Environment and Artifacts in Jadex

Mirko Morandini
Università di Trento, Italy
morandini@dit.unitn.it

May 21, 2007

Abstract

Most multi-agent systems are embedded into a specific environment, shared among the agents. *Artifacts* can be used to model this environment and the operations agents can use to access it.

This article shows how the environment is currently modelled in implementations based on the Jadex BDI agent framework and evaluates the possibility to add to Jadex an explicit definition of the environment using artifacts.

1 Introduction

Jadex [4] is a framework for constructing and running software agents which base on a BDI (Belief, Desire, Intention) architecture. BDI is a well-known model for representing mentalistic concepts in the design and implementation of agent-oriented systems and enables to view an agent as a goal-directed entity that acts in a rational manner.

Following an architecture based on these concepts, agents are typically designed defining the goals they have to achieve, the plans (the concrete actions) that can be executed to achieve these goals, and a belief base, where the agent's belief of the external world can be stored [6].

Software agents are defined to be autonomous entities which pursue their own goals and are able to interact with each other. Do do this, agents have to be *situated* in an environment, which is the context where their actions take place. On one hand we have the social environment, characterized by the surrounding agents and human actors, to whose an agent can build a relationship by communication and interaction, and the technical environment, consisting of the features of the platform where the agents run. On the other hand, most Multi-Agent Systems (MAS) work embedded into a specific "physical" environment, which represents the (real world or simulated) resources an agent can rely on, to sense and act. It is public to the agents and modifications on the environment by one agent should be immediately perceptible by the others. This is the environment we refer to in this paper.

In their examples the Jadex developers model the agent's environment for example including a singleton Java object included in each agent's belief bases, which gives global access to all information on the environment and the methods to manipulate it. Although this solution works, it seems like a workaround, as it doesn't fit well neither into the Jadex concept of *defining* the main parts of the system in XML files, nor into BDI theory, where an agent's *belief base* should reflect the agent's (possibly restricted) view of the world.

In this article we propose to add to the Jadex framework the possibility to explicitely model the environment where the agents are situated and the artifacts agents can use to interact with it, by following the *Agents and Artifacts* metamodel proposed in [3].

Section 2 provides some basic information on BDI agents and on the Jadex framework. Section 3 discusses the need for representation of the environment in a MAS and Section 4 proposes a possible implementation.

2 Technology and Theory

2.1 BDI Agents

The BDI-model [5], which roots in Bratman's *theory of practical reasoning* [1], bases on the three mental state concepts of Belief, Desire, and Intention. Beliefs are the agents knowledge about the current state of the environment; it may be incorrect or incomplete. Desires are the states of affairs, that the agents would, in an ideal world, wish to achieve. They should be consistent with one another. Intentions are the activities, which the agent has committed for execution, to satisfy a desire.

The BDI-model naturally fits to agents that are goal directed, communicate with others and can react to changes of the environment they are embedded in.

For an executable model, Rao and Georgeff introduce the more concrete concepts of Goals and Plans [6]. Goals are chosen, consistent desires an agent tries to fulfill (believing that they are achievable), plans contain the means of achieving certain future world states (the possible actions to execute in order to reach a goal), where intentions are a set of plans the system tries to execute to reach a goal. Most concrete BDI architectures use a model based on the three concepts belief, goal and plan.

2.2 Jadex

Jadex [4] is a well-known framework for implementing and running agents based on a BDI architecture. It is built on top of the Jade platform and allows agent developers, to implement agents which exhibit a rational, goal-oriented (opposed to task-oriented) behavior, with an explicit representation of goals. Jadex consists of an API, an agent platform and development tools.

A Jadex agent is programmed defining an XML file, called agent definition file (ADF), and Java classes. In the ADF, the agent is specified by declaring beliefs, goals, available plans and some other properties. In addition to this, for each plan used by the agent the plan body (the concrete actions that have to be executed) has to be implemented in a separate Java class.

The beliefs are stored, independently for every agent, in a so-called *belief base*. The belief base is an object-oriented database, which can store Java objects and be queried using simple SQL statements. Objects can be inserted into the belief base at definition time in the ADF or at run-time from a Java file. Goals are entirely defined in the ADF file and can have different types: goals to achieve some state in the belief base, to maintain a certain state in time or to execute at least one plan with success. For the plans, in the ADF there have to be defined the interface with input and output parameters, the goals that can be achieved by the plan and the Java file containing the plan code. In short, if a goal is activated, one of the plans (with the connected Java class) which can achieve it, is executed.

3 The Need for an Environment

> *Without an environment, an agent is effectively useless. Cut off from the rest of its world, the agent can neither sense nor act. (J. Odell [2])*

Most platforms for multi-agent systems, including Jadex, support no explicit modeling of the environment where the agents are situated. The platforms give only the technical support for message passing and the basic services for agent management and agent retrieval through white and yellow pages, as requested by the FIPA standard. However, the environment should provide much more than this. In most software systems, a global representation of the surroundings of the whole MAS is needed, a medium for sharing information and

mediating coordination among agents. An important functionality of the environment should be to embed resources and services, which are typically situated in a physical structure [8]. The following paragraphs show some examples for the purpose of giving evidence for the need of an explicit representation of the environment of a multi-agent system.

3.1 Actual representation of environmental information in Jadex

To see if and how the environment is represented in multi-agent systems based on Jadex and which architectural choices are typically made, we examined some of the developers' examples[1]. Not surprisingly, most examples are strongly bounded to a physical (simulated) environment, like a floor to clean, some terrain to exploit or the woods with prey to hunt.

In some examples (e.g. the simple Garbage Collector example and the Marsworld example) the Jadex developers model the agent's environment including a singleton Java object into each agent's belief base. This static object shared by all belief bases holds all the needed information on the environment, from the size of the world to the items in it, and the methods to access it. Moreover, the rules the agents have to respect are included within the methods provided to access and modify the variables in the environment.

Although this solution works, it relies on features of the Java language to create a partition of a belief base global to all agents. However, following the concepts of a BDI architecture, the belief base should reflect the agent's (possibly restricted) view on the world, not give direct access to the whole environment. Moreover, it doesn't fit well into the Jadex philosophy of *defining* the main parts of the system and the interfaces in XML files. To sense and act, agents have only to access their belief base. So for example the implicit communication between a garbage collector and a garbage burner agent is reduced to an access to a variable.

A second possibility to model the environment is used in many other MAS, e.g. the hunter-prey example. A special *environment agent* is implemented, which has to be contacted by all other agents to sense the environment and to perform actions on it. This agent contains a huge belief base and has to fulfill all the incoming requests, becoming de facto a service provider, rather than being autonomous. However, with exception of the environment agent, this approach seems to meet the underlying Jadex and BDI agent philosophies much better.

3.2 Evaluating an explicit representation of the environment

The examples given in the last section emphasize the need for adding the concept of environment into the Jadex agent architecture. To realize this, we base on the *Agents and Artifacts* metamodel proposed in [3]. By adding the concept of *Artifact* to a multi-agent system, we can propose the implementation of an environment which is much more complete than, for example, a simple database containing global variables or objects with attributes and the methods to access them, defined in Java.

Introducing Artifacts

Artifacts are new independent computational entities introduced in multi-agent systems, to represent any kind of resource or tool that agents can dynamically use, share and manipulate. Artifacts strongly differ from agents, because they are not autonomous entities, but entities aimed to serve the agents. Thereby they have to be predictable and to operate in a transparent way. Agents can call the operations an artifact exhibits on its usage

[1] see `http://vsis-www.informatik.uni-hamburg.de/projects/jadex/examples.php`

interface, to change their state, to get information (to sense), or to produce the desired effect on the environment. Artifacts encapsulate and provide access to resources in the environment, like objects following the object-oriented paradigm. Although, the differ because they persist in the system and exhibit the interface to their available operations, invokeable through request messages. Moreover, artifacts can rely on other artifacts and exhibit a reactive behavior on environment changes.

Using artifacts, it is possible to model the environment and the mediated access to it. For example, they can enable agents to access a database or a web service or to interact through the environment using a blackboard functionality [8]. In addition, artifacts could rely on a public base of facts which represent environmental variables.

With the notion of artifact it should be possible to rearrange the environment in the examples of the previous section, respecting the autonomy of agents and the locality of their belief bases, and avoiding the use of workarounds.

Therefore, we propose to introduce the concept of artifact natively into the Jadex agent architecture.

4 Solution Proposal

Analogous to the definition of agents in XML files, we propose to define the MAS environment in an XML file called "environment definition file" *(EDF)*. This file should be the executed prior to starting the agents belonging to the MAS. Alternatively, it should be possible to define the agents to be started directly here, avoiding the need for a *manager* agent with merely this purpose.

The proposed EDF contains two main sections: A *facts base* similar to the agents' belief base and the definition of artifacts. Each artifact section contains the definition of its own local facts base and the interface to its operations, including a reference to an external Java file, similar to the Java files used for the implementation of the agents' plans.

To facilitate the reuse of artifacts, artifact definition files are introduced, which can contain one or more artifacts, and can be included into the environment similar to the inclusion of *capability definitions* in agents. This definition of independent artifacts can also be used to implement individual artifacts and artifacts for social interaction as defined in [3].

The operations defined by artifacts should be accessible from the agents' Java plan files by sending special request messages. This should be the only way for an agent to access and modify the environment. An artifact itself can link to other artifacts and access the facts base defined in the environment. Moreover, triggers can be defined, which link to an entry in this database. This enables artifacts to exhibit a reactive behavior. An example for this could be a sensor which tells to an agent a new value, every time this value changes.

5 Conclusions

In this paper we pointed out that agent platforms such as Jadex currently lack of the explicit definition of the environment surrounding a multi-agent system. Although, a representation of the environment is needed in almost every system, starting from simple examples. Actually, it is usually represented as a shared area accessible through each agent's private belief base or by a single agent which endows all the information; both solutions seem workarounds and do not fit very well into the agent-oriented programming paradigm.

Basing on the *Agents and Artifacts* metamodel, this paper proposes to add to Jadex the possibility to define the environment by defining the artifacts it contains. Artifacts are non-autonomous entities, agents can use the operations they expose, to sense and act on the environment. Using this new concepts, multi-agent systems can be defined not only

being compliant with the paradigm of autonomous agents, but also with a better structure and reducing the necessary Java code.

References

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

[2] J. Odell, H. V. D. Parunak, M. Fleischer, and S. Brueckner. Modeling agents and their environment: The physical environment. *Journal of Object Technology*, 2(2):43–51, 2003.

[3] A. Omicini, A. Ricci, and M. Viroli. *Agens Faber*: Toward a theory of artefacts for mas. *Electr. Notes Theor. Comput. Sci.*, 150(3):21–36, 2006.

[4] A. Pokahr, L. Braubach, and W. Lamersdorf. Jadex: A bdi reasoning engine. In J. D. R. Bordini, M. Dastani and A. E. F. Seghrouchni, editors, *Multi-Agent Programming*, pages 149–174. Springer Science+Business Media Inc., USA, 9 2005. Book chapter.

[5] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1991.

[6] A. S. Rao and M. P. Georgeff. Bdi agents: From theory to practice. In *ICMAS*, pages 312–319, 1995.

[7] M. Viroli, A. Omicini, and A. Ricci. Engineering mas environment with artifacts. In H. V. D. P. Danny Weyns and F. Michel, editors, *2nd International Workshop Environments for Multi-Agent Systems (E4MAS 2005), AAMAS 2005, Utrecht, The Netherlands*, 2005.

[8] D. Weyns, A. Omicini, and J. Odell. Environment as a first class abstraction in multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 14(1):5–30, 2007.