Programming and Reasoning about Social Relationships: a Commitment-based Infrastructure

MATTEO BALDONI, Università degli Studi di Torino, Dipartimento di Informatica
CRISTINA BAROGLIO, Università degli Studi di Torino, Dipartimento di Informatica
FEDERICO CAPUZZIMATI, Università degli Studi di Torino, Dipartimento di Informatica

2COMM is a new agent-oriented middleware for realizing cross-organizational systems. Due to the autonomy of the interacting peers, which generally correspond to independent organizations, the need is the decoupling of the design of the business process, which shapes the interaction, and the design of the interacting agents. 2COMM does so by realizing the principles advocated for Socio-Technical Systems, it combines the use of agents with the use of artifacts, and it exploits commitments and commitment-based protocols for implementing the social layer of the system as well as for implementing cross-organizational business processes.

Categories and Subject Descriptors: I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems; I.2.11 [**Distributed Artificial Intelligence**]: Languages and structures

General Terms: Design, Infrastructures

Additional Key Words and Phrases: Interaction protocols, commitments, social relations, middleware, artifacts, sociotechnical systems

ACM Reference Format:
Matteo Baldoni, Cristina Baroglio, Federico Capuzzimati, 2013. Programming and Reasoning about Social Relationships: a Commitment-based Infrastrcture. *ACM Trans. Internet Technol.* 9, 4, Article 39 (March 2010), 20 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Modern, complex information technology systems require new software engineering techniques in order to scale adequately and to reduce risks of undesired, unpredictable behaviours [Sommerville et al. 2011]. In particular, traditional approaches do not fit the needs of large-scale, multi-party, cross-organizational systems, especially those needs which concern the design of interaction and distributed computation.

A different way of imagining and of engineering applications that fit the described setting is via *socio-technical systems* (STS for short) [Trist 1981; Cherns 1976]. These are systems where several, autonomous actors (people who use or interact with other system components) use resources of some nature to achieve individual or shared goals. STS can be viewed as an evolution of computing [Whitworth and Ahmad 2013] into *social computing* [Dalpiaz et al. 2011], with a transition from an individualistic to a societal view, where notions like *social structure*, *role* and *norm* come into play.

Socio-technical systems support human users by mechanizing processes and by aiding stakeholders in interacting with each other, e.g. for contending resources, asking

Author's addresses: M. Baldoni, C. Baroglio, F. Capuzzimati, Dipartimento di Informatica, Università degli Studi di Torino c.so Svizzera 185, I-10149 Torino (Italy).

DOI: http://dx.doi.org/10.1145/0000000.0000000

for co-working activities, or for assigning sub-units of work. A common way to model user activities is by means of *business processes*, that the performers have to follow for achieving some result. Business processes, in their classic definition, are sets of structured, ordered tasks which impose a strict arrangement onto subtask execution: diverging from it results in a failure. A *de facto* standard notation for business processes is provided by the Business Process Model and Notation (BPMN) [White 2004], a graphical language that models a business process as a decorated, enriched flowchart.

Although BPMN is a preferred choice for modeling single-actor activities, and even if its latest release (BPMN 2.0) introduces support for a *message* element, it still suffers from major drawbacks in modeling cross-organizational business processes (CO-BP for short) [Desai et al. 2009] and, in general, systems with different stakeholders. The reason relies in the procedural/imperative nature of BPMN: a step-by-step process representation does not consider the autonomy of the interacting entities and hinders their decisional capabilities, when it is used to define both interaction *and* how it must be carried out.

Opposed to this approach, we think that the systems at issue can easily be modeled as *Open Multi-Agent Systems* (O-MAS). Among the others, this claim is supported by [Singh 2013; Porello et al. 2013; Bresciani and Donzelli 2003]. A Multi-Agent System (MAS) is an system where different autonomous agents act and interact with each other to fulfill some goal. Whilst in a closed MAS agents may know each other's individual capabilities, typically because agents and system designers belong to the very same organization, in an O-MAS no assumption on the other agents' behaviour or design can be made. O-MAS can be used to design systems with several stakeholders, that share the very same computational environment, that have to cooperate or to compete to reach individual or group achievements: in other words, O-MAS can be used to develop socio-technical systems.

The regulation of stakeholder interactions calls for a formalism that allows capturing *1)* what partners want to achieve, *2)* what they can do during an interaction for achieving their goal, and *3)* the rules which norm the interaction. How to design a pattern of interaction without imposing behavioural constraints on actors? Being stakeholders inherently autonomous, the adoption of a workflow-based approach (like BPMN) is not adivisable because it would over-constrain the interaction. The alternative is to envisage a business process, encompassing many actors, as an *interaction protocol* defining *social relationships* among the involved parties. Following [Singh 2013], social relationships are norms, they provide the *invariants* of an STS, and govern the behavior of the peers taking part into it. We will, then, shape business processes by starting from the *social relationships* between the stakeholders. The designers will capture the relationships among the parties within a certain coordination pattern, rather than imposing strict procedures. The focus here is on relationships, and how they evolve during the interaction.

Nowadays, agent-oriented software engineers can choose from a substantial number of agent platforms [Mascardi et al. 2004; Bordini et al. 2006; Fisher et al. 2007; Baldoni et al. 2010]. The choice is related to very different and heterogeneous factors, like: scope and purpose of the system; formal model the platform is based on; richness of the agent programming language (APL), if devised; platform support, maintenance and update; (graphical) tools for supporting design and development; simplicity of integration with other (agent) programming languages. In our work, agents support the various stakeholders, and their implementations may be heterogeneous. For this reason we do not define nor adopt a specific APL but rather look for a *middleware* that is capable of supporting the realization of STS, respecting the principles so far advocated. Platforms and frameworks like JADE [Bellifemine et al. 2005], TuCSoN [Omicini and Zambonelli 1998], DESIRE [Brazier et al. 1997], JIAC [Thiele et al. 2009], all provide

coordination mechanisms and *communication infrastructures* [Bordini et al. 2006]; we claim, however, that none of the current infrastructures for MAS is adequate to the purpose of realizing STS because they are too much "communication-centric", and *do not account* for the social aspects of interaction.

For this motives, we propose in this work a new middleware. The realization uses JADE as basic agent platform, providing a FIPA compliant communication framework and an agent-developing middleware. In order to reify the social relationships we rely on the Agents & Artifacts meta-model (A&A) [Weyns et al. 2007; Omicini et al. 2008], which provides abstractions for environments and artifacts, that can be acted upon, observed, perceived, notified, and so on. When embodied inside artifacts, social relationships can be examined by the agents (to take decisions about their behavior), as advised in [Chopra and Singh 2009], used (which entails that agents accept the corresponding regulations), constructed, e.g., by negotiation, specialized, composed, and so forth. For supplying a normative characterization of social relationships, we propose to rely on commitments, which feature a social and observational semantics [Singh 2000]. The approach allows the realization of both direct and indirect forms of communication, and the distributed nature of artifacts and their compositionality perfectly fits the characteristics of open CO-BP. Last but not least, the use of artifacts enables the implementation of monitoring functionalities for verifying that the on-going interactions respect the commitments and for detecting violations and violators.

The article is organized as follows. Section 2 introduces Socio-Technical Systems, it reports the requirements on the implementation, the motivations to our proposal and some of the main choices and characteristic of it. Section 3 describes 2COMM, the system that we developed. Section 4 shows an application, set in a financial context, that helps explaining the proposal. Conclusions and related works end the paper.

2. MODELING SOCIAL RELATIONSHIPS: TOWARDS THE REALIZATION OF STS

In the 50's, a new model of work organization was developed by the Tavistock Institute of Human Relations. The founders believed that the tayloristic approach, based on rigid structures of pipelines, led to dishumanization of work and degradation of working life. They proposed an alternative, moving towards a different goal: work humanization, weighting technical systems as equal as social systems. This design principle tends to assign responsibilities, decentralize work structures, group employees into mindful teams; they referred to this with the term *socio-technical design* of *socio-technical systems* [Trist 1981].

Different design principles were proposed as basis for socio-technical design. One of them is particularly relevant for the design of modern systems [Cherns 1976], the *Minimal Critical Specification*: only the essential must be specified. In his work the author, member of the Tavistock Institute, highlights how in the organization design the focus is traditionally put on *how* rather than on *what*. Quote: "In any case, it is a mistake to specify more than is needed because by doing so options are closed that could be kept open. *This premature closing of options is a pervasive fault in design.*" Several companies tried to apply socio-technical principles, among them Philips in Netherlands, Volvo in Sweden or Olivetti in Italy.

Through decades, socio-technical principles evolved, adapting to new technologies and new business activities, in particular, after the introduction of computer systems starting from the 80's. The design focus remained the same: *interactions* between autonomous, conscious entities; a definition of work, based on what, rather than on how; working activities not restricted to routine tasks; design of work aggregation units decoupled from tasks design; decentralization of control. Despite practical, positive results in applying these principles, enterprises moved in the direction of restructuring assets and cutting costs, espousing the "computer-aided neo-Taylorism" approach

[Mumford 2006], or "lean production", focused on the standardization of work processes. Nowadays, the humanization of work and the decentralization of control are still debated issues, both at research and business levels. Some industries are shifting from centralized to decentralized structures, using networks instead of hierarchies to manage complexity, and leveraging interaction and cooperation instead of pipe-lining workers.

In parallel with real-world work organization cases, socio-technical design principles were adapted to software and systems engineering [Sommerville et al. 2011]. The very introduction of the agent metaphor represents a paradigm shift, from a *control-flow*-based organization of software (typical of the functional and the object-oriented paradigms) to a *message-based* one, shaped around message exchanging between autonomous, computational entities. Complex software systems can, thus, be modeled by means of *interactions* between components, whose coordination (either collaborative or competitive) is the fundamental pillar; besides interactions, norms and policies for resource usage are provided.

2.1. Commitments for Social Relationships

Companies adopt Business Process Management Systems to manage the business processes and the flow of activities. Each of the involved actors (including external systems and individuals) is referred to as a stakeholder, i.e. someone who has some interest in using or taking part to the system. When modeling complex domains, with different stakeholders belonging to different organizations or units of work, system engineers and analysts usually adopt the same formalism to describe behaviour of a particular component and interactions between components (system resources and/or stakeholders). BPMN is the preferred language; it allows to specify in detail what the expected behaviour of a resource (e.g. a printer, a web service, a web application) is, as well as that of an actor (e.g. the receptionist answering a call, the support team managing an help desk). It provides a graphical, intuitive means to organize tasks for achieving some result.

Problems arise when BPMN is used to design interactions between different stakeholders, each with its own set of business processes. Think, for example, to a health care system, whose purpose is to manage remote assistance to elder people and, if required, to send a medical support. Such system is a service integrator; it interrelates services provided by different entities when needed. Specifying how, for example, the physician will administer cures is out of its scope. The system only has to shape relationship between the physician and the patient who requested assistance. By using BPMN, one would build a multi-actor business process, where the tasks to be accomplished would be expressed in a procedural manner, and the interaction between the stakeholders (physician and patient) would be expressed as the sending and receiving of messages. This procedural view makes it difficult, for the involved actors, to be proactive or to adapt to variations in the environment, unless the business process is redesigned. This is basically due to to the imposition of (often) unnecessary orderings. In other words, this approach does not suit the representation of business processes that are inherently *destructured*, as it is often the case in cross-organizational settings, and where seldom authority is centralized or clearly associated to a single stakeholder, but, rather, the involved actors are all peers.

Pesic and van der Aalst proposed a shift of paradigm, passing from a procedural representation, leading to an over-constrained, non-realistic organization of work, to a declarative representation [Pesic 2008]. In our opinion, this shift of paradigm satisfies the Minimal Critical Specification requirement illustrated before: specify "what" should occur rather than "how" making it occur. The question to answer, then, becomes: how to model interactions between autonomous entities without considering

them as non-autonomous ones? Following [Singh 2013], norms provide the *invariants* of an STS, and govern the behavior of the peers taking part into it. We will, then, shape business processes by starting from the *social relationships* between the stakeholders.

Inspired by [Sommerville 2010], we see complex organizations as structured in three architectural macro-level: *infrastructural*, *functional* and *socio-organizational*. The first concerns hardware equipment, software and data management; the second organizes activities, involving actors and components; the third regulates and norms how actors can use the system, what the constraints are, and laws that rule some aspects of the system.

Traditional software engineering techniques and theories tendentiously do not tackle the third level explicitly, and, therefore, model the second as a collection of standard step-by-step, rigidly structured sequences of activities, performed by one or more actors, possibly interacting with each other, or using some system resource. On the contrary, we adopt the basic ideas used, among others, in [Telang and Singh 2011], where interactions and CO-BP are modeled in terms of regulated relations between entities, and where the founding element of a relation is a *commitment*. We rely on the Multi-agent Systems paradigm because it can represent domains that are composed by several autonomous entities, whose interaction determines the evolution of the system. Thus, we naturally model each stakeholder as an agent, and express relations between agents as commitments.

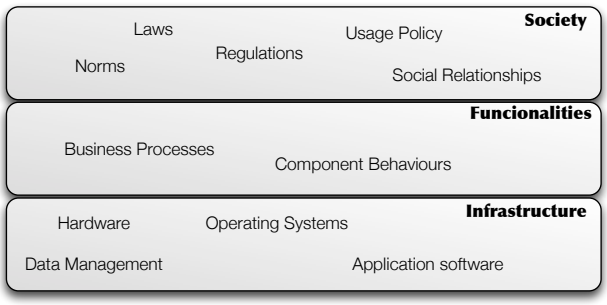

Fig. 1. Architectural macro-levels in a STS.

Commitments define the agents normative context. They constitute the *top level of a socio-technical system*, the one which accounts for the norms and the policies of the system, as well as for the laws and for the societal regulations which rule its functioning [Sommerville 2010]. Commitments [Singh 1999] are represented with the notation $\mathsf{C}(x, y, r, p)$, capturing that the agent x commits to the agent y to bring about the consequent condition p when the antecedent condition r holds. Antecedent and consequent conditions generally are conjunctions or disjunctions of events and commitments. When r equals $\top$, we use the short notation $\mathsf{C}(x, y, p)$ and the commitment is said to be *active*. Commitments have a *regulative* nature, in that debtors are expected to behave so as to satisfy the engagements they have taken. This practically means that an agent is expected to behave so as to achieve the consequent conditions of the active commitments of which it is the debtor.

The stakeholders share a social state that contains commitments. Some of these elements can hold since the beginning of the interaction. Stakeholders affect the social state while performing their activities. Specifically, the manipulation of commitments is done by means of the standard operations *create*, *cancel*, *release*, *discharge*, *assign*,

delegate. As in [Singh 1999], we postulate that discharge is performed concurrently with the actions that lead to the given condition being satisfied and causes the commitment to not hold. Delegate and assign transfer commitments respectively to a different debtor and to a different creditor. For details see [Singh 1999; Yolum and Singh 2002; Chopra 2009].

2.2. Artifacts for Developing the Socio-Organizational Layer

In a system made of autonomous and heterogeneous actors, social relationships cannot but concern the observable behavior [Dastani et al. 2008]. On the other hand, following [Conte et al. 1998], we need social relationships to be recognized as norms – so that they will provide social expectations on the stakeholders' behavior, to be accepted explicitly by the participants to the interaction, to be inspected so as to allow stakeholders to decide whether conforming to them. To this aim, we acknowledge social relationships as *resources* that should be explicitly modeled and made availalable to the interacting peers, in the very same way as other kinds of resources (e.g. hardware resources) are accounted for, see for instance [Sommerville 2010]. In order to realize STS, it is, therefore, necessary to provide the stakeholders the means to create and to manipulate the commitments, as well as to observe them so as to take proper decisions along their activities. Nothing of this kind exists, although Chopra and Singh in [Chopra and Singh 2009] proposed a commitment-based Multi-agent System layered architecture, and stated that a commitment-based middleware should offer abstractions for working with commitments.

Given that stakeholders and social relationships are both first-class entities, that interact in a bi-directional manner, how to model these two elements in the Multi-agent paradigm? This provides a single first-class entity, the *agent*, which is not suitable for representing inspectable elements because an agent's internal state is not available to external entities by definition. To solve this issue, we adopt the *Agents and Artifacts* (A&A) meta-model [Weyns et al. 2007; Omicini et al. 2008], that extends the agent paradigm with another primitive abstraction, the *artifact*. An artifact is a computational, programmable system resource, that can be manipulated by agents, residing at the same abstraction level of the agent abstraction class. The A&A paradigm provides ways for defining and organizing workspaces, i.e. 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, and encapsulating services and functionalities. For their very nature, artifacts can encode a mediated, programmable and observable means of communication and coordination between agents. Following [Baldoni et al. 2011], we rely on artifacts to realize the social layer of STS.

The next question to answer is how to integrate CO-BP inside the outlined agent and artifact setting. In our proposal, this is done by relying on the notion of *commitment-based interaction protocol*. This kind of protocols consists of a set of social actions (process activities), whose semantics is shared, and agreed upon, by all of the participants to the interaction [Chopra 2009; Yolum and Singh 2002]. The semantics of the social actions is given in terms of the operations which allow the modification of the social state and that have already been described.

In modeling protocols, we propose to start from social relationships, represented as commitments, and, then, define the activities that make them evolve during the interaction. This represents an inversion w.r.t. traditional, activity-centric design phase: the evolution of commitments guides the definition of those activities, the interacting parties need. An example of how to leverage commitments to build commitment-based business models can be found in [Telang and Singh 2010].

We interpret the fact that an agent uses an artifact as the explicit acceptance, by the agent, of the norms encoded by that artifact, and modeled by the interaction protocol that the artifact reifies. This is an important aspect because it allows the interacting parties to perform some kind of *practical reasoning*, based on expectations: participants will expect that the debtors of commitments will behave so as to satisfy the corresponding consequent conditions; if this will not happen, a violation will occur. Relying on artifacts allows also other kinds of manipulations, including (but not limited to) an agent playing a role in the interaction, an agent observing the interaction that is being carried on, a group of agents composing (negotiating, specializing) a business process.

2.3. Roles as Affordances for Decoupling the Interaction

From an organizational perspective, a protocol is structured into a set of *roles*. A role represents a way of manipulating the social state and belongs to the artifact which reifies a business process. Roles and stakeholders are different entities, and we assume that roles cannot live autonomously: they exist in the system in view of the interaction, because agents, for interacting, use artifacts and execute actions on them. From a cognitive perspective, roles organize the *affordances* [Gibson 1979] of an artifact. Citing Gibson: "The same layout will have different affordances for different animals, of course, insofar as each animal has a different repertory of acts. Different animals will perceive different sets of affordances therefore." In other words, different actors will perceive different ways of interaction with a same environment, depending on their own capacities.

The concept of affordance was brought inside the object-oriented paradigm in [Baldoni et al. 2006; Baldoni et al. 2007]. There, the authors claim that objects have affordances when they are considered relative to some other objects managing them. In this case, the properties which characterize an object in the environment depend on the properties of the interactant, as well as the ways in which it is possible to interact with a same object depend on the properties of the manipulator. These concepts are introduced in the object-oriented paradigm by introducing *role types*, where a role type captures the interaction possibilities with an object, thereby, capturing a set of affordances of the object. The interaction possibilities depend on the class of the interactant manipulating the object: in other words, they depend upon the player of the role.

In our current proposal, artifacts reify business processes (interaction protocols), thus, we can see them as *interaction artifacts*, realizing a form of mediated communication. Roles are supplied by the protocol and, in turn, supply actions that enable interaction through the artifact: they are *powers* for modifying the social state, e.g. by creating commitments. On the other hand, the agents that will be the *role players* need to satisfy the related *requirements*: specifically, in order to play a role an agent needs to have the capabilities of satisfying the related commitments – capabilities which can be internal of the agent or supplied as powers as well. For example, in order to play the role *Initiator* of the Contract Net Protocol (used later in the paper), an agent needs to be able to satisfy the commitment of assigning tasks to some of the participants.

In order to account for the continuity of the interaction with each object, the authors of [Baldoni et al. 2006] also propose a notion of state of the interaction, which, in the considered distributed scenario, is realized as a *session*. Thus a given role type can be instantiated, depending on a certain player of a role (which must have the required properties), and the role instance represents the state of the interaction with that role player. In our proposal, the artifact maintains both the social state, with all the commitments, and further elements that contribute to the state of the interaction, and that are stored in a blackboard-style memory, that we call communication state. For instance, in the contract net protocol case, the description of the task for which the call

is issued, is recorded in this area. Agents, playing roles, can inspect the state of the interaction or be notified when events of interest occur.

It is easy to see that the above proposal follows the ontological model proposed in [Boella and van der Torre 2007], which is characterized by three aspects: (1) *Foundation*: a role must always be associated with the 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. The proposal also realizes the seminal ideas coalesced around the Mercurio architecture [Baldoni et al. 2011]. Mercurio is the model upon which we built the proposed framework for programming social relationships.

The advantage of our proposal, compared to previously existing approaches, is that it clearly *decouples* the protocol design from the agent design, using artifacts to encode the interaction logic. Protocols provide a centralized means of coordination, based on the *notification* of social events, e.g. the execution of an action or the creation of a commitment. Interacting agents use artifacts to coordinate and perform social actions, depending on the role an agent plays and on its objectives. If and how making the interaction progress is entirely under the distributed control of the agents, along with artifact notifications. One further advantage is that, thanks to artifacts and to commitments and to their normative value, it is possible to realize monitoring functionalities of the interaction, without imposing any specific behavior. As such, our proposal respects the principles advocated for STS.

Traditional approaches, on the contrary, do not support the decoupling of the design of agents and of the design of their interactions, which would be necessary in a cross-organizational setting: either the proper interactive capabilities are hardcoded in the implementation of the agents' behaviours (thus realizing no decoupling) or it is necessary to adopt off-the-shelf behaviors, which fully implement the specification but that are "injected" and that hide, in an external component, what should be the proactivity of the agent. This is, for instance, the case of JADE.

3. A COMMITMENT-BASED INFRASTRUCTURE FOR SOCIAL RELATIONSHIPS

After providing an overview of the modeling requisites, we describe the corresponding implementation framework, that we named 2COMM. The focus is to provide adequate support for programming social relationships, exploiting a declarative, interaction-centric approach and by relying on existing technologies as far as possible. Specifically, the implementation framework combines two well-known platforms: JADE [Bellifemine et al. 2005] and CArtAgO [Ricci et al. 2011]. 2COMM allows programming complex systems as O-MAS. JADE agents represent stakeholders. The CO-BP, which tie the stakeholders, are defined by using interaction protocols as building blocks. Each interaction protocol is realized as a CArtAgO artifact. Each artifact provides social relationships as environmental resources.

JADE is a popular and industry-adopted agent framework. It offers to developers a Java middleware that is FIPA-compliant [Foundation for Intelligent Physical Agents 2002]. Its robustness and well-proven reliability makes JADE a preferred choice in developing MAS. 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.). JADE provides communication and infrastructure services, allowing agents, deployed in different containers, to dis-

cover and interact with each other, in a transparent way from the developer's logical point of view.

CArtAgO is a framework based on the A&A meta-model [Weyns et al. 2007; Omicini et al. 2008]. 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. It provides a way to define and organize *workspaces*, that are logical groups of artifacts, that can be joined by agents at runtime. The environment is itself programmable and encapsulates services and functionalities. 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 former 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*, which can be equipped with *guards*: conditions that must hold in order for operations to produce their effects. When guards do not hold, actions can still be performed but without consequences.

2COMM is organized as follows: JADE supplies standard agent services (message passing, distributed containers, naming and yellow pages services, agent mobility); when needed, an agent can enact a protocol role, thus using a *communication artifact* – implemented by exploiting CArtAgO, which provides a set of operations by means of which agents participate in a mediated interaction session. Each communication artifact corresponds to a specific protocol enactment and maintains an own social state and an own communication state. Figure 2 reports an excerpt of the 2COMM UML diagram[1]. Let us get into the depths of the implementation.

CommunicationArtifact (CA for short) provides the basic communication operations *in* and *out* for allowing mediated communication. by means of which agents respectively ask to play or to give up playing a role. CA extends an abstract version of the *TupleSpace* CArtAgO artifact: briefly, a blackboard that agents use as a tuple-based coordination means. In and out are, then, operations on the tuple space. CA also traces who is playing which role by using the property *enactedRoles*.

Class *Role* extends the CArtAgO class *Agent*, and contains the basic manipulation logic of CArtAgO artifacts. Thus, any specific role, extending this super-type, will be able to perform operations on artifacts, whenever its player will decide to do so. Role provides static methods for creating artifacts and for *enacting/deacting* roles. This is done by passing a reference to the JADE agent behavior that will actually play the role. The class *CARole* is an inner class of CA and extends the Role class. It provides the *send* and *receive* primitives, by which agents can exchange messages. Send and receive are implemented based on the *in* and *out* primitives provided by CA.

ProtocolArtifact (PA for short) extends CA and allows modeling the social layer with the help of commitments. It maintains the state of the on-going protocol interaction, via the property *socialState*, a store of social facts and commitments, that is managed only by its container artifact. This artifact implements the operations needed to manage commitments (create, discharge, cancel, release, assign, delegate). PA realizes the commitment life-cycle and for the assertion/retraction of facts. Operations on commitments are realized as *internal operations*, that is, they are not invokable directly: the protocol social actions will use them as primitives to modify the social state. Being an extension of CA, PA maintains two levels of interaction: the social one (based on commitments), and the communication one (based on message exchange). The class

[1]The source files of the system and examples are available at the URL http://di.unito.it/2COMM.

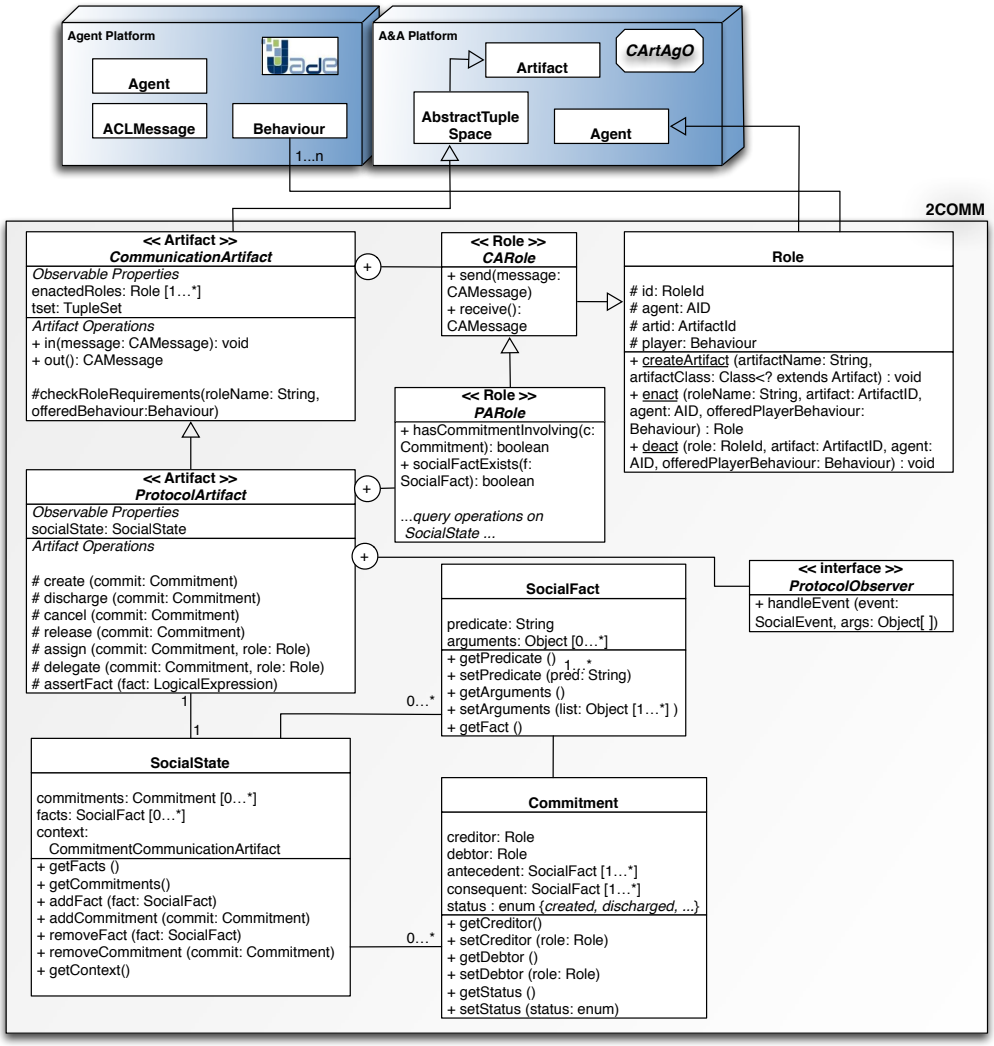

Fig. 2. UML Architecture of 2COMM.

PARole is an inner class of PA and extends the CARole class. It provides the primitives for *querying the social state*, e.g. for asking the commitments in which a certain agent is involved, and the primitives that allow an agent to become, through its role, an *observer of the events* occurring in the social state. For example, an agent can query the social state to verify if it contains a commitment with a specific condition as consequent, via the method `existsCommitmentWithConsequent(InteractionStateElement el)`. Alternatively, an agent can be notified about the occurrence of a social event, provided that it implements the inner interface *ProtocolObserver*. Afterwards, it can start observing the social state. PARole also inherits the communication primitives defined in CARole.

In order to specify a *commitment-based interaction protocol*, it is necessary to extend PA by defining the proper social and communicative actions as operations on the artifact itself. Actions can have guards that correspond to *context preconditions*: each such condition specifies the context in which the respective action produces the described social effect. Since we want agents to act on artifacts only through their respective roles, when defining a protocol it is also necessary to create the roles. We do so by creating as many extensions of PARole as protocol roles. These extensions are realized as inner classes of the protocol: each such class will specify, as methods, the powers of a role. Powers allow agents who play roles to actually execute artifact operations.

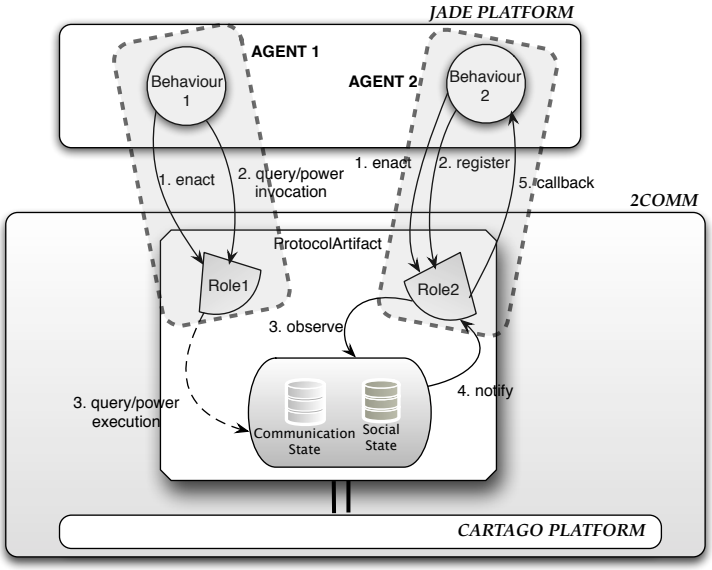

Fig. 3. 2COMM at runtime.

Let us, now, consider Figure 3, which sketches the relationships between the various components of the platform at run-time. When a JADE agent plays a role, its behavior will use the powers offered by the role itself in order to act upon the social and communicative layers, offered by the artifact, which implements the interaction protocol. In this context, the agent behavior is extended with the powers of the role without injecting specific code into it and without adding to it new behaviors. The agent will be free to decide if and when using its new powers by using its own business logic. The business logic is programmed by exploiting both knowledge which is internal to the agent, and that is represented and managed according to the agent model that is used, as well as the social relations, that are reified in the social state of the communication artifact. This can be done either by (Agent 1 in the figure) exploiting the *query powers*, supplied by the role and inherited from PARole, or (Agent 2 in the figure) by relying on a *social event-driven approach*. In this latter case, the role registers as an observer of all events occurring in the social state (including all operations on commitments). When the interaction protocol notifies an event to the role, this calls back the behavior of its player, passing the occurred event. The event handler, defined in the behavior,

will manage the event. Whatever the model, the agent will plan its course of action, according to its own goals. The commitments that the agent must fulfill provide a guide for taking decisions.

4. PROGRAMMING SOCIAL RELATIONSHIPS

We, now, briefly describe *FinancialMAS* as an example of application of 2COMM. This application is inspired by [Parliament 2004], which was a benchmark of the ICT4Law (http://www.ict4law.org) project and was used in [Baldoni et al. 2013c]. FinancialMAS allows financial interactions between three categories of stakeholders: (1) investors, (2) financial promoters and (3) banks. Each stakeholder has different goals: for investors, to place an investment; for promoters, to propose and finalize an investment contract, on behalf of the bank and on the basis of the investor's profile; for banks, to conclude contracts placed by their promoters.

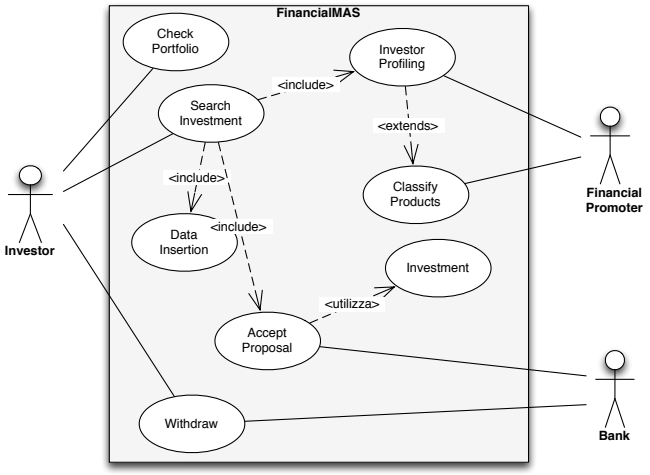

Fig. 4. The FinancialMAS Use Cases.

Figure 4 reports a part of the UseCase Diagram of the system, pointing out the macro-functionalities the system should offer and the involved stakeholders. Starting from this, we individuated a collection of agent types and environment artifacts. Each agent has *responsabilities*, in the specific case, tasks it should accomplish (see Table I), and related *social relationships*. Table II reports the *Interaction Table* for the domain. It contains the "patterns of interactions" that allow performing the various tasks, and that we realize as interaction protocol artifacts. The protocols that are used to shape CO-BP are standard FIPA protocols: we implement them as *protocol artifacts*.

Let us focus on the Contract Net Protocol (CNP for short) [Foundation for Intelligent Physical Agents 2002], which is used inside FinancialMAS to manage the interaction between the Investor and the Financial Promoter when the former looks for some kind of investment (Table II). We adopt the following CNP formulation[2]:

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

[2]Alternatives and variants can be found in [Yolum 2007; El-Menshawy et al. 2010].

Table I. 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 II. 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.

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*

where i stands for the role *Initiator* and p for *Participant*. The instance used by FinancialMAS will tie the Investor to the i and the Financial Promoter to p. Initiator supplies its player the powers *cfp*, *accept*, and *reject*, while Participant supplies its player the powers *propose*, *refuse*, *done*, and *failure*. Powers allow agents to affect the social state. For instance, when an agent playing the role Initiator executes *cfp*, the social state is modified by creating the commitment *C(i, p, propose, accept* $\vee$ *reject)*. This addition binds i to either accept or reject a proposal, if one is received. The agent is free to decide not only which course of action to take but also how to realize acceptance or rejection.

Let us see how CNP is realized in 2COMM. The class CNP extends ProtocolArtifact and implements as artifact operations all the protocol actions. In the sketch below we detail, for the sake of brevity, only the action *cfp*. Lines 4–8 represent the construction and sending of a call for proposals, which will be inserted in the blackboard and made available to participants (mediated communication). The recipient of the message is

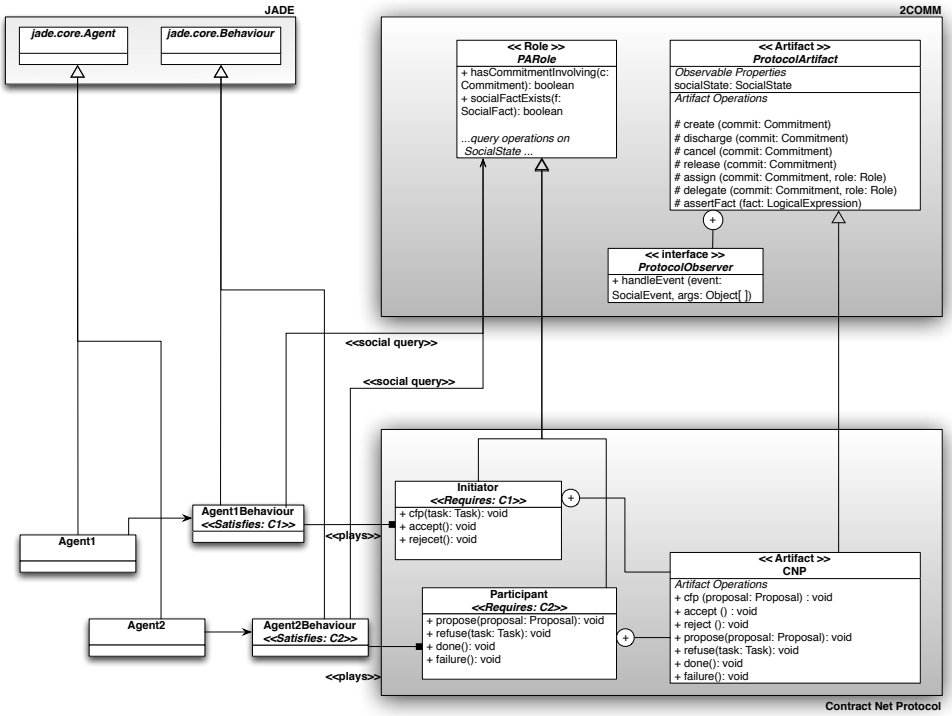

Fig. 5. CNP in 2COMM.

left generic because this is an O-MAS, and the Initiator may not know his peers, who may join even later in the future. Lines 9–11 modify the social state by adding a commitment and by stating that the event *cfp* has occurred. Lines 16–22 defines the role in terms of its powers. Here (lines 18–20), in fact, the power *cfp* is defined in terms of the homonym artifact operation.

```
1  [...]
2  @OPERATION
3  public void cfp(Task task, RoleId initiator) {
4    RoleMessage cfp = new RoleMessage();
5    RoleId dest = new RoleId(PARTICIPANT_ROLE, RoleId.GENERIC_ROLE);
6    cfp.setContent(task);  cfp.setRoleSender(initiator);
7    cfp.setRoleReceiver(dest);  cfp.setPerformative(ACLMessage.CFP);
8    send(cfp);
9    createCommitment(new Commitment(initiator, dest, "propose",
10     new CompositeExpression(LogicalOperatorType.OR,
11       new Fact("accept"), new Fact("reject"))));
12   assertFact(new Fact("cfp", initiator, task));
13 }
14 public void propose(Proposal prop, RoleId participant, RoleId initiator) { [...] }
15 [...]
16 public class Initiator extends PARole {
17   [...]
18   public void cfp(Task task) {
19     doAction(this.getArtifactId(), new Op("cfp", task, getRoleId()));
20   }
21   [...]
22 }
23 public class Participant extends PARole {
24   public void propose(Proposal proposal, RoleId proposalSender) { [...] }
```

```
25      [...]
26    }
27  }
```

The following, instead, is a sketch of the definition of a JADE agent, playing the role Initiator by using 2COMM. In this example, the artifact is created by the agent which, then, enacts the role of interest. At line 34, the agent uses the power *cfp* through the role it plays. Finally, at line 36, it inspects if someone committed to accomplish the task, and reads the proposals in the tuple space in order to make a choice. Figure 5 reports the complete UML diagram for the implementation.

```
28    [...]
29    public class InitiatorBehaviour extends OneShotBehaviour {
30      [...]
31      public void action() {
32        ArtifactId art = Role.createArtifact(ARTIFACT_NAME, CNPArtifact.class);
33        initiator = (Initiator)(Role.enact(CNPArtifact.INITIATOR_ROLE, art, this, myAgent.getAID()));
34        initiator.cfp(t);
35        [...]
36        boolean proposalPerformed = initiator.existsCommitmentWithConsequent(new CompositeExpression(
37          LogicalOperatorType.OR, new Fact("done"), new Fact("failure"));
38        [...]
39      }
40    }
41  }
```

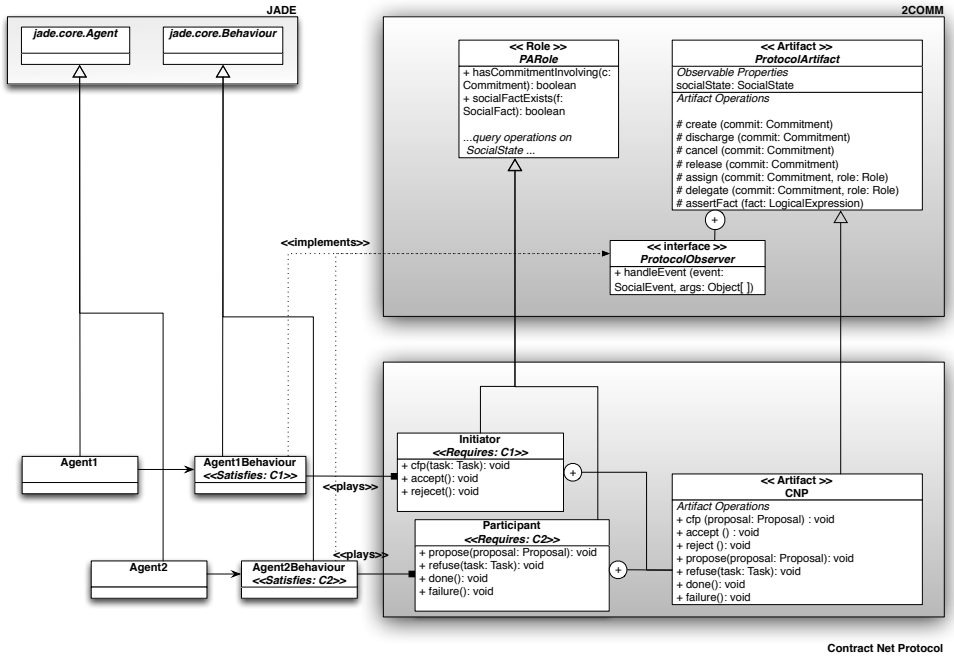

Fig. 6. CNP with 2COMM and support for event handling.

Another way for inspecting social states can be realized by exploiting a feature of 2COMM, allowing roles to register for social state event notification. After line 8, all events occurring in the social state are notified to the role Initiator, which will handle

them by executing *handleEvent* after a callback. In this case, the agent's action contains only the invocation of the power *cfp*, all the others are inside *handleEvent* (e.g. line 17 where the first proposal is accepted). The whole process resembles an event-driven program. Figure 6 contains the UML diagram for this solution.

```
1  public class InitiatorAgent extends Agent {
2    [..]
3    public class InitiatorBehaviour extends OneShotBehaviour {
4      [...]
5      public void action() {
6        ArtifactId art = Role.createArtifact(ARTIFACT_NAME, CNPArtifact.class);
7        initiator = (Initiator) (Role.enact(CNPArtifact.INITIATOR_ROLE, art, this, myAgent.getAID()));
8        initiator.startObserving(this);
9        initiator.cfp(t);
10     }
11     public void handleEvent(SocialEvent e, Object... args) {
12       if (e.getElementChanged().getElType() == interactionStateElementType.LOGICAL_EXPRESSION
13           && e.getElementChanged().toString().contains("propose")) {
14         RoleMessage cfp = initiator.receive(ACLMessage.PROPOSE);
15         Proposal prop = (Proposal) (cfp.getContents());
16         [...]
17         initiator.accept(proposals.get(0));
18         [...]
19       } else [...]
20     }
21   }
22 }
```

More sophisticate forms of event handling may be implemented by writing a handler for each kind of event and by introducing a dispatcher in the artifact, which overrides the default one.

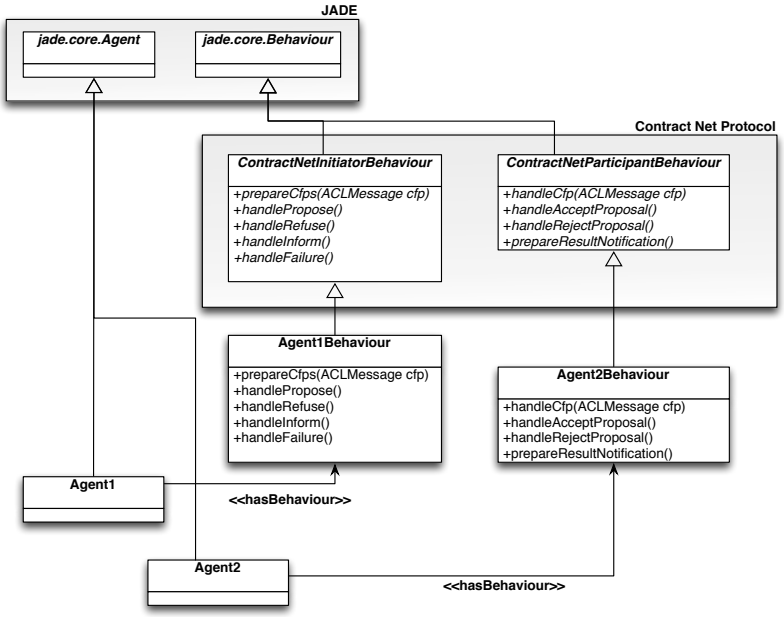

Fig. 7. UML diagram for the JADE implementation of CNP.

Figure 7 shows the UML diagram of an implementation of CNP as it would be done in JADE. A drawback of this approach is that to take part to an interaction ruled by

a protocol, a JADE agent must adopt an entire behavior, corresponding to the played role. As a consequence, the chance of conflict between behaviors is increased, as well as the overall agent design complexity. More importantly, the autonomy of the agent is compromised, unless the programmer writes the behavior from scratch instead of relying on off-the-shelf libraries. Another limit is that JADE does not offer run-time monitoring functionalities. On the whole, such characteristics do not allow JADE to satsfy the principles of STS.

5. CONCLUSIONS

STS are a challenging frontier of today systems realization, answering to a specific need of the contemporary society [Sommerville 2010], where independent and autonomous stakeholders, each with its own objectives, interact through the use of software systems – more and more often built on top and supervising resources, as diverse as application systems, mobile devices, sensors, or even domotic systems. Stakeholders are proactive and characterized by an own decisional capability, but at the same time they have an interest in that the system keeps on working, independently from the fact that the stakeholders use it for competing or for co-operating. We agree with [Singh 2013] that in order to implement STS, it is necessary to realize some kind of *self-governance*, i.e. to realize normative systems, whose actors accept a set of regulations which will guide their behavior. Although MAS supply the means for supporting distributed systems involving many actors, none of the implemented infrastructures currently allows the realization of self-governance.

In this work, we propose 2COMM, an infrastructure for achieving this very objective. 2COMM is based on JADE, for what concerns the support to the realization of interacting agents, and integrates self-governance mechanisms by relying on the reification of commitments and of commitment-based protocols. These are, at all respects, resources that are made available to stakeholders and that are realized by means of artifacts. The article also shows how to pass from a CO-BP to a set of reified protocols, representing it and that will enable the interaction. This is done based on a variant of the JADE Methodology [Nikraz et al. 2006]. 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 [Zambonelli et al. 2003]).

Commitments give a normative value to the encoded protocol, while the act of using an artifact that encodes a commitment protocol amounts to the explicit acceptance, by the agent, of the rules of the protocol. The proposal is characterized by the flexibility and the openness that are typical of MAS with the modularity and compositionality that are typical requirements of the methodologies for design and development. The realization of commitment-based protocols as artifacts implements the vision of middleware, proposed in [Chopra and Singh 2009], and represents an advancement also of research on commitment-based approaches.

The approach to the development of CO-BP followed in 2COMM avoids a critical facet of JADE protocols: here, a pattern of interaction is projected into a set of JADE behaviors, one for each role. Binding the very interaction to ad hoc behaviors does not allow having a global view of the protocol and complicates its maintenance. Thereby, the risk of raising conflicts, not devised at design time, increases.

This work has also relationships with works concerning e-institutions, like [Fornara et al. 2007; Fornara et al. 2008; Okouya et al. 2013]. E-institutions introduce a normative aspect that lies on a different level with respect to the one we capture with commitments. Indeed, in our proposal the normative layer stands in the observational semantics of the social actions. We mean to extend the regulative aspects, as a first step, by enriching commitment protocols as proposed in [Baldoni et al. 2013b; Marengo

et al. 2011]. More in general, we mean to move towards a representation of business processes as suggested in the Comma methodology [Telang and Singh 2012], where commitment patterns regulate activities instead of being used for giving the action semantics. The advantage of our proposal is that it allows the direct use of the well-known JADE Methodology in the development of STS [Baldoni et al. 2013a]. Another advantage is that artifacts are an actual system for implementing mediated interaction, they supply efficient, simple, and direct means for checking the powers for performing institutional actions, and they offer runtime mechanisms for letting the agents perceive the state of the interaction.

Concerning organizations, ORA4MAS [Hubner et al. 2009] and JaCaMo are examples of integrations with artifacts. The latter also accounts for BDI agents. The organizational infrastructure of ORA4MAS is based on $Moise^+$. It allows the enforcement and the regimentation of regulations, by defining both a set of conditions to achieve and the roles that are allowed to or that must achieve them. The limit of this approach is that it cannot manage norms that cannot be restricted to goal achievement. Recently, the use of a communication infrastructure based on artifacts has been proposed to define, in an explicit and clear way, interaction in JaCaMo [Rodrigues et al. 2013]. Nevertheless, the proposal does not supply a normative account of communication.

Finally, concerning the realization of CO-BP and, in particular, of human-oriented workflows, whose nature is intrinsically social and where the notion of commitment plays a fundamental role [Medina-Mora et al. 1993], it is relevant to cite LoST [Singh 2011]: a commitment-based model for defining declarative protocols. It exploits vectors, each associated to one of the interacting agents, containing the local history of the sent/received messages. LoST enables the representation and monitoring of 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.

REFERENCES

Matteo Baldoni, Cristina Baroglio, and Federico Capuzzimati. 2013a. 2COMM: a commitment-based MAS architecture. In *Proc. of the 1st Int. Workshop on Engineering Multi-Agent Systems, EMAS 2013*. St. Paul, Minnesota, USA, 17–32.

Matteo Baldoni, Cristina Baroglio, Elisa Marengo, and Viviana Patti. 2013b. Constitutive and Regulative Specifications of Commitment Protocols: a Decoupled Approach. *ACM Transactions on Intelligent Systems and Technology, Special Issue on Agent Communication* 4, 2 (March 2013), 22:1–22:25.

Matteo Baldoni, Cristina Baroglio, Elisa Marengo, Viviana Patti, and Federico Capuzzimati. 2013c. Engineering commitment-based business protocols with the 2CL methodology. *Autonomous Agents and Multi-Agent Systems*. To appear.

Matteo Baldoni, Cristina Baroglio, Elisa Marengo, Viviana Patti, and Alessandro Ricci. 2011. Back to the future: An interaction-oriented framework for social computing. In *First Int. Workshop on Req. Eng. for Social Computing, RESC*. IEEE, 2–5.

Matteo Baldoni, Cristina Baroglio, Viviana Mascardi, Andrea Omicini, and Paolo Torroni. 2010. Agents, Multi-Agent Systems and Declarative Programming: What, When, Where, Why, Who, How?. In *25 Years GULP (Lecture Notes in Computer Science)*, Vol. 6125. Springer, 204–230.

Matteo Baldoni, Guido Boella, and Leendert van der Torre. 2006. Interaction among objects via roles: sessions and affordances in Java. In *PPPJ (ACM International Conference Proceeding Series)*, Ralf Gitzel, Markus Aleksy, and Martin Schader (Eds.), Vol. 178. ACM, 188–193.

Matteo Baldoni, Guido Boella, and Leendert van der Torre. 2007. Interaction between Objects in powerJava. *Journal of Object Technology* 6, 2 (2007).

F. Bellifemine, F. Bergenti, G. Caire, and A. Poggi. 2005. JADE - A Java Agent Development Framework. In *Multi-Agent Programming: Languages, Platforms and Applications*. Multiagent Systems, Artificial Societies, and Simulated Organizations, Vol. 15. Springer, 125–147.

Guido Boella and Leendert W. N. van der Torre. 2007. The ontological properties of social roles in multi-agent systems: definitional dependence, powers and roles playing roles. *Artificial Intelligence and Law* 15, 3 (2007), 201–221.

Rafael H. Bordini, Lars Braubach, Mehdi Dastani, Amal El Fallah-Seghrouchni, Jorge J. Gmez-Sanz, Joo Leite, Gregory M. P. O'Hare, Alexander Pokahr, and Alessandro Ricci. 2006. A Survey of Programming Languages and Platforms for Multi-Agent Systems. *Informatica (Slovenia)* 30, 1 (2006), 33–44.

Frances M. T. Brazier, Barbara M. Dunin-Keplicz, Nick R. Jennings, and Jan Treur. 1997. Desire: Modelling Multi-Agent Systems in a Compositional Formal Framework. *International Journal of Cooperative Information Systems* 06, 01 (March 1997), 67–94.

Paolo Bresciani and Paolo Donzelli. 2003. A Practical Agent-Based Approach to Requirements Engineering for Socio-technical Systems. In *AOIS (Lecture Notes in Computer Science)*, Paolo Giorgini, Brian Henderson-Sellers, and Michael Winikoff (Eds.), Vol. 3030. Springer, 158–173.

Albert Cherns. 1976. Principles of Socio-Technical Design. *Human Relations* 2 (1976), 783–792.

Amit K. Chopra. 2009. *Commitment Alignment: Semantics, Patterns, and Decision Procedures for Distributed Computing*. Ph.D. Dissertation. North Carolina State University, Raleigh, NC.

Amit K. Chopra and Munindar P. Singh. 2009. An Architecture for Multiagent Systems: An Approach Based on Commitments. In *Proc. of ProMAS*.

Rosaria Conte, Cristiano Castelfranchi, and Frank Dignum. 1998. Autonomous Norm Acceptance. In *ATAL (Lecture Notes in Computer Science)*, Vol. 1555. Springer, 99–112.

Fabiano Dalpiaz, Amit K. Chopra, and Soo Ling Lim. 2011. The first International Workshop on Requirements Engineering for Social Computing. In *RESC*. IEEE, 1.

Mehdi Dastani, Davide Grossi, John-Jules Ch. Meyer, and Nick A. M. Tinnemeier. 2008. Normative Multi-agent Programs and Their Logics. In *KRAMAS (Lecture Notes in Computer Science)*, John-Jules Ch. Meyer and Jan Broersen (Eds.), Vol. 5605. Springer, 16–31.

Nirmit Desai, Amit K. Chopra, and Muhindar P. Singh. 2009. Amoeba: A Methodology for Modelling and Evolving Cross-Organizational Business Processes. *ACM Transactions on Software Engineering and Methodology* 19, 2 (2009).

Mohamed El-Menshawy, Jamal Bentahar, and Rachida Dssouli. 2010. Symbolic Model Checking Commitment Protocols Using Reduction.. In *DALT (Lecture Notes in Computer Science)*, Andrea Omicini, Sebastian Sardia, and Wamberto Weber Vasconcelos (Eds.), Vol. 6619. Springer, 185–203.

Michael Fisher, Rafael H. Bordini, Benjamin Hirsch, and Paolo Torroni. 2007. Computational Logics and Agents: A Road Map of Current Technologies and Future Trends. *Computational Intelligence* 23, 1 (2007), 61–91.

Nicoletta Fornara, Francesco Viganò, and Marco Colombetti. 2007. Agent communication and artificial institutions. *Autonomous Agents and Multi-Agent Systems* 14, 2 (2007), 121–142.

Nicoletta Fornara, Francesco Viganò, Mario Verdicchio, and Marco Colombetti. 2008. Artificial institutions: a model of institutional reality for open multiagent systems. *Artif. Intell. Law* 16, 1 (2008), 89–105.

Foundation for Intelligent Physical Agents. 2002. FIPA Specifications. (2002). http://www.fipa.org.

James J. Gibson. 1979. *The Ecological Approach to Visual Perception*. Lawrence Erlbaum Ass., New Jersey.

J. F. Hubner, O. Boissier, R. Kitio, and A. Ricci. 2009. 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).

Elisa Marengo, Matteo Baldoni, Cristina Baroglio, Amit K. Chopra, Viviana Patti, and Munindar P. Singh. 2011. *Commitments with regulations: reasoning about safety and control in REGULA*. International Foundation for Autonomous Agents and Multiagent Systems, 467–474.

Viviana Mascardi, Maurizio Martelli, and Leon Sterling. 2004. Logic-Based Specification Languages for Intelligent Software Agents. *TPLP* 4, 4 (2004), 429–494.

Raul Medina-Mora, Terry Winograd, Rodrigo Flores, and Fernando Flores. 1993. The Action Workflow Approach to Workflow Management Technology. *Inf. Soc.* 9, 4 (1993), 391–404.

Enid Mumford. 2006. The story of socio-technical design: reflections on its successes, failures and potential. *Information Systems Journal* 16 (2006), 317–342.

Magid Nikraz, Giovanni Caire, and Parisa A. Bahri. 2006. A Methodology for the Analysis and Design of Multi-Agent Systems using JADE. May (2006).

Daniel Okouya, Nicoletta Fornara, and Marco Colombetti. 2013. An Infrastructure for the Design and Development of Open Interaction Systems. In *Proc. of the 1st Int. Workshop on Engineering Multi-Agent Systems, EMAS 2013*. St. Paul, Minnesota, USA, 128–143.

Andrea Omicini, Alessandro Ricci, and Mirko Viroli. 2008. Artifacts in the A&A meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems* 17, 3 (2008), 432–456.

Andrea Omicini and Franco Zambonelli. 1998. TuCSoN: a Coordination model for Mobile Information Agents. IDI-TR-5/98. In *1st International Workshop on Innovative Internet Information Systems (IIIS'98)*. IDI – NTNU, Trondheim (Norway), Pisa, Italy, 177–187.

European Parliament. 2004. Directive 2004/39/EC of the European Parliament and of the Council of 21 April 2004 on markets in financial instruments. *Official Journal of the European Union* L145 (2004), 1–44.

Maja Pesic. 2008. *Constraint-Based Workflow Management Systems: Shifting Control to Users*. Ph.D. Dissertation. Eindhoven University of Technology.

Davide Porello, Davide Setti, Roberta Ferrario, and Marco Cristani. 2013. Multiagent Socio-Technical Systems. An Ontological Approach. In *Proc. of the 15th Int. Workshop on Coordination, Organisations, Institutions and Norms (COIN 2013)*. St. Paul, Minnesota, USA, 1–15.

Alessandro Ricci, Michele Piunti, and Mirko Viroli. 2011. Environment programming in multi-agent systems: an artifact-based perspective. *Autonomous Agents and Multi-Agent Systems* 23, 2 (2011), 158–192.

Thiago Fredes Rodrigues, Antônio Carlos da Rocha Costa, and Graçaliz Pereira Dimuro. 2013. A Communication Infrastructure Based on Artifacts for the JaCaMo Platform. In *Proc. of the 1st Int. Workshop on Engineering Multi-Agent Systems, EMAS 2013*. St. Paul, Minnesota, USA, 97–111.

Munindar P. Singh. 1999. An Ontology for Commitments in Multiagent Systems. *Artif. Intell. Law* 7, 1 (1999), 97–113.

Munindar P. Singh. 2000. A Social Semantics for Agent Communication Languages. In *Issues in Agent Communication (LNCS)*, Vol. 1916. Springer, 31–45.

Munindar P. Singh. 2011. LoST: Local State Transfer - An Architectural Style for the Distributed Enactment of Business Protocols.. In *ICWS*. IEEE Computer Society, 57–64.

Munindar P. Singh. 2013. Norms as a Basis for Governing Sociotechnical Systems. *ACM Transactions on Intelligent Systems and Technology (TIST)* in press (2013), 1–21.

Ian Sommerville. 2010. *Software Engineering* (9 ed.). Addison-Wesley, Harlow, England.

Ian Sommerville, Dave Cliff, Radu Calinescu, Justin Keen, Tim Kelly, Marta Z. Kwiatkowska, John A. McDermid, and Richard F. Paige. 2011. Large-scale Complex IT Systems. *CoRR* abs/1109.3444 (2011).

Pankaj R. Telang and Munindar P. Singh. 2010. Abstracting and Applying Business Modeling Patterns from RosettaNet.. In *ICSOC (Lecture Notes in Computer Science)*, Paul P. Maglio, Mathias Weske, Jian Yang, and Marcelo Fantinato (Eds.), Vol. 6470. 426–440.

Pankaj R. Telang and Munindar P. Singh. 2011. Specifying and Verifying Cross-Organizational Business Models: An Agent-Oriented Approach. *IEEE Transactions on Services Computing* (2011), 1–14.

Pankaj R. Telang and Munindar P. Singh. 2012. Comma: a commitment-based business modeling methodology and its empirical evaluation.. In *AAMAS*, Wiebe van der Hoek, Lin Padgham, Vincent Conitzer, and Michael Winikoff (Eds.). IFAAMAS, 1073–1080.

Alexander Thiele, Thomas Konnerth, Silvan Kaiser, Jan Keiser, and Benjamin Hirsch. 2009. Applying JIAC V to Real World Problems: The MAMS Case.. In *MATES* (2009-09-21) *(Lecture Notes in Computer Science)*, Lars Braubach, Wiebe van der Hoek, Paolo Petta, and Alexander Pokahr (Eds.), Vol. 5774. Springer, 268–277.

Eric Trist. 1981. *The Evolution of Socio-technical Systems: A Conceptual Framework and an Action Research Program*.

Danny Weyns, Andrea Omicini, and James Odell. 2007. Environment as a first class abstraction in multiagent systems. *Autonomous Agents and Multi-Agent Systems* 14, 1 (2007), 5–30.

Stephen A. White. 2004. *Introduction to BPMN*. Technical Report.

Brian Whitworth and Adnan Ahmad. 2013. *Socio-Technical System Design*. The Interaction Design Foundation, Aarhus, Denmark.

Pinar Yolum. 2007. Design time analysis of multiagent protocols. *Data Knowledge Engineering* 63, 1 (2007), 137–154.

Pinar Yolum and Munindar P. Singh. 2002. Commitment Machines. In *Intelligent Agents VIII, 8th International Workshop, ATAL 2001 (LNCS)*, Vol. 2333. Springer, 235–247.

Franco Zambonelli, Nicholas R. Jennings, and Michael Wooldridge. 2003. Developing multiagent systems: The Gaia methodology. *ACM Trans. Softw. Eng. Methodol.* 12, 3 (2003), 317–370.

Received ??; revised ??; accepted ??