"Give Agents their Artifacts": The A&A Approach for Engineering Working Environments in MAS

Alessandro Ricci
University of Bologna
Via Venezia 52, Cesena, Italy
a.ricci@unibo.it

Mirko Viroli
University of Bologna
Via Venezia 52, Cesena, Italy
mirko.viroli@unibo.it

Andrea Omicini
University of Bologna
Via Venezia 52, Cesena, Italy
andrea.omicini@unibo.it

ABSTRACT

In human society, almost any cooperative working context accounts for different kinds of object, tool, artifacts in general, that humans adopt, share and intelligently exploit so as to support their working activities, in particular social ones. According to theories in human sciences—Activity Theory and Distributed Cognition are two main examples [5, 4]—and to related disciplines in computer science—such as Computer Supported Cooperative Work (CSCW) and Human-Computer Interaction (HCI)—such entities have a key role in determining the success or failure of the activities, playing an essential function in simplifying complex tasks and—more generally—in designing solutions that scale with activity complexity. Such a perspective can be found also in some works in the context of Distributed Artificial Intelligence [1, 2].

Analogously to the human case, we claim that also (cognitive) multi-agent systems (MAS) could greatly benefit from the definition and systematic exploitation of a suitable notion of *working environment*, composed by different sorts of *artifacts*, dynamically constructed, shared and used by agents to support their working activities.

Along this line, in this paper first we introduce a conceptual framework called A&A (Agents and Artifacts) which aims at directly modelling and engineering working environments in the context of cognitive multi-agent systems; then, we provide a brief overview of the basic technologies that support such an approach, CARTAGO in particular—a Java-based framework for engineering working environments to be integrated with heterogeneous agent platforms.

Such a perspective is strenghtened by recent efforts in AOSE (Agent-Oriented Software Engineering) that remark the fundamental role of the environment for the engineering of MAS [8]. The A&A framework can be considered an instance of such approaches, with some specific peculiarity: *(i) abstractions and generality*—the aim is to find a basic set of conceptual abstractions and related theory which, analogously to the agent abstraction, could be general enough to be the basis to define concrete architectures and programming environments, but specific enough to capture the essential properties of systems; *(ii) cognitive*—analogous to designed environment in human society, the properties of such environment abstractions should be conceived to be suitably and effectively exploited by cognitive agents, as intelligent constructors / users / manipulators of the environment.

The work presented in this paper generalises and extends our previous work focussed on *coordination artifacts* presented at [6], and more recent works about the notion of artifact [7].

AAMAS'07 May 14–18 2007, Honolulu, Hawai'i, USA.

Categories and Subject Descriptors

D.1 [**Software**]: Programming Techniques; D.2 [**Software**]: Software Engineering; I.2 [**Computing Methodologies**]: Artificial Intelligence

General Terms

Artifact theory, Environment design, Multi-agent System languages

Keywords

Programming Multi-Agent Systems, Artifact-based Environments, Multi-Agent System Infrastructures

1. AGENTS, ARTIFACTS, WORKSPACES

The A&A conceptual framework introduces the notion of *working environment* analogously to the one typically found in CSCW literature: a working environment is defined as the part of the MAS that is designed and dynamically constructed and used by agents to support their working activities.

1.1 The Artifact Abstraction

In A&A an agent working environment is conceived as a dynamic set of *artifacts*, organised in *workspaces*. The notion of artifact is the core *abstraction* of A&A (see Figure 1 for an abstract representation): it is meant to represent any entity belonging to the working environment—hence existing outside the agent mind—that is created, shared & used (and eventually disposed) by agents to carry on their activities, in particular social ones. So, an artifact (type) is typically meant to be explicitly designed by MAS engineers so as to encapsulate some kind of *function*, here synonym of "intended purpose".

Artifacts work then as the basic building blocks to compose complex working environments: MAS designers can define different types of artifacts, according to the need of the application at hand or to devise a library of reusable artifacts. Analogously to artifacts as studied in human science disciplines, two basic categories of artifacts could be identified: *resources* and *tools*. While resources are primary source and target result of the agent activities, tools are

978-81-904262-7-5 (RPS)

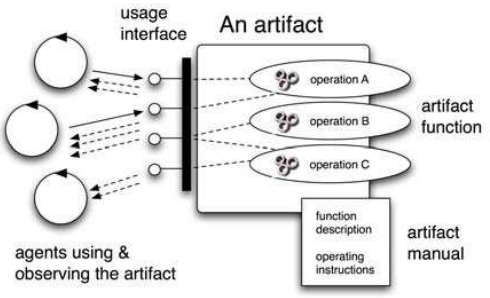

Figure 1: An abstract representation of the artifact abstraction. Agents use artifacts by triggering operation execution and by perceiving artifact observable state evolution. In evidence: the artifact usage interface and the artifact manual.

artifacts used as "instrument" to achieve some objective or execute some task.

Workspaces are the logical containers of artifacts, useful to define the topology of the working environment. Through workspaces it is possible to model a notion of locality, in terms of the artifacts that an agent can use and observe.

In this overall picture, nothing is said about the specific cognitive model of the agent: actually A&A is meant to be orthogonal to this aspect: agents are simply conceived as autonomous entities executing some kind of working activities, either individually or collectively—typically in order to achieve some individual or social goal, or to fulfill some individual or social task. Such activities—from an abstract point of view—are seen as the execution of sequences of actions, which according to the A&A model can be roughly classified as: *(i)* internal actions, *(ii)* communicative actions, involving direct communications with one or more agents through some kind of ACL, and *(iii)* pragmatical actions, as interactions within working environments that concern construction, sharing, and use of artifacts.

1.2 How Do Agents Use Artifacts: the Usage Interface

The artifact abstraction leads to a notion of *use* that is the basic kind of relationship among agents and artifacts, besides creation and disposal. Accordingly, the notion of *usage interface* is defined as the basic set of *operations* and *observable states* and *events* that an artifact exposes so as to be usable by agents. Informally, we can think about an agent interacting with an artifact through its usage interface as follows: an agent executes actions that result in the triggering of some artifact operations, which then leads to the observation of events or the evolution of the artifact state.

Such an abstraction strictly mimics the way in which humans use their artifacts: a simple example is the coffee machine, whose usage interface includes suitable controls—such as the buttons—and means to make (part of) the machine behaviour observable—such as displays—and to collect the results produced by the machine—such as the coffee can. It is quite evident by now that, differently from agents, artifacts are not meant to be autonomous or pro-active: they are meant to represent passive entities that are useful if and only if properly (created and) used by agents.

1.3 How Agents Reason About Artifacts: The Artifact Manual

As far as artifacts designed by humans for humans are concerned, a fundamental role is played by the artifact *manual*, which typically contains information about the purpose of the artifact, how it can be used, and possibly what to do in the case of malfunctioning. Actually, there could be different sorts of manuals: user manuals and technical manuals, the former written for people that only exploit the artifact, the latter for people that aim at understanding the internal working machinery—typically for either repairing or enhancing the artifact. Such concepts are essential for supporting humans in understanding which kind of artifacts could be useful for their work, and how to use them effectively.

We think that an analogous concept would be essential and effective also for artifacts in our A&A working environment, especially when considering cognitive agents in open MAS. Then, we envision that each artifact type is to be equipped by the artifact designer with a manual composed essentially by the *function description*—as the formal description of the purpose intended by the designer—, the *usage interface description*—as the formal description of artifact usage interface and observable states—, and finally the *operating instructions* [9]—as the formal description of how to properly use the artifact so as to exploit its functionalities.

2. PROTOTYPING TECHNOLOGIES

Starting from the A&A abstract model, we developed some first concrete technologies, with the objective to have concrete frameworks for prototyping MAS-based applications engineered upon A&A basic abstractions, and for being integrated with existing agent technologies extended with the A&A support.

A primary technology is called CARTAGO (Common ARtifact Infrastructure for AGent Open environment), which is a framework essentially providing the capability to define new artifacts type, suitable API for agents to work with artifacts and workspaces, and a runtime supporting the existence and dynamic management of working environments. Another technology is called simpA (simple A&A programming environment), which is a framework extending CARTAGO with a support for defining and running agents (MAS) besides the working environments.

While CARTAGO is meant to be integrated with existing (cognitive) agent models and technologies as a seamless support to define and create artifact-based working environments, simpA can be exploited alone to develop full-fledged applications engineered in terms of agents, artifacts and workspace. For lack of space, in this paper simpA is not described: the interested reader can refer to simpA web site[1]. Both technologies are based on Java and are available as open-source projects freely downloadable from the project web sites[2]. As first real-world application examples, CARTAGO and simpA are currently investigated as agent-based technologies for prototyping service-oriented applications based on Web Services in the context of logistics [3][4].

2.1 CARTAGO

An abstract view of CARTAGO architecture is shown in Figure 2. It is basically composed by three main parts:

- *API for creating and interacting with artifact-based working environments* — These API are meant to extend the existing basic set of agent actions with new ones, essentially for creating and disposing artifacts, interacting with them through their usage interface—by executing operations and

[1] http://www.alice.unibo.it/projects/simpa
[2] http://www.alice.unibo.it/projects/cartago
[3] http://www.alice.unibo.it/projects/simpa-ws
[4] http://www.alice.unibo.it/projects/a4stil

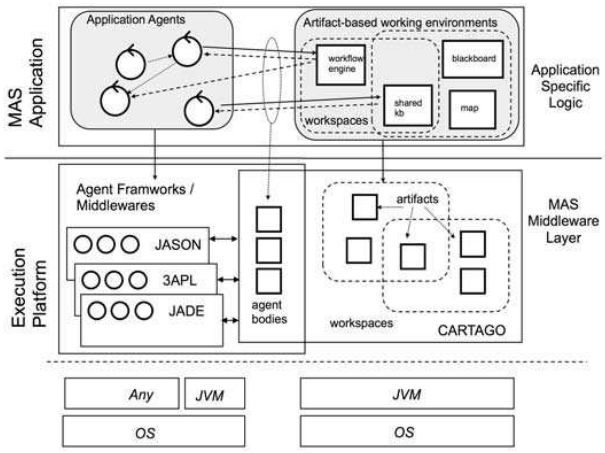

Figure 2: Abstract representation of a MAS application using CARTAGO. At the application layers there are application agents sharing and using some kind of application working environment, composed by workspaces populated by some kinds of artifact. At the MAS execution layer, CARTAGO functions as the virtual machines for the working environments—enabling the management and execution of artifacts—, exploited by means of suitable API by agent platforms, such as Jason, 3APL, JADE—which function as virtual machines for the management and execution of the agents.

perceiving artifact observable states and events—reading artifact function description and operating instructions, managing sensors, and so on.

- *API for defining artifact types* — Pragmatically, we chose Java as the programming language to define types of artifact and programming artifact behaviour, without introducing a new special purpose language. An artifact type can be defined by extending the basic `Artifact` class provided in the API: at runtime, artifacts instances are instances of this class. For defining artifact features, we adopted a choice that would favour rapid prototyping, that is, reusing as much as possible the support given by the Java Object-Oriented framework.

- *Runtime environment and related tools* — This is the part actually responsible of the life-cycle management of working environments at runtime. Conceptually, it is the *virtual machine* where artifacts and agent bodies are instantiated and managed that is responsible of executing operations on artifacts and collecting and routing observable events generated by artifacts. Some tools are also made available in CARTAGO for online inspection of working environment state, in particular artifact state, and above the observation of artifact behaviour, in terms of operation executed and events generated.

More details about CARTAGO API and architecture, along with complete examples, can be found on the web site.

2.2 Integration with Existing MAS Programming Environments

As mentioned previously, an important aspect of A&A and of technologies such as CARTAGO is the possibility of integration with existing cognitive MAS architectures and models / languages / platforms, so as to extend them to create and work with artifact-based environments.

Actually, most available agent programming models and platforms for developing general-purpose applications—such as *Jason*, 3APL, Jadex, JACK, and others surveied in [3]—lack a true notion of environment, and when such a notion is accounted for, it is typically modelled and implemented in terms of low-level interfaces to the hosting VM or OS environment, or by considering a general monolithic abstraction of "Environment" and of "Event". This is perfectly reasonable according to the notion of environment as traditionally dealt with in agent theories. By integrating these platforms with A&A, the environment notion is seamlessly extended with the capability for cognitive agents written in existing programming environments to create, share and use artifacts according to the specific needs, with MAS designers directly programming artifacts so as to create the best working environments for supporting agent activities. Also, existing types of artifact can be reused, especially those providing general purpose functionalities such as coordination artifacts like blackboards, maps, workflow engines.

3. CONCLUDING REMARKS

A&A and related technologies such as CARTAGO provide a new general-purpose abstraction layer for conceiving artifact-based working environments that MAS engineer can suitably design and build aside to agents, so as to systematize the engineering of resources and tools that agents can instantiate, share and cooperatively use to support their activities.

4. REFERENCES

[1] P. Agre and I. Horswill. Lifeworld analysis. *Journal of Artificial Intelligence Reserach*, 6:111–145, 1997.

[2] R. S. Amant and A. B. Wood. Tool use for autonomous agents. In M. M. Veloso and S. Kambhampati, editors, *AAAI/IAAI'05 Conference*, pages 184–189, Pittsburgh, PA, USA, 9–13 July 2005. AAAI Press / The MIT Press.

[3] R. Bordini, L. Braubach, M. Dastani, A. E. F. Seghrouchni, J. Gomez-Sanz, J. Leite, G. O'Hare, A. Pokahr, and A. Ricci. A survey of programming languages and platforms for multi-agent systems. In *Informatica 30*, pages 33–44, 2006.

[4] D. Kirsh. Distributed cognition, coordination and environment design. In *European conference on Cognitive Science*, pages 1–11, 1999.

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

[6] A. Omicini, A. Ricci, M. Viroli, C. Castelfranchi, and L. Tummolini. Coordination artifacts: Environment-based coordination for intelligent agents. In *AAMAS'04*, volume 1, pages 286–293, New York, USA, 19–23July 2004. ACM.

[7] A. Ricci, M. Viroli, and A. Omicini. Programming MAS with artifacts. In R. P. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Programming Multi-Agent Systems*, volume 3862 of *LNAI*, pages 206–221. Springer, Mar. 2006.

[8] M. Viroli, T. Holvoet, A. Ricci, K. Schelfthout, and F. Zambonelli. Infrastructures for the environment of multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 14(1):49–60, 2007.

[9] M. Viroli, A. Ricci, and A. Omicini. Operating instructions for intelligent agent coordination. *The Knowledge Engineering Review*, 21(1):49–69, Mar. 2006.