IRIS AperTO

UNIVERSITÀ DEGLI STUDI DI TORINO

AperTO - Archivio Istituzionale Open Access dell'Università di Torino

2COMM: A Commitment-Based MAS Architecture

This is the author's manuscript

Original Citation:

Availability:

This version is available http://hdl.handle.net/2318/145576 since 2016-06-27T23:12:16Z

Publisher:

Springer

Published version:

DOI:10.1007/978-3-642-45343-4_3

Terms of use:

Open Access

Anyone can freely access the full text of works made available as "Open Access". Works made available under a Creative Commons license can be used according to the terms and conditions of said license. Use of all other works requires consent of the right holder (author or publisher) if not exempted from copyright protection by the applicable law.

(Article begins on next page)

24 April 2024

IRIS AperTO

UNIVERSITÀ DEGLI STUDI DI TORINO

This is the author's final version of the contribution published as:

Matteo Baldoni, Cristina Baroglio, and Federico Capuzzimati. 2COMM: a commitment-based MAS architecture. In M. Cossentino, A. El Fallah Seghrouchni, and M. Winikoff, editors, Proc. of the 1st International Workshop on Engineering Multi-Agent Systems, EMAS 2013, held in conjuction with AAMAS 2013, pages 17-32, St. Paul, Minnesota, USA, May 2013. ISBN: 978-3-642-45342-7, DOI: 10.1007/978-3-642-45343-4_3

The publisher's version is available at:
http://dx.doi.org/10.1007/978-3-642-45343-4_3

When citing, please refer to the published version.

Link to this full text:
http://hdl.handle.net/2318/145576

This full text was downloaded from iris-AperTO: https://iris.unito.it/

iris-AperTO

University of Turin's Institutional Research Information System and Open Access Institutional Repository

2COMM: A commitment-based MAS architecture

Matteo Baldoni, Cristina Baroglio, Federico Capuzzimati

Università degli Studi di Torino — Dipartimento di Informatica
c.so Svizzera 185, I-10149 Torino (Italy)
{matteo.baldoni,cristina.baroglio,federico.capuzzimati}@unito.it

Abstract. Social expectations and social dependencies are a key characteristic of interaction, which should be explicitly accounted for by the agent platform, supporting the coordination of the involved autonomous peers. To this aim, it is necessary to provide a normative characterization of coordination and give a social meaning to the agents' actions. We focus on one of the best-known agent platforms, Jade, and show that it is possible to account for the social layer of interaction by exploiting commitment-based protocols, by modifying the Jade Methodology so as to include the new features in a seamless way, and by relying on the notion of artifact, along the direction outlined in the Mercurio proposal.

Keywords: Commitment-based Interaction Protocols, Agents & Artifacts Model, JADE, JADE Methodology, Agent-Oriented Software Engineering

1 Introduction and Motivation

Interaction creates *social expectations* and *dependencies* in the involved partners [38, 18, 34, 23]. These should be explicitly accounted for by the *agent platform* to allow the coordination of autonomous entities. In order to create social expectations on the agents' behavior, it is necessary to introduce a *normative* characterization of coordination and give a social meaning to their actions. An agent that understands such a specification and that publicly accepts it (i.e. that declares it will behave according to it) allows reasoning about its behavior [21]. This is the key to the development of open environment systems, made of autonomous and heterogeneous components. By not supplying such abstractions, current platforms do not supply agents the means for *observing* or *reasoning* about such meanings of interaction, and do not supply the designers the means to explicitly *express* and *characterize* them when developing an interaction model.

One prominent example is JADE [10, 11], which is a well-established development environment for multi-agent systems, FIPA-compliant and actually used for industrial applications, and which notoriously does not provide of a social and observational semantics. One of the aims of the Mercurio framework [4, 3] is to fill this gap by introducing in JADE the means for exploiting *commitments* and *commitment-based protocols*, which are well-known for featuring the social

and observational semantics [34, 35, 41], JADE lacks of. Our starting point for introducing commitment-based protocols inside JADE is the JADE Methodology [30]. This methodology is particularly interesting because it is intrinsically agent-oriented and it is not the adaptation of an object-oriented methodology, and it combines a top-down approach with a bottom-up one, possibly allowing the integration with legacy, non agent-based systems. It concerns two of the four main phases of the standard software development cycle: the *analysis phase* and the *design phase*.

Following [4], we rely on a form of indirect communication among agents that envisages the use of *artifacts*: commitment-based communication artifacts implement interaction protocols as well as monitoring functionalities for the verification that the on-going interaction respects the protocol, for detecting violations and violators, and so forth. Artifacts, therefore, encode the social layer of the multi-agent system: as a programmable communication channel an artifact contains what in the terminology of commitment protocols is called "the social state", and captures it as an interaction session among the parties. Artifacts also supply agents the social actions that are necessary to the interaction – that is, actions that allow agents to enter into and to comply with commitments – together with their social meaning and, as a consequence, they capture the coordination rules of the protocol. The reification of commitment protocols allows agents to act on them, e.g. to examine them (for instance, to decide whether to play one of the foreseen roles), use them (which entails that they explicitly accept the corresponding regulation), negotiate their construction, specialize them, and compose them. The advantage of relying on indirect communication is that it allows more variegated ways of interacting, not hindering message exchange when necessary.

In this paper we show that our proposal can be integrated *seamlessly* within the JADE Methodology, simply by substituting the selection of JADE FIPA protocols with the selection/construction of appropriate communication artifacts. We also use the methodology to show the differences between these two alternatives with the help of an example from a financial setting.

Section 2 reports the relevant background, necessary to understand the proposal. Section 3 is the core of the paper, containing the original proposal. Section 4 applies the concepts to an illustrative example, from a financial setting. A discussion also involving related works ends the paper.

2 Background

We briefly report the technical, methodological and theoretical background required for our work. We use the proposal in [4] as a high-level reference architecture. In this work, the authors outline the basic ideas for an interaction-oriented agent framework, grounding the social semantics of interaction on commitments, and proposing the A&A (Agents and Artifacts) Metamodel as a means to obtain a form of indirect, observable communication. Let us, then, explain the fundamental bricks to build our architecture, whose overview is reported in Figure 1.

JADE framework. JADE is a popular and industry adopted agent framework. It offers to developers a Java middleware 100% FIPA-compliant (Foundation for Intelligent Physical Agents, [1]) plus a set of command-line and graphical tools, supporting development and debugging/testing activities. Its robustness and well-proven reliability makes JADE a preferred choice in developing MAS. It is currently used in many research and industrial projects jointly with its most popular and promising extension, WADE [17]. A JADE-based system is composed of one or more *containers*, each grouping a set of agents in a logical *node* and representing a single JADE runtime. The overall set of containers is called a *platform*, and can spread across various physical hosts. The resulting architecture hides the underlying layer, allowing support for different low-level frameworks (JEE, JSE, JME, etc.). The platform reference container is called *main container*, and represents the entry point to the system. JADE provides communication and infrastructure services, allowing agents, deployed in different containers, to discover and interact with each other, in a transparent way from the developer's logical point of view.

Commitment Protocols. Agents share a social state that contains commitments and other literals that are relevant to their interaction. A *commitment* $C(x, y, r, p)$ denotes a contractual relationship between a debtor x and a creditor y: x commits to y to bring about the consequent condition p when the antecedent condition r holds. A commitment, when active, functions as a directed obligation from a debtor to a creditor. However, unlike a traditional obligation, a commitment may be manipulated, e.g., delegated, assigned, or released [37]. Importantly, commitments have a regulative value: the social expectation is that agents respect the commitments which involve them and, in particular, the debtor is considered responsible of realizing the consequent condition. Thus, the agents' behavior is affected by the commitments that are present in the social state. A *commitment protocol* usually consists of a set of actions, whose semantics is shared (and agreed upon) by all of the interacting agents [41, 40, 20]. The semantics of the social actions is given in terms of operations which modify the social state by, e.g., adding a new commitment, releasing another agent from some commitment, satisfying a commitment, see [41].

CArtAgO. CArtAgO is a framework based on the A&A model. It extends the agent programming paradigm with the first-class entity of *artifact*: a resource that an agent can use, and that models working environments ([32]). In order to properly model a MAS, CArtAgO proposes to explicitly model the environment where pro-active agents live, work, act and communicate. It provides a way to define and organize *workspaces*, logical groups of artifacts, that can be joined by agents at runtime and where agents can create, use, share and compose artifacts to support individual and collective, cooperative or antagonistic activities. The environment is itself programmable as a dynamic first class abstraction, it is an active part of a MAS, encapsulating services and functionalities. The A&A model decouples the notion of agent from the notion of environment. The overall engineering of the MAS results more flexible, easy to understand, modular and

reusable. CArtAgO provides an API to program *artifacts* that agents can use, regardless of the agent programming language or the agent framework used. This is possible by means of the *agent body* metaphor: CArtAgO provides a native agent entity, which allows using the framework as a complete MAS platform as well as it allows mapping the agents of some platform onto the CArtAgO agents, which, in this way, becomes a kind of "proxy" in the artifacts workspace. The developed agent is the mind, that uses the CArtAgO agent as a body, interacting with artifacts and sensing the environment. An agent interacts with an artifact by means of public *operations*. An operation can be equipped with a *guard*: a condition that must hold so that the operation will produce its effects. It is not an execution condition: when the guard does not hold the action is performed anyhow but without consequences.

3 Reifying Commitment Protocols with Artifacts

Artifacts naturally lend themselves to provide a suitable means for realizing mediated communication channels among agents. To this aim, it is necessary to encode inside the communication artifacts a normative characterization to the actions it offers to agents and that allow them to interact. We propose to interpret commitment protocols as environments, within which agents interact. The public interface of artifacts allows agents to examine the encoded interaction protocol. As a consequence, the act of using an artifact can be interpreted as a declaration of acceptance of the coordination rules. This will generate social expectations about the agent's behavior and agrees with the characterization of norms in [21]. Moreover, the fact that the behavior of agents on artifacts is observable and that interactions only occur through artifacts, agrees with the view that regulations can only concern observable behavior [22]. The resulting programmable environment provides a flexible communication channel that is suitable for realizing open systems. Notice that the use of a programming environment does not mean that the social state will necessarily be centralized: an artifact can be composed by a distributed network of artifacts.

Figure 1 sketches the way in which we propose to use CArtAgO so as to account also for social commitments inside JADE. We named this first realization of the Mercurio architecture *2COMM* (standing for "Communication & Commitment"). 2COMM realizes mediated interaction by means of communication artifacts, which, in our proposal, replace the JADE-based FIPA protocols and which reify commitment-based protocols [4]. At the bottom level, the JADE framework supplies standard agent services: message passing, distributed containers, naming and yellow pages services, agent mobility. When needed, an agent can *enact* a certain protocol role, thus using a communication artifact by CArtAgO. This provides a set of operations by means of which agents participate in a mediated interaction session. Each artifact (protocol enactment) maintains a *social state*, that is, a collection of *social facts* and *commitments* involving the roles of the corresponding protocol, following Yolum and Singh's commitment protocol model [40].

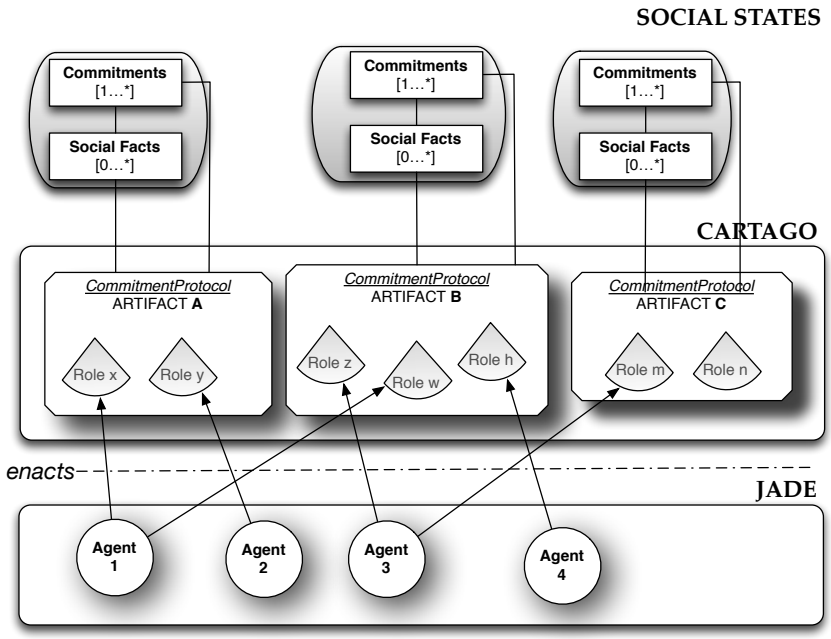

Fig. 1. A sketch of 2COMM.

3.1 Communication Artifact

We follow the ontological model for organizational roles proposed in [13, 14], which is characterized by three aspects: (1) *Foundation*: a role must always be associated with an institution it belongs to and with its player; (2) *Definitional dependence*: the definition of the role must be given inside the definition of the institution it belongs to; (3) *Institutional empowerment*: the actions defined for the role in the definition of the institution have access to the state of the institution and of the other roles, thus, they are called *powers*; instead, the actions that a player must offer for playing a role are called *requirements*.

Communication artifacts realize a kind of mediated interaction that is guided by commitment-based protocols. Figure 2 shows the UML schema of the super-type of communication artifacts implementing specific interaction protocols (e.g., Contract Net, Net Bill, Brokering): the *CommitmentCommunicationArtifact*. We call an instance of an artifact of type CommitmentCommunicationArtifact an *interaction session*. It represents an on-going protocol interaction, with a specific social state that is observable by the interacting agents, that play the protocol roles. The CommitmentCommunicationArtifact presents an observable property, *enactedRoles*, that is the collection of the roles of the protocol (definitional dependence [13, 14]). Actions have a social effect only when they are executed by

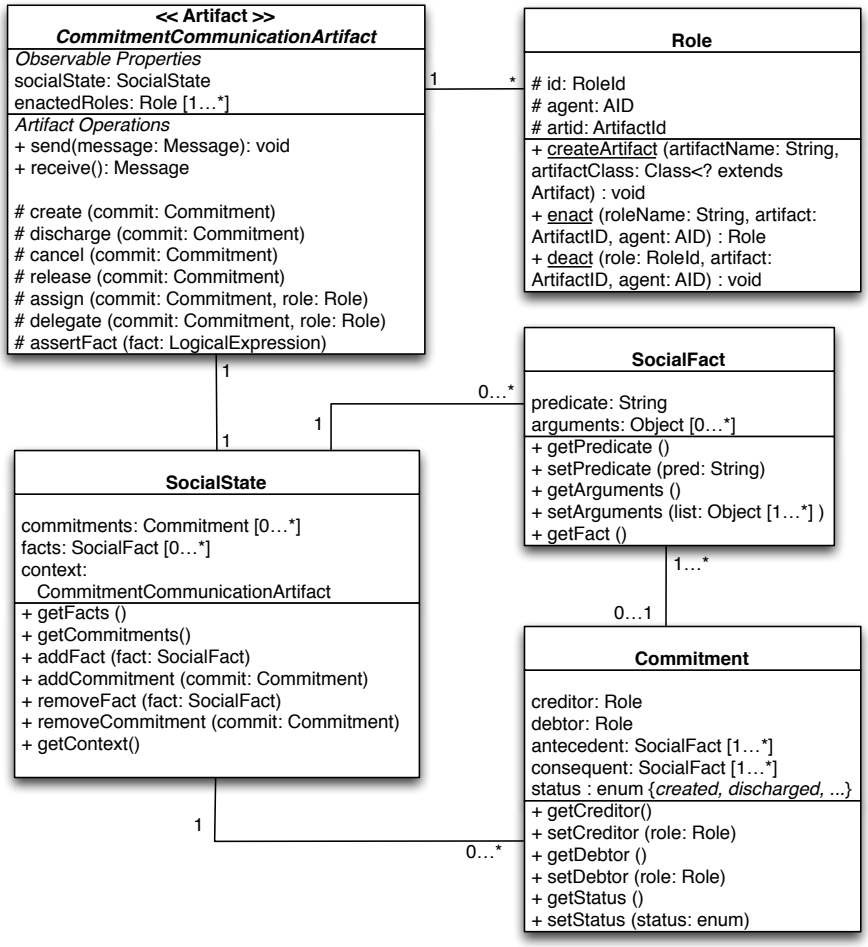

Fig. 2. The UML Class diagram for the core of 2COMM.

the role they are assigned to, but actions are not defined at this super level, rather they are provided by the instantiations of the CommitmentCommunicationArtifact, i.e. by artifacts implementing specific protocols. Each protocol action is implemented as a public *operation*, which is associated to a role by means of an *operation guard* (institutional empowerment [13, 14]): the guard checks who is performing the operation; if the agent is not the one playing the right role, the action simply has no effect, otherwise, the fact that the action was executed is registered in the social state together with its meaning. An action can have some additional guards, implementing *context preconditions*: this condition specifies

the context in which it makes sense that the action produces the described social effect. An artifact can be monitored by an observer agent, that, following the CArtAgO terminology, is *focusing* on that artifact, particularly on one or more public properties. A change of one of these properties causes a *signal*, from the artifact to the observer agents, about the property that changed: the agents *perceive* the new artifact state. In particular, when the creation of a commitment, involving an agent as a debtor, is signaled to it, this agent is expected to behave so as to satisfy the commitment. The agent is free to decide how (and if) it will handle the satisfaction of its commitments. Therefore, the requirement is that an agent has the capability to behave so as to achieve the involved conditions [13, 14]. An agent who does not show such capabilities is bound to violate its commitments.

CommitmentCommunicationArtifact provides a property, tracking the identity of the agents actually playing the various role. Two operations are provided, by class *Role*, in order to manage the association between an agent's identity and a role: *enact* and *deact*, by means of which an agent can explicitly assume/cease a protocol role (foundation [13, 14]). After enacting a role, the use of the associated operations on the artifact will have social consequences.

The communication artifact has an observable property, *social state*, that is a set of zero or more elements of type *Commitment* or *Social Fact*. As we can see in Figure 2, these structures are simple Java objects, representing the actual social state. The artifact is responsible to manage the Social State structure, i.e. the Commitments life-cycle, as well as the assertion or retraction of social facts, via methods called on commitment and on social fact objects. For Commitment management, we refer to the basic operations of commitment manipulation [40]: *create, discharge, cancel, release, assign, delegate*. The operations regarding the commitments life-cycle are implemented as artifact *internal operations*, therefore, the agents cannot modify commitments explicitly. The communication artifact exposes the social state, whose evolution is controlled by the agents via the protocol-provided actions. Finally, communication artifacts provide service operations, which can be performed only by the ArtifactManager Agent (see below) for managing the protocol roles and the identities of their players.

When the social state property changes, due to the execution of a protocol action (an artifact operation) on the communication artifact, all of the agents using the artifact will be notified, allowing them to react (or not) to the evolution of the interaction. This mechanism is a core part of the CArtAgO framework.

The *ArtifactManager Agent* plays the role of a Yellow Pages Agent for communication artifacts, or, in other terms, of an artifact broker. It has a crucial role: it is a "communication channel" broker, gathering requests for both focused or broadcasting calls for interaction. As such, it provides a collection of utility services. It supplies information about the interaction protocols (e.g. it provides the XML describing a given protocol, it allows a search for a protocol, a list of active communication channels, a list of interacting agents); it answers to requests about the status of an existing interaction session; it notifies the subscriber agents a particular session availability, and so on. Its main purpose is to

prepare the communication artifact among the interacting agents, and to supply it to the requesting agents. It can also enable other interested agents to monitor, audit, or, more generally, observe the social state evolution. The communications between the ArtifactManager Agent and the requesting agents is realized via FIPA-ACL messages: when a requester sends a request ACL message to the ArtifactManager Agent, specifying the protocol and the role it wants to enact, the latter will do the following steps:

1. Check if the requested protocol is available;
2. Check if the requested role is foreseen by the protocol;
3. Create/retrieve a communication artifact of the requested type;
4. Set the requested artifact role field to the agent identifier (AID) of the requester;
5. Respond to the requester with the artifact's reference;
6. Possibly inform other interested agents of the availability of the communication artifact.

The initialization procedure is modeled as a simple FIPA Request Interaction Protocol, where the content of messages consists of the communication artifact request parameters. After this phase, the agent can use the *enact* operation to start playing the requested role. The use of an agent does not necessarily imply a centralization of the yellow pages: agents may directly create communication artifacts; yellow pages can be federated.

3.2 Using Mediated Communication at Runtime

In the following, we show a scenario in which a communication artifact is used, to better explain how to leverage the communication artifacts and the ArtifactManager Agent. We adopt the well-known FIPA Contract Net Protocol (CNP), modeling it as a commitment-based protocol and implementing a corresponding artifact. The scenario is depicted in Figure 3.

The JADE infrastructure is extended with the ArtifactManager Agent, that provides a Yellow Pages service for communication artifacts. It can respond to ACL Messages, that encode requests of a new Communication Atifact, either with a *Failure* message or an *Agree* message. In the latter case, it will either prepare a new instance of the requested communication artifact, or it will return an already existing artifact. For instance, suppose that agent *A1* has to assign a task, and agents *A2* and *A3* have the capability of performing it. Suppose that *A2* and *A3* already registered to the ArtifactManager Agent (*ArA* for brevity), and that this has already instantiated a Contract Net Protocol communication artifact (*CNPCA* for brevity). At this time, the (partial) state of *CNPCA* is:

- **Initiator:** null
- **Participants:** {A2.*AID*, A3.*AID*}

where *AID* is the JADE Agent Identifier. *A1*, then, asks *ArA* for a *CNPCA*, following the procedure described before, without specifying a particular participant. *ArA* matches this request with the already prepared *CNPCA*: the match

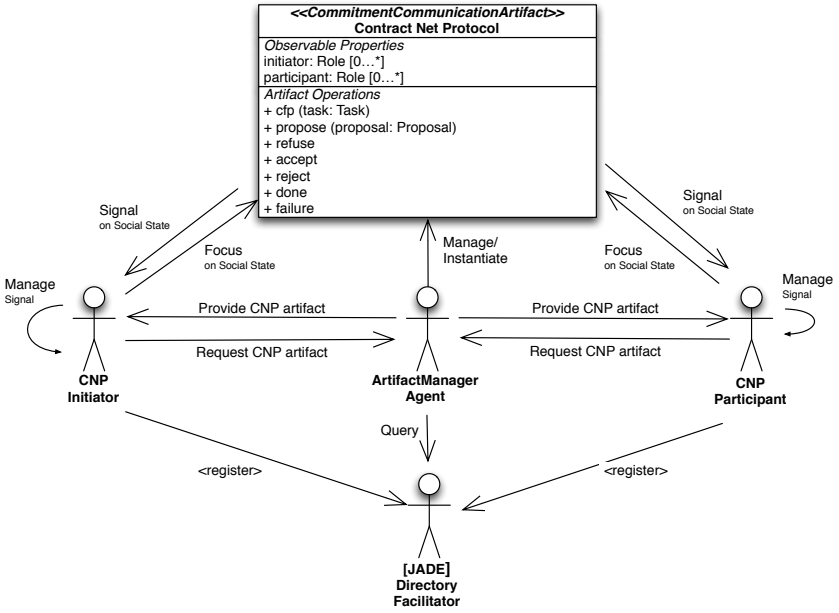

Fig. 3. Possible interactions between the main elements of our proposal, in a CNP example.

is successful, inasmuch the Initiator role is not played by any agent. So, *ArA* stores *A1*.AID in the *Initiator* property of *CNPCA*, and returns its reference to *A1*. Following the CArtAgO terminology, agents *A1, A2* and *A3 focus on* the *SocialState* property of *CNPCA* immediately after having its reference. This means that any change to the social state will be signaled to the three agents, who can take decisions accordingly. The agents interact with one another via operations on *CNPCA*, and observe the social state evolution in order to reason about which actions to take.

An agent can stop playing a protocol role at anytime by executing the *deact* operation. The artifact unregisters its AID from the AID-role mapping list. On the other hand, an agent may enact a partially executed role within an interaction session. What about commitments in such cases? In this work we focused only on the communicational and interaction-related aspects of playing protocol roles: sanctions or other action concerning the institutional (or organizational) levels are not accounted for yet. Simply, since responsibilities are associated to roles, deacting a role yields that the resigning agent will not need anymore to fulfill them, while a substituting agent needs to accept the current commitments of the role it is assuming [40]. A reference model to include, in the future, also institutional aspects could be the JaCaMo proposal [15].

3.3 Using Mediated Communication at Design Time

We assume that MAS designers know a collection of communication artifacts, each representing a commitment-based protocol. Each protocol is enriched with an XML-based description of it, a *Protocol Manual*, available both at design- and at run-time. It is an add-on to the CArtAgO artifact manual, with orthogonal scopes and purposes. It can be used by MAS and agent designers as a guideline for understanding whether an agent is suitable for a protocol role as well as for understanding whether a protocol role suits the purposes of an agent. From a methodological point of view, the designer needs the Protocol Manual to know the social consequences of the actions supplied by an artifact, in terms of social facts and commitments, so he/she can design agent behaviors accordingly. Then, depending on the implemented behavior, the agent will decide how to use information about the social state evolution, how to fulfill commitments, which social action (i.e. a public artifact operation) to execute and when. Ideally, the designer should equip the agent with the behaviors that are necessary to bring about the conditions of the commitments it will possibly take. This protocol-centric design, jointly with the commitment nature of protocols, avoids a critical facet of JADE protocols. Here, a pattern of interaction is projected on a set of JADE behaviors, one for each role, thus making a global view of the protocol and its maintenance difficult, and binding the very interaction to ad-hoc behaviors. Consequently, the risk of conflicting behaviors, not devised at design time, increases. This way, the designer can leverage a library of programmable communication artifacts, focusing on the internal agent behavior without being concerned about ad-hoc shaped communication behaviors.

4 JADE Methodology revised

The JADE Methodology is a JADE founded agent-oriented software engineering methodology. It proposes a fully agent-based approach, instead of adapting Object-Oriented techniques (like MASE [39], Adelfe [12] or MESSAGE [16]). It concerns the *analysis* and the *design* phases of the software development life cycle. The methodology considers agents as "pieces of autonomous code, able to communicate with each other" [30], thus following a weak notion of agency; it does not account for mentalistic/humanistic agents properties.

In the analysis phase, the first step is the identification of *use cases*, i.e. functional requirements of the overall system, which are captured as standard Use-cases UML Diagrams (Figure 4). Starting from this, the designer can point out an initial set of agent types: an agent type for each user/device and for each resource. The agent paradigm foresees that even external devices and software/hardware resources (e.g. legacy systems, databases, external data sources) are represented with an agent. The designer, then, identifies *responsibilities*, i.e. the activities provided by system each agent is responsible for; and *acquaintances*, that is relationships between agents aimed at fulfilling some responsibility. The results are a *Responsibility table* and an *Agent diagram* (Figure 5) with initial acquaintances. No distinction is made between acquaintances and

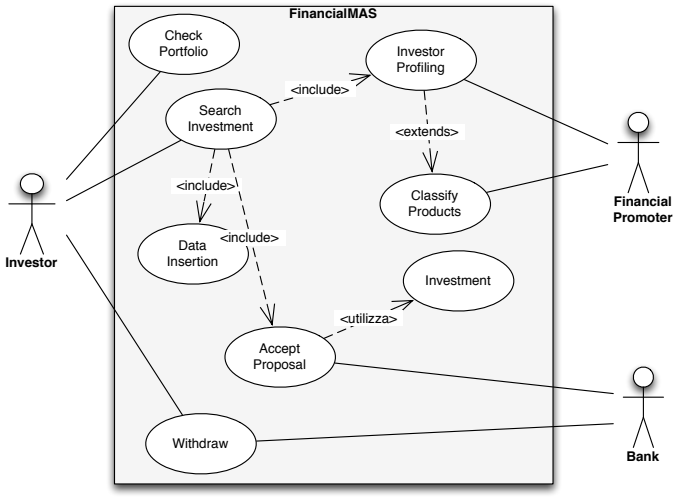

Fig. 4. The FinancialMAS Use Cases.

responsibilities: in fact, the mentioned table will contain both. The analysis is completed by executing activities related to agents/acquaintances refinement, to define discover services and to add management/deployment information. The design phase starts with the *interaction specification* step, where an interaction table is produced. It refers to the responsibility table in order to define interactions between JADE agents, specifying the interacting agents, the protocol and protocol role (e.g. Initiator or Responder), the reference responsibility, and a triggering condition.

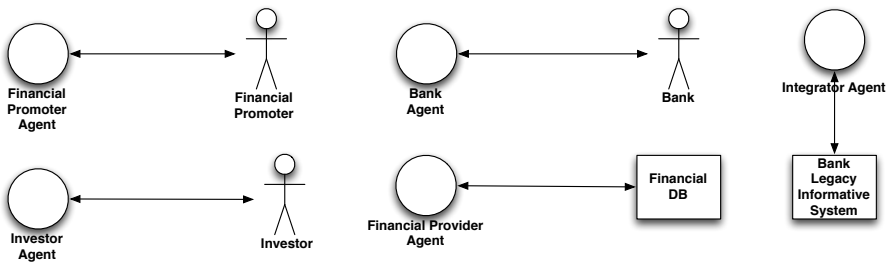

Fig. 5. The FinancialMAS Agent Type Diagram.

It is suggested to use, when possible, standard JADE protocol behaviors, that must be added to an agent's behavior set to implement the corresponding protocol role. The subsequent steps focus on the specification of agent interactions with users and resources; the definition of a yellow page services, using the JADE Directory Facilitator; the implementation of agent behaviors, starting from JADE protocol behaviors related to responsibilities. A last effort is the definition of a shared, system-wide ontology.

Table 1. Responsibility Table for FinancialMAS.

Agent Type	No.	Responsibility
Investor agent (IA)	1	Let investor search for investments proposals
	2	Assist investor in setting search parameters and data
	3	Support the individuation of the investor's risk profile
	4	Support in proposal acceptance
	5	Withdraw from an investment contract
Financial Promoter agent (FP)	1	Respond to investment searches
	2	Assist financial promoter in risk-classifying financial products
	3	Determine the investor's profile
	4	Support individuation of the investor's risk profile
Bank agent (BA)	1	Support bank in investment contract subscription
	2	Assist bank in investment conclusion
Financial Provider agent (FV)	1	Provide financial and aggregate news information
Integration agent (IntA)	1	Serve and support integration with legacy bank informative systems

Table 2. Interaction Table for FinancialMAS: who interacts with whom, to fulfill which duty, by using which protocol.

Interaction	R.ty	Interaction Protocol	Role	With	When
Investor Agent					
Search Investment	1	CNP	Initiator	FP	Investor searches an investment
Profiling	3	Query	Participant	FP	Investor chose a Financial Promoter
Proposal Acceptance	4	Query	Participant	BA	Investor chose a financial product
Withdraw	5	Request	Initiator	BA	After Investor accepted a proposal
Financial Promoter Agent					
Respond to Search	1	CNP	Participant	IA	Investor searches an investment
Profiling	3	Query	Initiator	IA	Investor chose a Financial Promoter
Fin. Prod. Classif.	2	Query	Initiator	FV	FP starts fin. prod. classif.
Bank Agent					
Proposal Acceptance	1	Query	Initiator	IA	Investor chose a financial product
Withdraw	3	Request	Participant	IA	After Investor accepted a proposal
Financial Provider Agent					
Fin. Prod. Classif.	1	Query	Participant	FP	FP starts fin. prod. classif.

We show how it is possible to integrate, within the JADE Methodology [30], an account of commitment-based protocols with the help of a real-world scenario,

we call *FinancialMAS*. For brevity, we show only the fundamental steps needed to draft the system and to highlight the benefits of reifying commitment-based protocols by means of artifacts, and thus based on mediated interaction. By applying the steps of the methodology, we obtained an initial design prototype for FinancialMAS, concerning an initial set of agents and the so called *responsibility table* (Table 1). In the terminology of the JADE Methodology, *responsibilities* amount to functional duties, agents are responsible for, from an overall MAS point of view. To handle them, agents possibly need to *interact* with one another. The result of this analysis is an *Interaction table* (Table 2). At this point, instead of realizing protocols via distributed JADE behaviors, we implement them via *commitment-based communication artifacts*. We assume to have already designed artifacts for common interaction protocols, like the Contract Net Protocol, the Query Protocol, and the Request Protocol. The resulting model is depicted in Figure 6. For the sake of comparison, in Figure 7 we zoomed into the one of the commitment artifacts, the *Contract Net Protocol* artifact, reported as a UML diagram, while in Figure 8 we highlight the very same protocol, implemented via pure JADE behaviors.

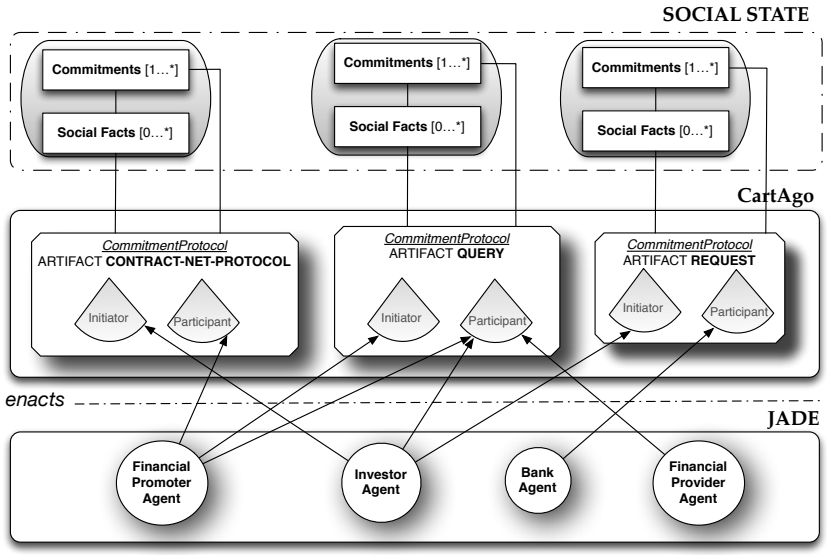

Fig. 6. FinancialMAS Commitment-based Interaction Architecture.

2COMM proposes a clear notion of *Role* that an agent must enact to participate in an interaction session, so the designer must only implement the behaviors for fulfilling the commitments caused by the execution of a protocol *actions*. We

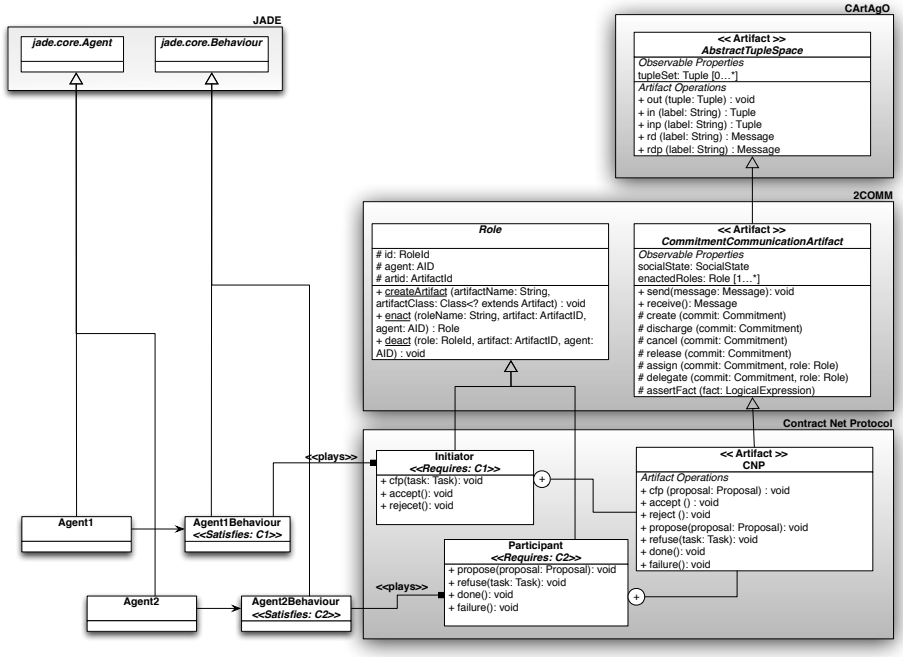

Fig. 7. The UML diagram for the 2COMM implementation of CNP.

refer to the following description of CNP based on commitment protocols (this is just an example, alternatives and variants can be found in papers like [42, 24]):

cfp **means** create(*C(i, p, propose, accept* $\vee$ *reject)*)
accept **means** *none*
reject **means** release(*C(p, i, accept, done* $\vee$ *failure)*)
propose **means** create(*C(p, i, accept, done* $\vee$ *failure)*)
refuse **means** release(*C(i, p, propose, accept* $\vee$ *reject)*)
done **means** *none*
failure **means** *none*

In the case of CNP, two roles are foreseen, *Initiator* (i) and *Participant* (p). Playing a role gives an agent *powers*, in terms of social state modification (i.e. the state of the interaction session) as a consequence of its actions, and the agent designer can use them if, when and how he/she wants. For instance, for what concerns the update of the social state, when an agent playing the role Initiator executes the artifact action *cfp*, the social state is modified by creating the commitment *C(i, p, propose, accept* $\vee$ *reject)*. On the one hand, this change binds i to either accept or reject a proposal, if one is received; the agent is free to

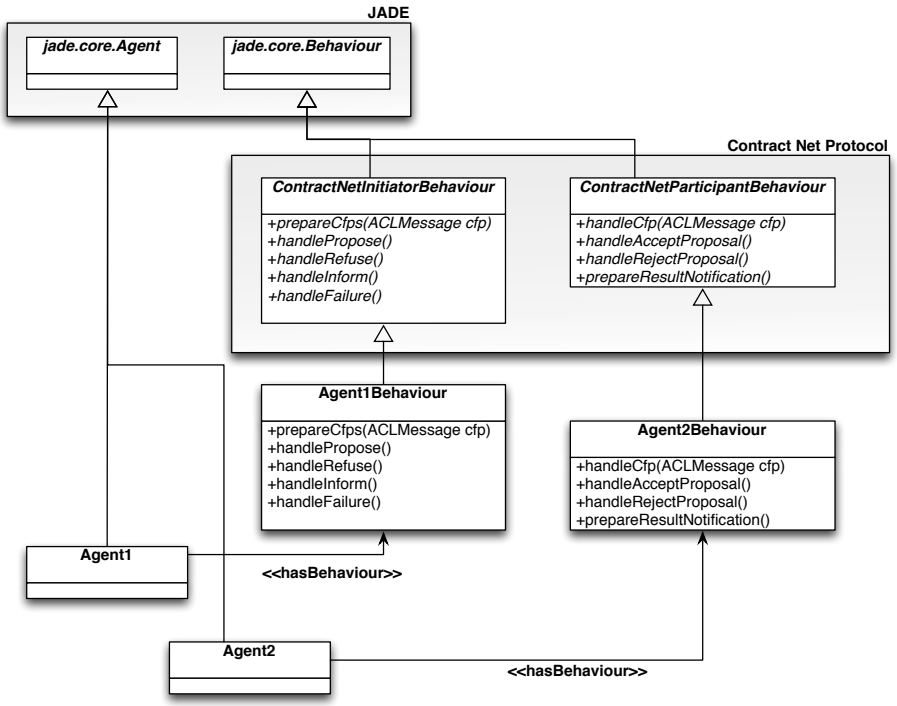

Fig. 8. UML diagram for the JADE implementation of CNP.

decide not only which course of action to take but also how to realize acceptance or rejection. On the other hand, this change is signaled to the agent playing the role Participant, who will handle it in some manner (depending on its behaviors) and decide whether sending a proposal. Instead, when a *accept* is executed the raised event automatically discharges a commitment created by a *cfp*.

This approach is illustrated in Figure 7. We modeled CNP as a Commitment Communication Artifact. Roles are *inner classes* within the artifacts, allowing JADE agents to use them. The protocol consists of a set of *social actions*, each of which has both an impact on the social state of the interaction and on the communication between agents. Actions are attributed to roles. For instance, action *cfp* is attributed to the role *Initiator*. For what concerns communication, the execution of a social action amounts to sending the content to be communicated through the tuple space provided by CArtAgO. This result is obtained by exploiting the method *send* of the CommitmentCommunicationArtifact. Commitments are handled as an instance of the class SocialState which is part of the CommitmentCommunicationArtifact. For example, consider the social action *cfp*, whose execution creates a commitment. This result is achieved through the execution of the following artifact operation:

```
@OPERATION
public void cfp(Task task, Role initiator, Role participant) {
    Message cfp = new Message();
    // setting of cfp parameters
    send(cfp);
    create(new Commitment(initiator, participant, new Fact(``propose''),
           new CompositeExpression(LogicalOperatorType.OR,
                       new Fact(``accept''), new Fact(``reject''))));
}
```

The first part of the operation manages the communication level, while the latter manages the creation of the commitment. The action *cfp* attributed to the role Initiator merely calls the described artifact operation.

An agent that will to play as a certain role can inspect the commitments that are required by the role itself, which are the commitments it will possibly be involved in as a debtor. In order to be able to satisfy them, the agent needs to have appropriate behaviors, otherwise its role execution is bound to fail. Notice that the agent is autonomous in selecting which social actions to execute and when as well as how to behave in order to satisfy its commitments.

Looking at Figure 8, the reader can perceive a major drawback of the original JADE approach: being part of an interaction protocol entails the adoption of an entire behavior, that must be added to the set of the internal agent behaviors. The resulting agent design breaks the autonomy of the agent, since the agent has an additional behavior for each role of each interaction it takes part to, increasing the possibility of conflicts between behaviors, and increasing the overall agent design complexity. In fact, being such behaviors FSMBehaviors, they implement Finite State Machines, i.e. they rigidly prescribe the sequences of actions that the agent is allowed to execute without any flexibility. Thus, it is not possible to intervene on the logic by which actions are sequentialized but only to realize the methods that the predefined behavior requires to redefine, which roughly correspond to decision points. Furthermore, this approach hinders the observability of the interaction, unless the designer adds specific sniffing or audit agents to log every message passed. In performance-critical applications, having more agents and producing a message overhead can produce undesirable scenarios.

5 Related works, discussion and future work

2COMM is a first step towards the implementation of the Mercurio architecture, proposed in [3, 4]. It realizes a programmable communication channel by means of artifacts, which is interaction-centric, exploits the social meaning of interaction supplied by commitment protocols, and enables the development of monitoring functionalities. The realization of roles is inspired by [8, 9]. The use of commitments gives a normative value to the encoded protocol, while the act of using a communication artifact amounts to the explicit acceptance, by the agent, of the rules of the protocol. This makes the current proposal very different from [7], whose aim was the introduction of the notion of role, as in [8,

9], inside JADE. The proposal conjugates the flexibility and the openness that are typical of MAS with the need of modularity and compositionality that are typical of design and development methodologies. The realization of commitment protocols as artifacts is an advancement of research on commitment-based approaches, w.r.t. approaches like [19], where commitment management resides in a middleware which, in turn, relies on a message-exchange communication infrastructure. Even though the function of the middleware recalls that of our artifacts, artifacts are, by their nature, distributed (and not centralized), they can be the result of the composition of other artifacts, can be manipulated and customized by the agents themselves. Moreover, the adoption of tuple spaces allows more variegated forms of communication where communication actions are not limited to utterances.

We believe that a commitment approach brings relevant advantages in terms of design and modeling flexibility, modularity and traceability. The resulting artifact explicitly provides a notion of *Role* that is decoupled from the interacting agent, instead of cabling it into an agent behavior (as in the JADE Methodology) or of composing different atomic roles to build an agent type (as in the GAIA Methodology [43]). Both approaches break into inner agent definitions, hindering the agent autonomy and the openness of the system. The artifact entity supplies a natural way for logging and audit purposes, leveraging the concept of social state (and its evolution). In a pure agent environment (like JADE), a similar result is obtained via a massive use of either message-sniffing agents and/or auditing agents, with a consequent overhead of the number of messages that are passed. This is, for example, the case of the proposal in [29]. By being an observable property, the social state provides the agent society a clear vision of who is responsible of what, in which protocol interaction, and when an agent acted so as to fulfill its commitments.

2COMM focuses on the interaction protocol layer, leaving aside issues concerning the society of agents in which the interaction takes place. Thus, it does not, for instance, tackle how to deal with violations of commitments. In order to properly handle these aspects it would be interesting to combine its use with proposals from the area of e-institutions. Concerning this field 2COMM would provide an improvement in that it would introduce the possibility to account for indirect forms of communication. As [25] witness, there is an emerging need of defining a more abstract notion of action, which is not limited to direct speech acts, whose use is not always natural. Along this direction, it is relevant to mention the OCeAN meta-model for artificial institutions [26], which encompasses a notion of commitment, and for which a possible architecture is discussed in [31]. For what concerns organizations, instead, there are some attempts to integrate them with artifacts, e.g. ORA4MAS [27] and JaCaMo `http://jacamo.sourceforge.net`, which also accounts for BDI agents. Following the A&A perspective, artifacts are concrete bricks used to structure the agents' world: part of which is the organizational infrastructure, part amounts to artifacts introduced by specific MAS applications, including entities/services belonging to the external environment. In [27] the organizational infrastructure

is based on $Moise^+$, which allows both for the enforcement and the regimentation of the rules of the organization. This is done by defining a set of conditions to be achieved and the roles that are permitted or obliged to perform them. The limit of this approach is that it cannot capture contexts in which regulations are, more generally, norms because norms cannot be restricted to achievement goals. Recently, the use of a communication infrastructure based on artifacts has been proposed to define, in an explicit and clear way, interaction in JaCaMo [33]. Nevertheless, the proposal does not supply a normative account of communication.

Finally, we think that our proposal can give significant contributions in industrial applicative contexts, for the realization of business processes and, in particular, of human-oriented workflows, whose nature is intrinsically social and where the notion of commitment plays a fundamental role [28]. In [36], the authors present LoST, a commitment-based model for the definition of declarative protocols, which is based on local history vectors of sent/received messages, associated to each of the interacting agents. LoST enables the representation and monitoring of (business) protocols when it is necessary to transfer local knowledge about occurring interactions between the agents. It works as an adapter for message transfer between agents. 2COMM, instead, provides agents an environment by which they communicate and, if this is requested, they can perform actions which do not amount to utterances but still entail social effects.

As a future work, we devise an extension of 2COMM for tackling a more expressive protocol language, with support for temporal constraints, see also [2]. This goal can easily be achieved by defining new artifact types that provide developers the appropriate protocol language primitives, such as those offered by 2CL [6][5].

Acknowledgments. The authors would like to thank the reviewers for their comments and the participants to the EMAS 2013 workshop for the discussions.

References

1. FIPA specifications. http://www.fipa.org.
2. M. Baldoni and C. Baroglio. Some Thoughts about Commitment Protocols (Position Paper). In *Post-Proc. of the 10th Int. Workshop on Declarative Agent Languages and Technologies X, DALT 2012, Revised Selected and Invited Papers*, *LNAI 7784*, pages 190–196. Springer, 2013.
3. M. Baldoni, C. Baroglio, F. Bergenti, E. Marengo, V. Mascardi, V. Patti, A. Ricci, and A. Santi. An interaction-oriented agent framework for open environments. *AI*IA 2011: Artificial Intelligence Around Man and Beyond*, 6934, 2011.
4. M. Baldoni, C. Baroglio, E. Marengo, V. Patti, and A. Ricci. Back to the future: An interaction-oriented framework for social computing. In *First Int. Workshop on Req. Eng. for Social Computing, RESC*, pages 2–5. IEEE, 2011.
5. M. Baldoni, C. Baroglio, E. Marengo, F. Capuzzimati, and V. Patti. A Generalized Commitment Machine for 2CL protocols and Its Implementation. In *Post-Proc. of the 10th Int. Workshop on Declarative Agent Languages and Technologies X, DALT*

2012, Revised Selected and Invited Papers, *LNAI 7784*, pages 96–115. Springer, 2013.
6. M. Baldoni, C. Baroglio, E. Marengo, V. Patti, and F. Capuzzimati. Engineering commitment-based business protocols with 2CL methodology. *J. of Autonomous Agents and Multi-Agent Systems*, August 2013 (to appear).
7. M. Baldoni, G. Boella, V. Genovese, A. Mugnaini, R. Grenna, and L. van der Torre. A Middleware for Modelling Organizations and Roles in Jade. In *Programming Multi-Agent Systems, 7th Int. Workshop, ProMAS 2009, Revised Selected Papers*, *LNAI 5919*, pages 100–117. Springer, 2010.
8. M. Baldoni, G. Boella, and L. van der Torre. Bridging Agent Theory and Object Orientation: Agent-like Communication among Objects. In *Post-Proc. of the International Workshop on Programming Multi-Agent Systems, ProMAS 2006*, *LNAI 4411*, pages 149–164. Springer, 2007.
9. M. Baldoni, G. Boella, and L. van der Torre. Interaction between Objects in `powerjava`. *Journal of Object Technology*, 6(2), 2007.
10. F. Bellifemine and A. Poggi. JADE – A FIPA-compliant agent framework. *Proceedings of PAAM*, 1999.
11. F. Bellifemine, A. Poggi, and G. Rimassa. Developing multi-agent systems with a FIPA-compliant agent framework. *Software-Practice and Experience*, (July 1999):103–128, 2001.
12. C. Bernon, M. P. Gleizes, S. Peyruqueou, and G. Picard. Adelfe: A methodology for adaptive multi-agent systems engineering. In *ESAW*, volume 2577 of *LNCS*, pages 156–169. Springer, 2002.
13. G. Boella and L. W. N. van der Torre. An agent oriented ontology of social reality. In *Procs. of Formal Ontologies in Information Systems (FOIS)*. IOS Press, 2004.
14. G. Boella and L. W. N. van der Torre. The ontological properties of social roles in multi-agent systems: definitional dependence, powers and roles playing roles. *Artificial Intelligence and Law*, 15(3):201–221, 2007.
15. O. Boissier, R. H. Bordini, J. Hübner, A. Ricci, and A. Santi. Multi-agent oriented programming with jacamo. *Science of Computer Programming*, 2011.
16. G. Caire, W. Coulier, Fr. J. Garijo, J. Gomez, J. Pavón, F. Leal, P. Chainho, Paul E. Kearney, J. Stark, R. Evans, and P. Massonet. Agent oriented analysis using message/uml. In *Proc. of AOSE 2001*, pages 119–135. Springer-Verlag, 2002.
17. G. Caire, D. Gotta, and M. Banzi. Wade: a software platform to develop mission critical applications exploiting agents and workflows. In *AAMAS (Industry Track)*, pages 29–36. IFAAMAS, 2008.
18. C. Castelfranchi. Principles of Individual Social Action. In *Contemporary action theory: Social action*, volume 2, pages 163–192, Dordrecht, 1997. Kluwer.
19. A. K. Chopra and M. P. Singh. An Architecture for Multiagent Systems: An Approach Based on Commitments. In *Proc. of ProMAS*, 2009.
20. A.K. Chopra. *Commitment Alignment: Semantics, Patterns, and Decision Procedures for Distributed Computing.* PhD thesis, North Carolina State University, Raleigh, NC, 2009.
21. R. Conte, C. Castelfranchi, and F. Dignum. Autonomous Norm Acceptance. In *Proc. of ATAL*, volume 1555 of *LNCS*, pages 99–112. Springer, 1998.
22. M. Dastani, D. Grossi, Meyer. J.-J. Ch., and N. A. M. Tinnemeier. Normative Multi-agent Programs and Their Logics. In *KRAMAS*, pages 16–31, 2008.
23. Jan L.G. Dietz. Understanding and modelling business processes with demo. In *Proc. of ER'99, 18th Int. Conf. on Conceptual Modeling*, volume 1728 of *LNCS*, pages 188–202. Springer, 1999.

24. Mohamed El-Menshawy, Jamal Bentahar, and Rachida Dssouli. Symbolic model checking commitment protocols using reduction. In *DALT*, volume 6619 of *Lecture Notes in Computer Science*, pages 185–203. Springer, 2010.
25. N. Fornara, F. Viganò, and M. Colombetti. Agent communication and artificial institutions. *Autonomous Agents and Multi-Agent Systems*, 14(2):121–142, 2007.
26. Nicoletta Fornara, Francesco Viganò, Mario Verdicchio, and Marco Colombetti. Artificial institutions: a model of institutional reality for open multiagent systems. *Artif. Intell. Law*, 16(1):89–105, 2008.
27. J. F. Hubner, O. Boissier, R. Kitio, and A. Ricci. Instrumenting multi-agent organisations with organisational artifacts and agents: "Giving the organisational power back to the agents". *Autonomous Agents and Multi-Agent Systems*, 20, 2009.
28. R. Medina-Mora, T. Winograd, R. Flores, and F. Flores. The action workflow approach to workflow management technology. *Inf. Soc.*, 9(4):391–404, 1993.
29. M. T. Nguyen, P. Fuhrer, and J. Pasquier-Rocha. Enhancing e-health information systems with agent technology. *Int. J. Telemedicine Appl.*, 2009:1:1–1:13, 2009.
30. M. Nikraz, G. Caire, and P. A. Bahri. A Methodology for the Analysis and Design of Multi-Agent Systems using JADE. (May), 2006.
31. D. Okouya, N. Fornara, and M. Colombetti. An Infrastructure for the Design and Development of Open Interaction Systems. In *Proc. of the 1st International Workshop on Engineering Multi-Agent Systems, EMAS 2013, held in conjuction with AAMAS 2013*, pages 128–143, St. Paul, Minnesota, USA, May 2013.
32. 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, 2011.
33. T. F. Rodrigues, A. C. da Rocha Costa, and G. P. Dimuro. A Communication Infrastructure Based on Artifacts for the JaCaMo Platform. In *Proc. of the 1st Int. Workshop on Engineering Multi-Agent Systems, EMAS 2013*, pages 97–111, St. Paul, Minnesota, USA, May 2013.
34. M. P. Singh. An Ontology for Commitments in Multiagent Systems. *Artif. Intell. Law*, 7(1):97–113, 1999.
35. M. P. Singh. A social semantics for agent communication languages. In *Issues in Agent Communication*, volume 1916 of *LNCS*, pages 31–45. Springer, 2000.
36. M. P. Singh. LoST: Local Transfer - An Architectural Style for the Distributed Enactment of Business Protocols. *Proc. of the 9th Internactional Conference on Web Services*, pages 57–64, IEEE Computer Society, 2011.
37. P. R. Telang and M. P. Singh. Specifying and Verifying Cross-Organizational Business Models: An Agent-Oriented Approach. *IEEE Transactions on Services Computing*, pages 1–14, 2011.
38. T. Winograd and F. Flores. *Understanding computers and cognition - a new foundation for design.* Addison-Wesley, 1987.
39. M. F. Wood and S. A. DeLoach. An overview of the multiagent systems engineering methodology. In *AOSE*, *LNCS 1957*, pages 207–222. Springer, 2000.
40. P. Yolum and M. P Singh. Designing and executing protocols using the event calculus. *Proc. of the 5th Int. Conf. on Autonomous agents - AGENTS '01*, pages 27–28, 2001.
41. P. Yolum and M. P. Singh. Commitment Machines. In *Intelligent Agents VIII, 8th International Workshop, ATAL 2001*, *LNCS 2333*, pages 235–247. Springer, 2002.
42. Pinar Yolum. Design time analysis of multiagent protocols. *Data Knowledge Engineering*, 63(1):137–154, 2007.
43. F. Zambonelli, N. R. Jennings, and M. Wooldridge. Developing multiagent systems: The gaia methodology. *ACM Trans. Softw. Eng. Methodol.*, 12(3):317–370, 2003.