2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology

Goal-Directed Interactions in Artifact-Based MAS: Jadex Agents playing in CARTAGO Environments

Michele Piunti, Alessandro Ricci
DEIS
Università di Bologna
Sede di Cesena, Italy
{michele.piunti | a.ricci}@unibo.it

Lars Braubach, Alexander Pokahr
Distributed and Information Systems
Computer Science Department
University of Hamburg, Germany.
{braubach | pokahr}@informatik.uni-hamburg.de

Abstract

In the context of cognitive agent programming frameworks, a main research effort accounts for exploiting goal-orientation for specifying and enacting agent interaction. Existing research work focuses almost totally on direct communication models, typically based on speech-acts and FIPA ACL. In this paper we focus instead on mediated interactions, and in particular on interaction taking place in artifact-based environments, where artifacts are first-class mediating tools that are used by cognitive agents in goal-directed way. The investigation is concretely supported by integrating the Jadex platform (enrolling the Belief-Desire-Intention model of agency) with the CARTAGO technology (enabling the design of artifact based environments).

1. Introduction

Interaction is a primary aspect of Multi-Agent Systems (MAS), and interaction models and mechanisms are fundamental elements in the design and programming of any non-naive MAS application. In the context of agent programming frameworks, interaction models are typically designed on the basis of direct communication based on speech-act, protocols and conversations: a main research effort here aims at exploiting cognitive models and agent's goal-orientation for specifying interaction protocols (see [2] for a survey). This made it possible for developers to focus on aspects related to the problem domain, more than on the specification of lower level details of message passing. Besides direct speech-based communication, recent research works remarked the value of *environment-mediated* interaction models. This approach promotes the environment abstraction to be exploited as a suitable locus to encapsulate services and functionalities. Accordingly it has been assumed to improve particular agent interaction, coordination, and cooperation activities (see [19] for a survey). Among the various approaches, the "Agents and Artifacts" model (A&A) introduces first-class abstractions called *artifacts* to design and program those *functional* parts of a MAS in terms of workspaces with computational resources and tools that agents can share and co-use to attain their goals. Analogously to artifacts in human cooperative environments, A&A artifacts are conceived to support cognitive agents in their individual and collective tasks [15, 12]. From a software engineering perspective, the A&A provides a general methodology to structure and program the agent computational environment inside the MAS, and then—thinking in particular to *coordination artifacts*—the agent interaction space besides message based interaction protocols.

The aim of this paper is to extend the issue of goal-oriented interaction, as devised for protocol based message exchange, to include also mediated kinds of interaction in artifact-based environments. In this respect, artifacts can be seen by agents as middleware facilities, allowing various forms of mediated or supervised interaction, and representing an alternative mean to extend agent communication and cooperation besides direct message-based protocols. In particular, this paper investigates goal-directed use of artifacts for mediated interactions in MAS, assumed as open working environments where heterogeneous agents can share activities and interact through the same artifacts. For this purpose different kinds of artifacts use will be sketched. We thus describe the integration of *Jadex*, a programming platform for BDI agents [13], and CARTAGO, an infrastructure providing a programming model and executable framework for building and running A&A environments [16].

By adopting the notion of goal directedness, we place here an important distinction between actual approaches in cognitive agent design. In fact, abilities to handle goals traditionally are variously characterized by agent platforms using procedural or declarative goals [17]. Differently from other BDI inspired systems which have already been integrated in CARTAGO [14], *Jadex* makes us possible to improve agent/artifact interaction at an higher cognitive level.

DOI 10.1109/WIIAT.2008.349

As showed in Section 3, *Jadex* declarative goals are processed upon an event based execution model allowing more flexible form of deliberation and reasoning [4]. Besides, *Jadex* pivotal reasoning processes as goal adoption and plan selection can be governed by customizable internal events, which in turns can be targeted on the basis of the events coming from artifacts. Finally, *Jadex* adopted the notion of capability as external and reusable components programmable in goal directed fashion [3], thus allowing the developer to focus on domain objectives and scale complex interactions details. The programming model has been discussed through a contract-net based negotiation where agents interactions are mediated by artifacts, analogously to the approach based on direct communication described in [2].

2. Artifact-Based Environments

One of the main concepts put forward by Activity Theory [10] –along with Distributed Cognition and other movements inside cognitive science– is that artifacts and tools, in human societies, play a pivotal role in coping with the scaling up of complexity, in particular when social activities are concerned. Inspired by Activity Theory, the A&A approach introduces the notion of *artifact* as first-class abstraction along with agents in MAS, to model and design those resources and tools that are eventually used by agents to perform their activities and cooperate. Examples in MAS range from blackboards, task schedulers, calendars – as kinds of coordination artifacts– to shared knowledge bases, maps, device wrappers as kinds of informational resources. Differently from agents, which in their strong notion are autonomous and goal-oriented entities, artifacts are non-autonomous, function-oriented entities, that can be used by agents though suitable *usage interface* that control their behaviour.

The set of artifacts in a MAS are organised in terms of *workspaces*, defining the topology of the work environment where agents are logically situated and works creating, sharing, using, and manipulating artifacts as first-class environment building blocks. The next sections briefly describe basic notions of CARTAGO, a framework to develop MAS based on A&A notions. More on A&A as a meta-model can be found in [11, 15].

2.1. A Programming Model and Infrastructure for Artifacts: CARTAGO

CARTAGO (CommonARtifact Infrastructure for AGent Open environment) is a framework/infrastructure based on A&A meta-model, providing a concrete programming model and (Java-based) technology[1] for building and running artifact-based environments in MAS [16]. Essentially, CARTAGO provides: *(a)* a basic API to program artifacts according to a programming model briefly accounted in next section; *(b)* a basic set of API to be used on the agent side to work within artifact-based environments, with actions for using, observing, manipulating artifacts; *(c)* a runtime distributed (Java-based) system to execute and manage artifact-based environments.

A foremost important aspect of CARTAGO is its orthogonality with respect to agent models and platforms. Accordingly, the technology has been designed to be integrated with existing agent platforms, so to enable the development of open MAS with agents belonging to heterogeneous platforms working together within common artifact-based environments [14].

The integration is based on the notion of *agent body* conceived as that part of an agent conceptually belonging to a workspace (once the agent is inside it). Agent body contains logical sensory-motor capabilities to interact with artifacts (see Figure 1), and which is controlled by the agent mind, programmed and executed on top of some external agent platform. By following a cognitive approach, agent body is assumed to provide agents with actuators to execute actions (i.e., to manage workspaces or act upon (*use*) artifacts) and sensors to perceive noticeable events (i.e., to proactively keep track of observable events generated by artifacts, possibly applying filters and specific kinds of "buffering" policies). According to the specific interaction modality adopted for using and observing artifacts, agents are provided with basic internal actions for managing and inspecting sensors residing in their body, as a kind of external working memory. In doing so, an agent can organize in a flexible way the perception of interleaved observable events, possibly generated by multiple artifacts that the agent is using for different, even concurrent, activities.

2.2. Artifact Based Contract-Net

As an example clarifying the artifact programming model in CARTAGO, we describe here a book-trading scenario based on contract-net analogous to the one described in [2]: here – differently from original specification where interactions are message based [8]– interactions between buyer and seller agents have been conceived as mediated by artifacts coordinating agent activities[2].

To this end, a contract-net workspace has been conceived to constitute the basic working environment of the system. Two instances of artifacts have been deployed in the contract-net workspace: `BuyerBoard` (instrumental to buyer activities) and `SellerBoard` (instrumental to sellers activities). They are assumed *(i)* to mediate agent-agent interactions and *(ii)* to provide agents with additional means to attain their goals. Following the CARTAGO programming model, artifact functionalities are structured in terms of *operations*,

[1]CARTAGO is an open source project, available at `http://cartago.sourceforge.org`.

[2]The example is assumed to provide a case study to enlighten the programming model behind *Jadex*-CARTAGO interactions. A detailed discussion on the contract-net problem is beyond the scope of this paper.

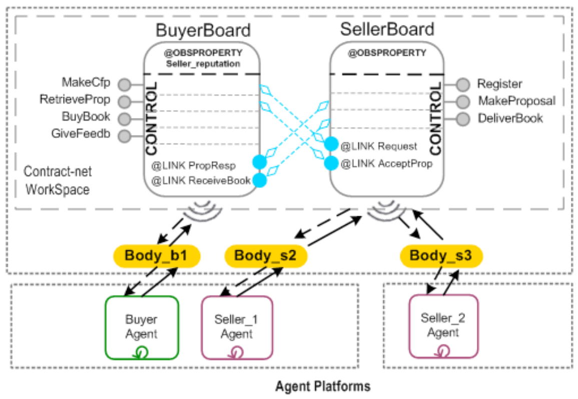

Figure 1. Contract-net workspace with BuyerBoard and SellerBoard artifacts, with in evidence usage and link interfaces.

whose execution can be triggered/invoked by agents by acting on their usage interface. The usage interface is composed by a set of *operation controls*, each identified by a label (typically equals to the operation name) and a list of input parameters (in Figure 1, parameters are omitted for brevity). `BuyerBoard` exposes a `MakeCfp` operation to initiate a call for proposal (*cfp*), `RetrieveProp` to retrieve proposals for a given *cfp* `BuyBook`, to purchase an item from a given seller, and `GiveFeedb`, by which the buyer can provide a reputation feedback about quality of service. On the other side, `SellerBoard` provides a usage interface with `Register`, by which a seller can register to the contract-net, `MakeProposal`, to make a sale proposal for a given book request, and `DeliverBook`, to finally deliver the purchased book. The execution of an operation upon an artifact can result both in changing the artifact's inner (i.e. non-observable) state, and in the generation of a stream of *observable events* that can be perceived by agents through their perceptive activities. Observable events rise as the result of an operation usage. They convey non-persistent information and signals carrying some additional information content in form of agent readable *percepts*. Interested readers can find elsewhere the detailed model related to operation triggering and execution, in particular with respect to synchronisation and concurrency [15].

Besides controls, the usage interface might contain also a set of *observable properties*, as persistent symbolic properties whose dynamic values can be intentionally read by agents without practically operating upon the artifact controls. `BuyerBoard`, for instance, contains an observable property about the seller's reputation (in this case, reputation collects a simple list of values in [0,1], which can be related to the various sellers). Additional observable events percievable by agents are supposed to signal changes in observable properties.

Finally, artifacts can be linked together (by agents, in particular) so as to enable inter-artifact interactions and artifact compositionality. For this purpose, an artifact may expose – besides the usage interface– a *link interface*, including those operations that can be triggered solely by artifacts that have been linked to it. In the contract-net example, artifacts are linked in order to maintain a consistent view of the problem domain for their users: once a buyer initiates a *cfp* (i.e. by using a `MakeCfp` upon his board) the links guarantee changes to be propagated to the `SellerBoard` –and thus consistently retrieved by the sellers– by triggering the `Request` link operation (see Figure 1).

3. Goal-Directed Use of Artifacts

In this section interaction approaches for artifacts and goal directed agents will be discussed. A non invasive integration will be detailed for the *Jadex* platform.

3.1. BDI Programming Model in *Jadex*

Originally proposed as a conceptual model to explain cognitive behavior [1], the Belief-Desire-Intention model assumes that rational actions can be derived from desires (motivational states), beliefs (epistemic states) and intentions (practical activities the agent is committing to attain goals). The process for deducing actions given these mental attitudes is called *practical reasoning* and consists of two different phases [21]. The *goal deliberation* phase has the purpose to select a consistent subset of goals for further processing, while the *means-end reasoning* phase is responsible for finding ways to achieve the adopted goals.

Using technologies like XML and Java, *Jadex*[3] natively faces with BDI notions such as plans, beliefs and goals. The *Jadex* execution model uncouples goal processing from plan execution. It assumes that declarative goals are defined to express the world states to achieve, while plans contain the procedural code for their realization. At runtime an agent can possess an alternating hierarchy of goals and plans, which emerge from top-level goals over plans to subgoals and so forth.

We guess using goals as an explicit part of the programming model has key advantages. First, differently from plans which in turn constitute the agent's reactive abilities, goals have a proper processing ruled by deliberation mechanisms. This on the one side promotes a more flexible deliberation mechanism and, on the other side, allows to scale up means-end processes (as intention reconsideration and action selection) allowing agents to be situated and targeted in the context of goal processing. Then, means-end reasoning phases introduce a flexible plan-based machinery for achieving goals. For any given goal description, a list of applicable plans

[3]*Jadex* is an open source project for implementing BDI agents available at: `http://vsis-www.informatik.uni-hamburg.de/projects/jadex/`.

is searched and ranked according to a customizable policy. Then the highest ranked plan is executed and if the goal could not be attained by this plan the process starts over again and looks for the next plan to execute until no more options (plans) are available (see [4, 13] for more details).

Second, since *Jadex* goals are declarative they can be compared, evaluated, appraised and eventually updated upon their completion [4]. Given goal representations, agents can handle on-line gloal processing, for instance appraising mismatches and possibly introducing recovery and learning mechanisms. In these terms, *Jadex* goals can be exploited as *mental states* and used to reason in anticipatory terms (i.e. making decisions not only on the basis of what is believed but also on what is desired to be in the future).

3.2. Introducing *Jadex* Capabilities for Agent-Artifact Interactions

Following a cognitive approach, we consider basic sensory-motor aspects to allow *Jadex* agents to play in CARTAGO environments. On the one hand, agents need mechanisms to practically interact with artifacts (*actuators*); on the other hand they have to be enabled to perceive noticeable events generated by artifacts (*sensors*). In this view, the integration approach has been realized at the language level. The set of artifact-related actions –i.e., *use* and *sense*– have been added to the repertoire of natively available actions. In addition, the external events coming from artifact have been integrated at an architectural level by automatically promoting such events as “relevant” signals to be addressed to the deliberation cycle.

On these basis, two different integration levels are envisaged. At a first level, a basic usage of artifacts is targeted by allowing agents to get access to artifacts and their operations. In this view, the usage should be facilitated for the developer in such a way that it naturally fits to existing programming metaphors and make use of the concepts of the BDI canon. This is achieved via a **general purpose module**, which exposes basic operations in a goal-directed fashion. In addition, the perceptive abilities are encapsuled inside the module and are based on the propagation of internal events, which in turn can be signalled outside of the module within the agent’s reasoning process.

At a second level, a more domain oriented integration is envisioned, in order to provide specific support for using complex artifact types, i.e. requiring to be used in a sequence of steps. This approach tackles this problem by associating the interaction logic for specific artifact types into domain specific, **special purpose modules**, assumed to scale up the complexity of artifact observation and usage. For implementing modules as described above, we adopted the notion of *capability* [3]. Originally proposed in [6] in the context of BDI agents, the notion of capability is intended to functionally group related resources (i.e. beliefs, goals, plans) and to exhibit mechanisms to make them available to agents as

(1)	`joinWorkspace(WName,-Node)`
(2)	`moveToWorkspace(WName,-Node)`
(3)	`quitWorkspace(WName)`
(4)	`use(AName,UICntrlName(Params),?SName,?Timeout,?Filter)`
(5)	`sense(SName,?Filter,?Timeout,+Perception)`
(6)	`focus(AName,?SName,?Filter)`
(7)	`stopFocussing(AName)`
(8)	`observeProperty(AName,?SName,?PFilter,+Property)`

Table 1. Basic set of actions to interact with CARTAGO environments.

reusable components.

3.3. General Purpose Capability

The *Jadex*- CARTAGO bridge component is realized as a *Jadex* capability encapsulating the required functionalities to operate with artifacts. As a basic building block, the capability provides a one-to-one mapping of CARTAGO basic actions to agent goals and plans conceived to interact. Table 1 provides a synthetic view of the basic actions implemented by the capability grouped into four main groups. In particular, the available basic actions make it possible for an agent to: join, move, and leave workspaces (1-3); use an artifact by acting on its control interface and perceive generated events (4-7); observe artifact properties (8). In Table 1 a pseudo-code first-order logic-like syntax is adopted, while semantics is described informally: `?` indicates optional parameters, while `-`/`+` in/out parameters. Each of the basic CARTAGO actions is realized in the capability by a self-contained goal and a related plan. Following the semantics adopted in the agent-oriented programming approaches considered here, an action consists in the atomic execution of a statement which can result in changing the agent’s state and/or interacting with the agent’s environment, and can succeed or fail. Besides, the *Jadex* model allowed the explicit and persistent representation of relations between agents and environments using special types of goals. For instance, the `use` action is modeled as an achieve goal, because it may fail due to a timeout (e.g., if the artifact is currently in use by another agent); the `sense` action is modeled as a query goal, thereby indicating that the action is used to attain information (from the artifact, in this case) and succeeds when the information has been obtained.

In addition to the basic actions, some common usage patterns have been identified, that combine typical A&A interactions and can be implemented as a combination of abstract BDI notions. For instance, instead of the programmer having to explicitly perform sense operations, the capability can enable a continuous ’background-sensing’ and allow programming agent behavior that reacts to changes in focused artifacts. There could be at least two models for realizing this: on the one hand observable events coming from artifacts could be stored in agent’s belief base. These beliefs would then be updated automatically (when some relevant artifact property changes or when a percept is signalled) and could elicit the agent to enter in the deliberation phase. On the other hand, observable events could automatically be posted to the agent in form of internal events. Using these events as triggers, the developer can use the events to reactively trigger plans. An example of this model will be shown later in section Section 4.

3.4. Special Purpose Capabilities

Because A&A interactions can be arbitrarily complex, i.e. in dynamic environments with presence of concurrent agents and shared artifacts, a developer may need to better focus on agents in terms of problem solving, thus evaluating the domain space at an higher level of abstraction. An alternative A&A interaction approach may thus suggest to provide terminal goals and related coarse grained plans, allowing agents to interact with artifacts at an higher level of abstraction, in a tight goal directed fashion. This approach borrows the hierarchical abstraction from cognitive systems and AI, where hierarchical planning is traditionally used to reduce a composite activity or a given task to a greater number of independent sub-activities or tasks placed at a lower level of abstraction. This eases the usage of artifacts and shifts low level details of agents/artifact interactions into the mechanisms embedded in the capability. The special-purpose capability approach can thus be defined having in mind the problem domain (the goal to be achieved) and then used to encapsulate fine grained operations which in turn may transparently control the interaction from agents to artifacts. Instead of offering only the basic operations to the user (i.e. initiating a purchasing activity making a request for a given book, retrieve the sellers' proposals and select the best one), also aggregated functions can thus be included, which can be described at the domain objective one wants to achieve, e.g. to buy a given book. In so doing, plans are assumed to possibly handle low level implementation details (i.e. events, exceptions, fails etc.) hence improving the goal directed approach to artifacts. In addition, one could develop modules which offer aggregated functionalities which may be assembled since multiple artifact types, by including the corresponding rules for their intertwined usage. The realization of this integration level is envisaged via a goal directed approach by which an agent can simply adopt a goal and wait for its completion.

4. Artifact based Contract-net

This section discusses the usage of artifacts within the contract-net scenario introduced in Subsection 2.2. The example has been chosen in order to compare the approach with the one described in [2], in which the same scenario was used and a special 'Protocol' capability was introduced to handle message passing between agents. Whereas the Protocol capability is aimed at hiding low-level programming details to the developer, here we place the problem in terms of A&A mediated interactions and describe solution sketches using the generic and special purpose capability approaches[4].

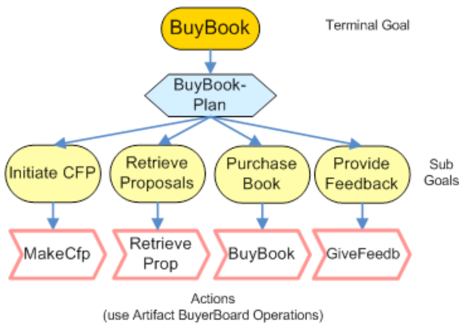

Figure 2. Buyer's goal overview.

4.1. General Purpose Capability

From a buyer viewpoint, the *BuyerBoard* artifact is serviceable for: *(1)* initiating the negotiation for a (*cfp*), and for propagating an initial signal to buyer agents who may want to participate; *(2)* collecting sellers proposals, *(3)* purchasing a book, and *(4)* providing feedback on seller's reputation. Each of these options affords the buyer for a given activity, where each of these activities corresponds to the leafs of buyer's goal hierarchy (see Figure 2). In the same way as in [2], the basic actions to interact have been designed using subgoals, which are dispatched within agent's `BuyBook` plan. Differently from [2], the concrete realization of the interaction between the actors is here external to the agents: noticeably, some part of the workflow of actions required to attain goals can be outsourced and delegated to the artifact operations.

As indicated by the general purpose capability description in Section 3, according to the goal to be adopted some specific parameters are required. For instance, for initiating a call for proposal (*cfp*) using the `MakeCfp` operation, a buyer agent may adopt a sub-goal by encoding the following instructions:

```
IGoal initiate = createGoal("use");
initiate.getParameter("AName").setValue("BuyerBoard");
initiate.getParameter("SName").setValue(sensorName);
Op op = new Op("MakeCfp", cfp, cfpinfo, sellers);
initiate.getParameter("op").setValue(op);
dispatchSubgoal(initiate);
```

Similarly, a final subgoal can be adopted by the buyer once he intends to provide feedback information by using the `GiveFeedb` operation:

```
IGoal feedback = createGoal("use");
feedback.getParameter("AName").setValue("BuyerBoard");
Op op = new Op("GiveFeedb", sellerName, feedbackValue);
feedback.getParameter("op").setValue(op);
dispatchSubgoal(feedback);
```

As discussed, the general purpose capability introduces two main usage approaches for interacting with artifacts.

[4]The full source code of the example, as well as the CARTAGO/*Jadex* integration technology, is available on CARTAGO web site.

From an operational viewpoint, the utilization of the primitives guarantees agents to fully operate in CARTAGO workspaces through execution of basic actions. The most common programming style is thus to directly exploit goals defined by the capability as basic actions and primitives to be dispatched from within a plan body. This allows an agent to directly exploit the provided operation through the artifact usage interface (see Figure 1).

The second approach exploits artifact events by translating them in *Jadex* native internal events. Differently from buyers who exploit their artifact in a goal directed fashion (with the purpose to receive a book), sellers can be conceived as reactive agents who: *(1)* focus on the `SellerBoard` artifact and retrieve the various *cfp*s coming from buyers; *(2)* once a call arrives, react by making an offer; *(3)* finally, upon a positive response coming from the buyer, execute the order by delivering the book. In this case the realization of the sellers is thus based upon the use of customized *Jadex* internal events which are generated on the fly by the capability on the basis of the events coming from the focused artifact. Due to a mechanism defined in the general purpose capability, once the seller has focused the `SellerBoard` by adopting a focus goal, he can automatically translate artifact events to proper *Jadex* internal events.

```
IGoal focus = createGoal("focus");
focus.getParameter("AName").setValue(aName);
dispatchSubgoalAndWait(focus);
```

Notice that, in order to initiate the automatic generation of internal events, the focus goal is dispatched without providing a `sensorName` parameter. Without having a sensor to store percepts, these events are posted in the reasoning process. Then, by carrying additional relevant information (i.e., the event name, type), they allow agents to simply react triggering the execution of specific plans. The following cutout shows the ADF descriptor used by the seller agent to autonomously retrieve *cfp* from `SellerBoard` artifact:

```
<plan name="incomingcfp">
  <body>new MakeProposalAction()</body>
    <trigger>
     <internalevent ref="artifactevent"><match>
       $event.label.equals("proposalfor"+getAgentName())
     </match></internalevent>
    </trigger>
</plan>
```

A new `MakeProposalAction` is triggered once an internal event matching a specified label is retrieved. Inside the `MakeProposalAction`, the goal directed approach can be used by the seller to interact with the board.

4.2. Special Purpose Capabilities

To further hide the single steps of the contract-net a special Contract-Net capability can be devised. At an interface level, such a capability provide the same resources as the one described in [2]. But, even if the goal directed approach to interact with artifacts remains the same, here the implementation of envisions a further layer by embedding all the mechanisms to interact with the artifact in a domain dependent module. Using artifacts instead of messages a buyer has just to create and dispatch the terminal goal *cfp_BuyBook*, providing the information about the participants, the concrete *cfp* and possibly a policy for evaluating the received proposals.

Exploiting special purpose capabilities to interact with artifacts is useful in attaining a twofold outcome. Firstly, it promotes a deep separation of concerns: the cost of arranging complex activities in subactivities or subtasks is be smaller once the developer can refer to the terminal goals placed by the problem domain. Secondly, this enables a customizable way to make agent's practical behavior easily adaptive to situated conditions, also improving promptness to proactively react to particular events which can be related to a particular context (i.e. action fails, unexpected changes etc.).

4.3. Discussion

Both message and artifact based approaches aim at disburden the agent developer from the tedious details of message passing. Accordingly, they provide a simple, goal-based interface for contract-net negotiations. Nevertheless the underlying model of the approach in [2] is still based on explicit speech-acts between a buyer agent and a set of seller agents. Therefore, the buyer needs to know potential sellers before starting the interaction (e.g. by doing a lookup in a yellow page service) and share the same protocol ontology. On the contrary, artifact based system allows to devise out a facility ruling and enabling the interaction between the agents. This, for instance, allows to abstract away from the pre-negotiation phases. The matchmaking between buyers and potential sellers can be moved from the agents into the artifacts logics (*precomputation*). One advantage of this is that different matchmaking strategies can be implemented in the environment without changing the code of the agents. A potential drawback is that the buyer agent has to abandon the control over the matchmaking process. In this view, artifacts assume a pivotal role in the context of open systems. They can be deployed in different nodes, thus realizing a distributed and reliable system spread over different working environments.

Moreover, this allows agents –independently from their platform– to coordinate activities by minimizing protocol handling, even more to cooperatively update and suitably retrieve information which is relevant for their tasks (i.e. seller's reputation, in this case). Besides, artifacts allow this information to persist in the overall system even beyond the presence of interacting agents. Once an agent uses an artifact he is both relying on its functionalities and delegating/externalizing part of his activities. In the contract-net example, `SellerBoard` at each round can sort the offers made by sellers –possibly following some policy specified by the buyer– thus freeing from the load to synchronize and sort proposals. From a cognitive perspective, this promotes the use of artifacts as suitable entities that agents may exploit to improve the repertoire of actions, thus intentionally externalizing some of their activities on external resources with the aim to to ease computational burden [7].

5. Related Work and Conclusion

The role of the environment as first-class abstraction in MAS and the importance of mediated interaction has been largely acknowledged in multi-agent system literature (see [20] for a survey). However, to authors' knowledge, few works consider the issue of integration of cognitive agents in properly designed environment and in particular goal-directed use of environment. Among the notable exceptions, we mention here Brahms [18], a multi-agent platform to develop and simulate multi-agent models of human and machine behavior, based on a theory of work practices and situated cognition. Another approach has been developed by Holvoet and Valckenaers [9], who introduced Delegate MAS as a mean to design environments in BDI-based agent architectures. Delegate MASs consist of light-weight agents, which are issued either by resources for building and maintaining information on the environment, or by task agents exploring the options on behalf of the agents in order to coordinate their intentions. GOLEM [5] introduced a platform for modeling situated cognitive agents in distributed environments by declaratively describing the representation of the environment in a logic-based form. Physical environments are modeled in terms of *containers* where agents based on the KGP model and *objects* are situated.

To conclude, in this paper we focussed the issue of goal-orientation in environment-mediated interaction, taking as a reference context artifact-based environments and BDI agents, with artifacts exploited by goal-directed agents using beliefs, plans and goals. As a concrete outcome, the integration of CARTAGO infrastructure and *Jadex* agent platform has been proposed and two different abstraction levels for goal-directed use of artifacts by *Jadex* agents have been identified: a first one that tackles a direct mapping of artifact operations to BDI goals, and a second one that is intended to provide a more abstract view on artifacts by focussing on the domain function they provide. With respect of our previous work [14], here we focussed on goal-directed interaction, tightly combining artifact functioning with agent perceptive and reasoning processes. Whereas [14] provides a suitable framowork allowing developers to build CARTAGO based environments where agents holding to different platforms are enabled to interact, we here mainly focus on the cognitive terms underlying those interactions and on the functional effects that such interaction may introduce in the overall system dynamics.

References

[1] M. Bratman. *Intention, Plans, and Practical Reason*. Harvard University Press, 1987.

[2] L. Braubach and A. Pokahr. Goal-oriented interaction protocols. In *MATES'07, Proceedings*. Springer, 2007.

[3] L. Braubach, A. Pokahr, and W. Lamersdorf. Extending the Capability Concept for Flexible BDI Agent Modularization. In *ProMAS'05,Proceedings*, 2006.

[4] L. Braubach, A. Pokahr, W. Lamersdorf, and D. Moldt. Goal Representation for BDI Agent Systems. In *ProMAS'04, Proceedings*, 2004.

[5] S. Bromuri and K. Stathis. Situating Cognitive Agents in GOLEM. In *Engineering Environment-Mediated Multiagent Systems (EEMMAS'07)*, 2007.

[6] P. Busetta, N. Howden, R. Rönnquist, and A. Hodgson. Structuring BDI agents in functional clusters. In *Agent Theories, Architectures and Languages (ATAL-00)*, 2000.

[7] A. Clark and D. Chalmers. The extended mind. *Analysis*, 58: 1:7–19, 1998.

[8] document no. FIPA0029. *FIPA Contract Net Interaction Protocol Specification*, chapter Foundation for Intelligent Physical Agents (FIPA). 2002.

[9] T. Holvoet and P. Valckenaers. Beliefs, desires and intentions through the environment. In *AAMAS'06, Proceedings*, pages 1052–1054, New York, NY, USA, 2006. ACM.

[10] B. A. Nardi. *Context and Consciousness: Activity Theory and Human-Computer Interaction*. MIT Press, 1996.

[11] A. Omicini, A. Ricci, and M. Viroli. Artifacts in the A&A meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 17 (3), 2008.

[12] M. Piunti and A. Ricci. Cognitive Artifacts for Intelligent Agents in MAS: Exploiting Relevant Information residing in Environments. In *Knowledge Representation for Agents and Multi-Agent Systems (KRAMAS-08)*, 2008.

[13] A. Pokahr, L. Braubach, and W. Lamersdorf. *Jadex: A BDI Reasoning Engine*, chapter Multi-Agent Programming. Kluwer Book, 2005.

[14] A. Ricci, M. Piunti, L. D. Acay, R. Bordini, J. Hubner, and M. Dastani. Integrating Artifact-Based Environments with Heterogeneous Agent-Programming Platforms. In *AAMAS'08, Proceedings*, 2008.

[15] A. Ricci, M. Viroli, and A. Omicini. The A&A programming model & technology for developing agent environments in MAS. In *ProMAS'07,Post-proceedings*, volume 4908 of *LNAI*, pages 91–109. Springer, 2007.

[16] A. Ricci, M. Viroli, and A. Omicini. CArtAgO: A framework for prototyping artifact-based environments in MAS. In *Environments for MultiAgent Systems III*, volume 4389 of *LNAI*, pages 67–86. Springer, May 2007.

[17] M. B. V. Riemsdijk, M. Dastani, and M. Winikoff. Goals in agent systems: A unifying framework. In *AAMAS'08, Proceedings*, 2008.

[18] M. Sierhuis and W. J. Clancey. Modeling and simulating work practice: A human-centered method for work systems design. *IEEE Intelligent Systems*, 17(5), 2002.

[19] D. Weyns, A. Omicini, and J. Odell. Environment as a First-class Abstraction in MAS. In *Autonomous Agents and Multi-Agent Systems* [20], pages 5–30.

[20] D. Weyns and H. V. D. Parunak. Special issue on environments for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 14(1):1–116, Feb. 2007.

[21] M. Wooldridge. *Reasoning about Rational Agents*. MIT Press, 2000.