the action has the *institutional power* to perform the declared institutional action (e.g., only an auctioneer can possibly open an auction). Such institutional powers are associated at design time to the different roles that can be played by a participant in an institution, and are checked at runtime by the Regulation Layer.

In practice, to achieve this transformation from basic events to institutional events, the OWL specifications of application-independent concepts (such as agent, action, event, object, time instant, time interval, etc.) are shared between the Content Language Ontology (see Section 2), the ontologies of the relevant institutions, and the domain ontologies over which the ongoing application operates and on which the institutions are grounded. The sharing is achieved thanks to the ontological architecture introduced in the Regulation Layer, which eliminates all the ontological mapping hurdles that would have otherwise been necessary to handle for the full transformation process to take place. Indeed it allows to seemingly go from one representation to another; for instance, going from the communicative action *promise (A, B, pay (book1, 5 euro))* (which involves the Content Language Ontology and a concrete domain ontology) to the institutional action *create-Obligation (A, B, pay (A, B, book, 5 euros), instant1)* (which involves the Commitment Institution Ontology and the same domain ontology) is achieved smoothly thanks to the underlying shared concepts of agent, action, object. If these concepts were not shared appropriately, mappings would have been necessary between the specifications of these concepts in different ontologies. The same principle applies, for example, when a non-communication action of payment happens that is represented by the OWL individual *Pay(A, B, book, 5 euros, inst1)*, which has to be transformed into the institutional action *Acquire-Ownership(A, B, book, 5 euros, instant1)* of an hypothetical Ownership Institution (where the target representation is understood as *A* getting the ownership of *book* from *B*, for the price of *5* euros at *instant1*).

6 Related Work

Among the recent multiagent infrastructures focused on OISs, which in particular share the aim of providing the regulation of the participants' interactions in the form of a neutral third-party functionality, as part of the overall support that they deliver, the Magentix2 Open multi-agent systems platform[15][3] represents the state of the art on the matter. In particular it is the most advanced operational infrastructure, which includes many of the recent advances in the OIS area. We therefore provide a comparison with our infrastructure as a way to relate our work to the state of the art in the field.

At a very abstract level the two infrastructures share the same architectural approach. More precisely, although their respective concrete layered architecture are slightly differently structured, they present the same abstract architectural organization: a top part concerned with regulation specification and management, a bottom part concerned with the support of observable interactions between heterogenous participants, and a middle part concerned with the monitoring of the participants' interactions according to the regulation in force and its enforcement when deemed appropriate. Consequently, differences only appears in the way the parts are concretely realized, with the most fundamental of them occurring in the middle part. This reflects a common vision of the role of the infrastructure, but divergences on how its different parts may concretely operate to achieve it.

More specifically, at the top level, Magentix2 adopts the metamodel of virtual organizations, which specifies roles with norms including platform generic roles such as OMS (Organization Management System) and DF (Directory Facilitator), for the specification of a regulation structure. Our infrastructure also defines a regulation structure at this level, but one that is based on the OCeAN metamodel of artificial institutions (see Section 4). While a thorough comparison of the two metamodels is outside the scope of this paper, it can be safely said that both infrastructure intend to provide similar regulating structures, which in particular are centered on non-regimented norms, to harness the participants' activities.

At the bottom, both infrastructures provide an observable vehicle for the participants to interact with each other. To that end, they use similar approaches, but differ in the general understanding of interactions. Indeed the OCeAN metamodel classifies actions into communicative and non-communicative ones, which Magentix2 does not, in that it only considers communicative actions. Consequently, while we divide the bottom part of the infrastructure into two layers (Messaging and Core Service), with the upper one devoted to non-communicative actions and the lower one devoted to communicative actions, Magentix2 only provides one interaction level which corresponds to our lower layer.

As far as communicative interactions are concerned, the two infrastructures operate in a similar manner (as they both provide an end point neutral messaging protocol with a broker for interoperable communication between het-

[15] `http://www.gti-ia.upv.es/sma/tools/magentix2/`

erogenous participants), but diverge in the choice of the technology. Where we use WS (SOAP, HTTP, WSDL) with the SOAP Body structure defined as an OCeAN-ACL message for messages exchange, Magentix2 adopts AMQP[16] with the message body structure defined as a FIPA-ACL message. We believe that the use of WS is more widespread and therefore easier to adopt than AMQP, which has yet to become a standard.

As previously mentioned, the sharpest differences between the infrastructures occurs in the middle part, whose functionality can be summarized as follows: (i), observing concrete events such as message exchanges or core-service events; (ii), representing observed events in a form suitable for regulation; (iii), checking them against the regulations for monitoring purposes; and (iv), enforcing the relevant regulations when deemed appropriate. It is with (ii) and (iii) that the two infrastructures differ substantially.

With our infrastructure, checking against regulations is done by means of reasoning over a representation of the state of the interaction, carried out within an OWL ontology that includes the institutions in force and the norms coming along. Our norms and their instantiations (in terms of obligations and prohibitions) are represented as OWL individuals, so that their activation, cancellation, fulfillment and violation conditions are represented as event types (i.e., as subclasses of class Event). Therefore we use the full power of DL reasoning to match the representations of concrete events with norms conditions. This process is much more powerful than the one adopted by Magentix2, which relies on the matching of a restricted subset of first-order logic formulas.

A further important difference between Magentix2 and our infrastructure is that the latter does not rely on an application-independent semantics of ACL messages. In our infrastructure, based on the OCeAN metamodel, the application-independent part of messages (i.e., all components of an ACL message with the exception of its content) is given a uniform semantics across applications. Moreover, such semantics allows for a representation of messages (produced by the Bridging Layer) that immediately relates message exchanges to the Regulation Layer. This means that only application-dependent non-communicative events will need to receive a special treatment in different applications of the infrastructure. Conversely, Magentix2 does not provide for any application independent connection between the participants' actions and regulation, thus making the conversion to different application more expensive and error-prone.

7 Conclusions

In this paper we have presented an infrastructure for Open Interaction Systems, based on the OCeAN metamodel and currently under implementation. Our main concerns in the development of the infrastructure are, on the one hand, to guarantee openness and interoperability, and, on the other hand, to rely as much as possible on technologies that are sufficiently mature and stable, like Service Oriented and Semantic Web technologies, to facilitate adoption by the industry.

[16] `http://www.amqp.org/`

The infrastructure has been divided into components to separate different concerns, which brings several advantages: on the one side, it enables us to distribute the infrastructure and to use techniques of dynamic adaptation (such as cloning and self-deletion) to manage overhead issues; on the other side it enables us to provide targeted upgrades and developments of the infrastructure. So far, for prototyping purposes the infrastructure is being implemented as a monolithic multi-threaded Java application. Nevertheless, the different components are present and well separated so that they can be easily extracted to provide a fully distributed infrastructure.

In the near future we intend to complete the implementation and test of the prototype. In particular we plan to complete the formalization in OWL of the semantics of the various type of communicative acts, to separate the various component of the prototype and to test it with the formalization and execution of an e-marketplace, inclusive of the OWL ontologies representing the relevant institutions and domain knowledge.

References

1. N. Balani and R. Hathi. *Apache CXF Web Service Development*. Packt Publishing, 2009.
2. L. Chiarabini. CORBA vs. Web Services. `http://www.itu.dk/~oladjones/mastersthesis/materialsfromportals/corbaversuswebservices.pdf`, May 2004. (accessed March 14, 2013).
3. N. Criado, E. Argente, P. Noriega, and V. Botti. MaNEA: A Distributed Architecture for Enforcing Norms in Open MAS. *Engineering Applications of Artificial Intelligence*, 26(1):76–95, 2012.
4. T. Erl. *Service-Oriented Architecture (SOA): Concepts, Technology, and Design*. Prentice Hall, Aug. 2005.
5. T. Erl. *SOA Principles of Service Design (The Prentice Hall Service-Oriented Computing Series from Thomas Erl)*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2007.
6. N. Fornara. Specifying and Monitoring Obligations in Open Multiagent Systems Using Semantic Web Technology. In A. Elçi, M. Koné, and M. Orgun, editors, *Semantic Agent Systems*, volume 344 of *Studies in Computational Intelligence*, pages 25–45. Springer Berlin / Heidelberg, 2011.
7. 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.
8. N. Fornara and M. Colombetti. Representation and monitoring of commitments and norms using OWL. *AI Communications - European Workshop on Multi-Agent Systems (EUMAS) 2009*, 23(4):341–356, 2010.
9. N. Fornara, D. Okouya, and M. Colombetti. A Framework of Open Interactions based on Web Services and Semantic Web Technologies. In *Proceedings of the 9th European Workshop on Multi-Agent Systems EUMAS 2011*, 2011.
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*, volume 7541 of *LNCS*, pages 97–113, 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. N. Fornara, F. Viganò, M. Verdicchio, and M. Colombetti. Artificial institutions: a model of institutional reality for open multiagent systems. *Artif. Intell. Law*, 16(1):89–105, Mar. 2008.
13. M. Hapner, R. Burridge, R. Sharma, J. Fialli, and K. Stout. Java Message Service Specification Version 1.1. *Sun Microsystems, Inc.*, April 2002.
14. P. Hitzler, M. Krötzsch, and S. Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009.
15. T. K. Kent. *Developing Web Services with Apache CXF and Axis2*. Lulu.com, 3rd edition, 2010.
16. M. J. O'Connor and A. K. Das. A Method for Representing and Querying Temporal Information in OWL. In A. Fred, J. Filipe, and H. Gamboa, editors, *Biomedical Engineering Systems and Technologies*, volume 127 of *Communications in Computer and Information Science*, pages 97–110. Springer Berlin Heidelberg, 2011.
17. OMG. The Common Object Request Broker: Architecture and Specification. *The Object Management Group*, pages 1–712, Nov. 1999.
18. C. Scordino. How Web Services relate to the well established CORBA Middleware. `http://retis.sssup.it/~scordino/documents/corba.pdf`, April 2004. (accessed March 14, 2013).
19. J. R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, United Kingdom, 1969.
20. D. N. Walton and E. C. Krabbe. *Commitment in Dialogue: Basic concept of interpersonal reasoning*. State University of New York Press, Albany NY, 1995.
21. S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. F. Ferguson. *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.