2COMM: a commitment-based MAS architecture

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. This paper proposes a multi-agent system architecture where agent interaction is ruled with the help of commitment-based interaction protocols. Commitment protocols are embodied into artifacts which can be accessed and used by the interacting agents. Ideally, the architecture is orthogonal to the language that is used to specify the commitment protocols. In this paper we rely on Yolum and Singh's proposal. The implementation that is described relies on the well-known JADE and CArtAgO frameworks.

Keywords: Commitment, 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 [27, 12, 24, 17]. 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 [15]. 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.

The aim of this work is to fill the 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 [24, 25, 30], JADE notoriously lacks of. Following [3], we perform such an extension by enabling a form of indirect communication among agents with the help 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.

JADE [4], [5] is a well-established development environment for multi-agent systems. It is FIPA-compliant and actually used for industrial applications. Our starting point for introducing commitment-based protocols inside JADE was the JADE Methodology [22]. 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, allowing integration with eventually current 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*. 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 [2] 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 Fig. 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 [11].

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 [26]. 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 [30, 29, 14]. 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 [30].

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 ([23]). 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.

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 [15]. 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 [16]. 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.

3 2COMM: Reifying Commitment Protocols with Artifacts

In this section, we describe the original contribution of this work, which is the realization of a commitment-based MAS architecture that we named *2COMM* (Communication & Commitment). To this aim, we rely upon the JADE and the CArtAgO frameworks, introducing a mediated form of interaction among JADE agents. We realized mediated interaction by means of communication artifacts, which, in our proposal, replace the JADE-based FIPA protocols and which reify commitment-based protocols [3]. In Fig. 1 we draw the basic architecture of 2COMM. 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 [29].

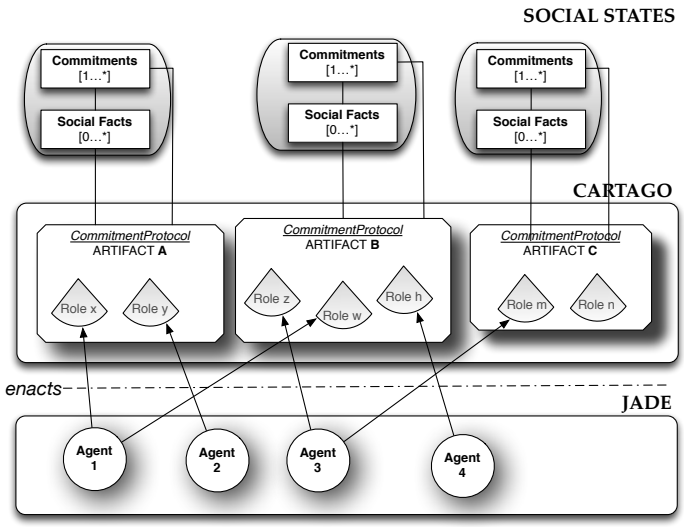

Fig. 1. The Basic Architecture of 2COMM.

3.1 Communication Artifact

We follow the ontological model for organizational roles proposed in [7, 8], 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 supertype of communication artifacts implementing specific interaction protocols (e.g., Contract Net, Net Bill, Brokering): the *BasicCommitmentCommunicationArtifact*. We call an instance of an artifact of type BasicCommitmentCommunicationArtifact 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 BasicCommitmentCommunicationArtifact presents an observable property, *Roles*, that is the collection of the roles of the protocol (definitional dependence [7, 8]). Actions have a social effect only when they are executed by the role they are assigned to, but actions are not defined at this super level, rather they are provided by the instantiations of the BasicCommitmentCommunicationArtifact, i.e. by artifacts implementing specific protocols. Each protocol action is implemented as a public *operation*, which is associated

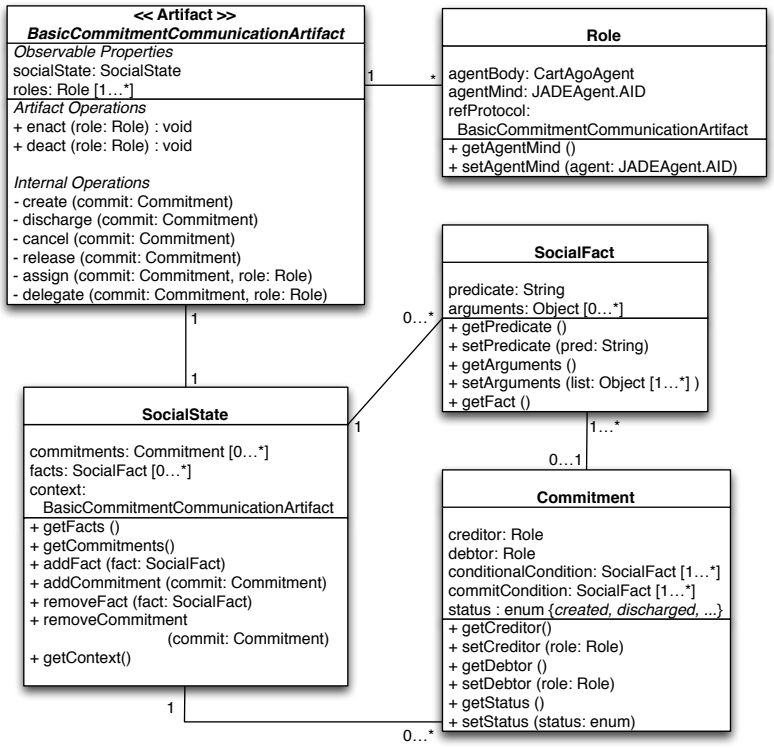

Fig. 2. The UML Class diagram for the commitment-based communication artifact.

to a role by means of an *operation guard* (institutional empowerment [7, 8]): 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, othewise, 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 [7, 8]. An agent who does not show such capabilities is bound to violate its commitments.

BasicCommitmentCommunicationArtifacts provide a property, tracking the identity of the agents actually playing the various role. Two operations are provided in order to manage the association between an agent's identity and a role: *enact(Role role)* and *deact(Role role*), by means of which an agent can explicitly assume/cease a protocol role (foundation [7, 8]). 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 Fig. 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 [29]: *create, discharge, cancel, release, assign, delegate*. The operations regarding the commitments life-cycle are implemented as artifact *internal operations*, therefore, the agents cannot modify them 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 Fig. 3.

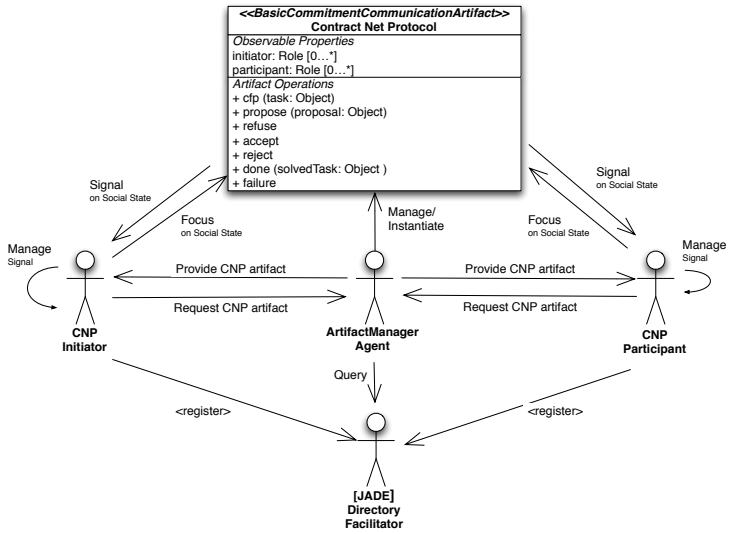

Fig. 3. The interaction between the main elements of our proposal, in a CNP example.

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 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 [29]. A reference model to include, in the future, also institutional aspects could be the JaCaMo proposal [9].

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

We use the JADE Methodology [22] to model a real-scenario MAS, 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.

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 [28], Adelfe [6] or MESSAGE [10]). 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" [22], 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. 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* with initial acquaintances. No distinction is made between acquaintances and 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. 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

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 kind of analysis is synthesized in 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 patterns (like Contract Net Protocol, Query Protocol and Request Protocol), thus shaping the MAS interaction patterns via reified commitment protocols. The resulting model is depicted in Fig. 4, whilst in Fig. 5 we zoomed into the implementation of one of the commitment artifacts, the *Contract Net Protocol* artifact.

In Fig. 6 we highlight the very same protocol, implemented via pure JADE behaviors. Looking at the picture, the reader can perceive a major drawback of the latter 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. 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.

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.

Instead of basing interaction design directly on JADE behaviors, we propose 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 *action*. 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. This approach is illustrated in Fig. 5, where the commitment-based definition of CNP is provided. The protocol consists of a set of *social actions*, each of which has an impact on the social state of the interaction. For example, when an agent playing the role p (participant) executes the artifact action *propose*, the social state is modified by creating the commitment *C(p, i, accept, done* $\vee$ *failure)*. This change is signaled to the agent playing the role i (initiator), who will handle it in some manner (depending on its behaviors) and decide whether accepting the proposal of that participant. Instead, when a *failure* is executed the raised event automatically discharges a commitment created by a *propose*.

5 Related works, discussion and future work

2COMM is a first step towards the implementation of the ideas proposed in [3], realizing a programmable communication channel by means of artifacts, which is interaction-centric, exploits the social meaning of interaction supplied by commitment protocols, and enables monitoring functionalities. The use of commit-

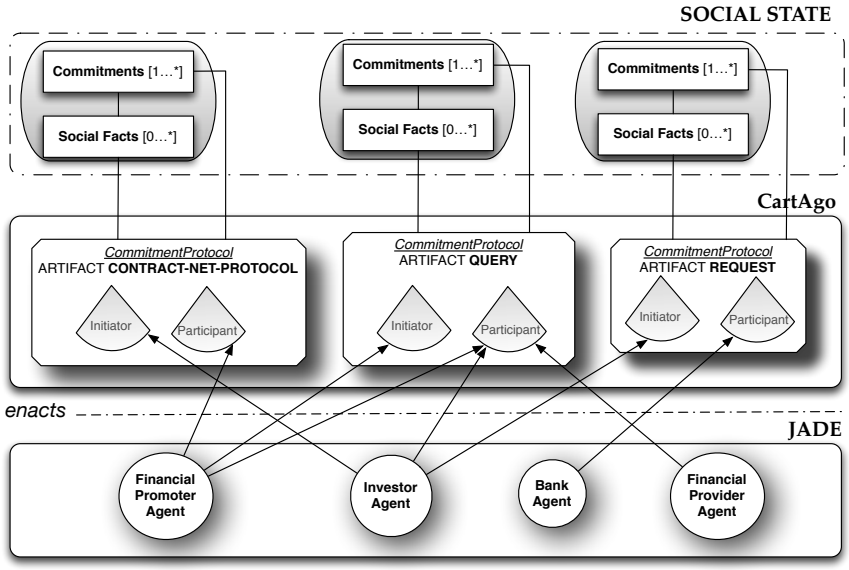

Fig. 4. The FinancialMAS Commitment-based Interaction Architecture.

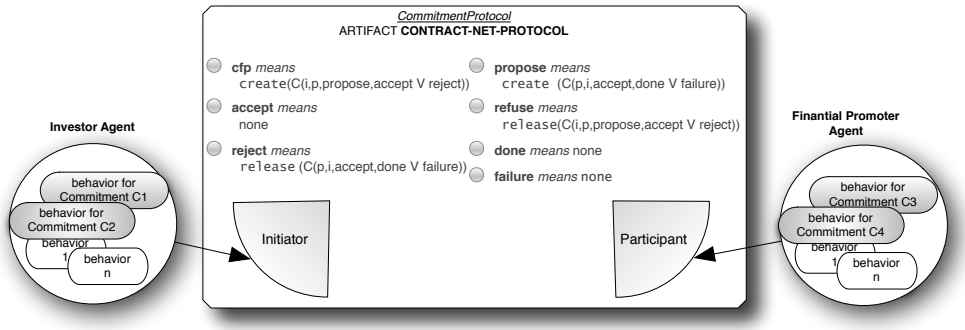

Fig. 5. The 2COMM artifact for Contract Net Protocol.

ments 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. 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/environments is an advancement of research on

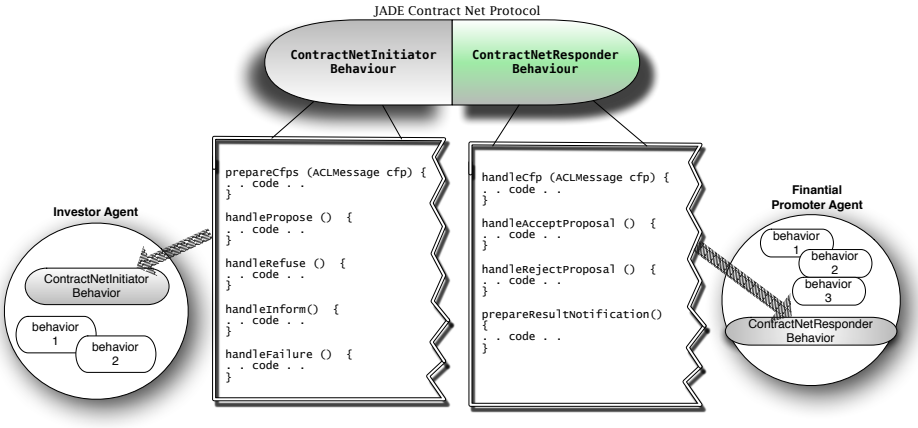

Fig. 6. The JADE implementation for Contract Net Protocol.

commitment-based approaches, w.r.t. approaches like [13], where these elements reside in the middleware, and are shielded from the agents and from the designer.

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 [31]). 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 [21]. 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 focusses 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 [18] witness, there is an emerging need of defining a more abstract notion of action, which is not limited to direct speech acts, whose use is not always natural. For what concerns organizations, instead,

there are some attempts to integrate them with artifacts, e.g. ORA4MAS [19] 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 [19] the organizational infrastructure is based on $Moise^{+}$, which allows both for the enforcement and the regimentation of the rules of the organization. This is done by defining a set of conditions to be achieved and the roles that are permitted or obliged to perform them. The limit of this approach is that it cannot capture contexts in which regulations are, more generally, norms because norms cannot be restricted to achievement goals.

Finally, we think that our proposal can give significant contributions to industrial applicative contexts, in particular 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 [20].

References

1. FIPA specifications. http://www.fipa.org.
2. 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.
3. 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.
4. F. Bellifemine and A. Poggi. JADE – A FIPA-compliant agent framework. *Proceedings of PAAM*, 1999.
5. 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.
6. 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.
7. 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.
8. 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.
9. O. Boissier, R. H. Bordini, J. Hübner, A. Ricci, and A. Santi. Multi-agent oriented programming with jacamo. *Science of Computer Programming*, 2011.
10. 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.
11. 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.

12. C. Castelfranchi. Principles of Individual Social Action. In G. Holmstrom-Hintikka and R. Tuomela, editors, *Contemporary action theory: Social action*, volume 2, pages 163–192, Dordrecht, 1997. Kluwer.
13. A. K. Chopra and M. P. Singh. An Architecture for Multiagent Systems: An Approach Based on Commitments. In *Proc. of ProMAS*, 2009.
14. A.K. Chopra. *Commitment Alignment: Semantics, Patterns, and Decision Procedures for Distributed Computing.* PhD thesis, North Carolina State University, Raleigh, NC, 2009.
15. R. Conte, C. Castelfranchi, and F. Dignum. Autonomous Norm Acceptance. In *Proc. of ATAL*, volume 1555 of *LNCS*, pages 99–112. Springer, 1998.
16. 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.
17. 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.
18. N. Fornara, F. Viganò, and M. Colombetti. Agent communication and artificial institutions. *Autonomous Agents and Multi-Agent Systems*, 14(2):121–142, 2007.
19. J. F. Hubner, O. Boissier, R. Kitio, and A. Ricci. Instrumenting multi-agent organisations with organisational artifacts and agents: "Giving the organisational power back to the agents". *Autonomous Agents and Multi-Agent Systems*, 20, 2009.
20. 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.
21. 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.
22. M. Nikraz, G. Caire, and P. A. Bahri. A Methodology for the Analysis and Design of Multi-Agent Systems using JADE. (May), 2006.
23. 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.
24. M. P. Singh. An Ontology for Commitments in Multiagent Systems. *Artif. Intell. Law*, 7(1):97–113, 1999.
25. M. P. Singh. A social semantics for agent communication languages. In *Issues in Agent Communication*, volume 1916 of *LNCS*, pages 31–45. Springer, 2000.
26. 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.
27. T. Winograd and F. Flores. *Understanding computers and cognition - a new foundation for design.* Addison-Wesley, 1987.
28. M. F. Wood and S. A. DeLoach. An overview of the multiagent systems engineering methodology. In Paolo Ciancarini and Michael Wooldridge, editors, *AOSE*, volume 1957 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2000.
29. 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.
30. P. Yolum and M. P. Singh. Commitment Machines. In *Intelligent Agents VIII, 8th International Workshop, ATAL 2001*, volume 2333 of *LNCS*, pages 235–247. Springer, 2002.
31. F. Zambonelli, N. R. Jennings, and M. Wooldridge. Developing multiagent systems: The gaia methodology. *ACM Trans. Softw. Eng. Methodol.*, 12(3):317–370, 2003.