Toward a pragmatic methodology for open multi-agent systems

Philippe Mathieu, Jean-Christophe Routier, and Yann Secq

Laboratoire d'Informatique Fondamentale de Lille – CNRS UMR 8022
Université des Sciences et Technologies de Lille
59650 Villeneuve d'Ascq Cedex
{mathieu,routier,secq}@lifl.fr

Abstract. This paper introduces the Rio methodology, which relies on the notions of Role, Interaction and Organization. We define a model of *runnable* specification of interaction protocols that represents interactions between agents globally, and that can be processed in order to generate Colored Petri Nets that handle agent conversations. Moreover, these specifications are not tied to a particular agent model and could therefore be used to enable multi-agent systems interoperability.
The Rio methodology aimed at the design of open multi-agent systems, and is based on an engineering of these *runnable* interaction protocols. They are described in term of conversation between *micro-roles* characterized by their skills, then *micro-roles* are gathered in *composite roles*. These *composite roles* are used to build *abstract agents*. Lastly, these latter can be distributed among the agents of a running multi-agent system.

1 Introduction

The idea of an agent based software engineering has appeared roughly ten years ago, with the paper from Shoham entitled *Agent Oriented Programming*[15]. Since these days, several methodologies have been proposed to help developers in their analysis and design[9, 2]. These methodologies are interesting but they generally do not propose pragmatic concepts or principles facilitating the realization of such systems. Moreover, they do not tackle the problem of multi-agent systems interoperability that should be a primary concern as more and more agent models and agent platforms are becoming available.

The FIPA association has published some standard specifications that provide a first step towards interoperability. Sadly, we believe that without a strong commitment to interaction specification and standardized ontologies, interoperability will not be reached. Thus, our proposal is a pragmatic methodology to ease the design of open multi-agent systems. To achieve this goal, we rely on a model of *runnable* specification of interaction protocols, which describes a global sight of the conversations between the agents of the system. Unlike AgentUML [13], our model of interaction is more than a description or a notation : we use it to generate code to handle agent conversations.

In the first part of this article, we briefly survey two methodologies that have been proposed for the design of complex distributed systems, then we put them in relation with interaction oriented approaches. In the second part, we propose a formalism for the *runnable* specifications of interaction protocols between *micro-roles*. The latter are assembled in composite roles which are then atributed to the agents of the system. This specification is made executable by the generation of Colored Petri Nets for each *micro-role*. This *runnable* specification and the use of a generic model of agent enable us to propose the RIO methodology facilitating the design, the realization and the effective deployment of multi-agent systems.

2 Agent methodologies and interaction oriented programming

Several agent oriented methodologies have been proposed like AALAADIN[4] or GAIA [18]. These two methodologies do not make any assumption on agent models and concentrate mainly on the decomposition of a complex system in term of roles. This point is fundamental, in particular because of the multiplicity of available agent and multi-agent systems models. This multiplicity makes the task of the developer difficult : which agent model should be used? Which organizational model should be chosen? Indeed, each platform imposes too often both its own agent model and its organizational model. These methodologies are interesting on many points, but remains too general to ease the transition from the design stage to its concrete realization. Moreover, the various levels of communication are not clarified in the description of interaction protocols. Indeed, works on agent communication languages (ACL)[5] identify three levels that constitutes a conversation: the semantic, the intention (these two are expressed through languages like KIF or SL, and KQML or FIPA-ACL), and the interaction level. However, even by considering heterogeneous platforms sharing the same ontology, it remains difficult to have guarantees on the respect of interaction protocols. This is the reason why works have been undertaken to formalize this aspect with several objectives: to describe the sequence of messages, to have certain guarantees on the course of a conversation and to ease interoperability between heterogeneous platforms.

To illustrate these approaches based on a formalization of the interactions, we studied three of them: APRIL[12], AGENTALK[8] and COOL[1]. APRIL is a symbolic language designed to handle concurrent processes (close to works done on actor languages [6]), that eases the creation of interaction protocols. In APRIL, the developer must design a set of *handlers* which treats each message matching a given pattern. According to the same principles, AGENTALK adds the possibility to easily create new protocols by specialization of existing protocols, by relying on a subclassing mechanism. Another fundamental contribution of AGENTALK is the explicit description of the protocol: the conversation is represented by a set of states and a set of transition rules. This same principle was employed in COOL, which proposes to model a conversation using a finite

state automata. In COOL, the need for the introduction of *conventions* between agents to support coordination is proposed. This concept of *convention* must be brought closer to the works of Shoham on *social rules*[16] and their contributions on the global performance of the system.

Thus, the introduction of this level of interaction management while rigidifying in a certain way the possible interactions between agents, brings guarantees on coordination and allows the reification of these interactions. The table below, inspired by work of Singh[17], illustrates the various levels of abstractions within a multi-agent system :

Applicative skills	Business knowledge
Agent models and system skills	Agent oriented design
Conversation management	Interaction oriented design
Message transport	Agent platform (i.e. agents container)

To conclude, we would like to cite a definition given by Singh[17] of the interaction oriented approach, which characterizes our approach :

We introduce interaction-oriented programming (IOP) as an approach to orchestrate the interactions among agents. IOP is more tractable and practical than general agent programming, especially in settings such as open information environments, where the internal details of autonomously developed agents are not available.

It is the point of view that we adopt, by proposing a pragmatic methodology for the design and realization of open multi-agent systems, relying on the concept of *runnable* specification of interaction protocols.

3 Rio : Roles, Interactions and Organizations

In this section, we will present the methodology that we are developing, and which relies on the concept of *runnable* specification of interaction protocols. This methodology falls under the line of Gaia[18], and thus aims the same applicability. However, Gaia remains too general to ease the transition from the system design stage to its realization. The purpose of our proposal is to facilitate this transition. The Rio methodology relies on four stages, the two first represent reusable specifications, while the two last are singular with the targeted application. Moreover, we will not speak about building an *application*, but about designing an agent society. Indeed, the Rio methodology proposes an incremental and interactive construction of multi-agent systems. We see a multi-agent system like a set of distributed containers that have to be enriched by interactions in order to achieve tasks. We will detail this approach by studying the four stages of our methodology (figure 1). It should be noted that the analysis stage is not represented here and could be done using Gaia. When entities and processes that are involved have been identified, we can start the Rio methodology by specifying interaction protocols.

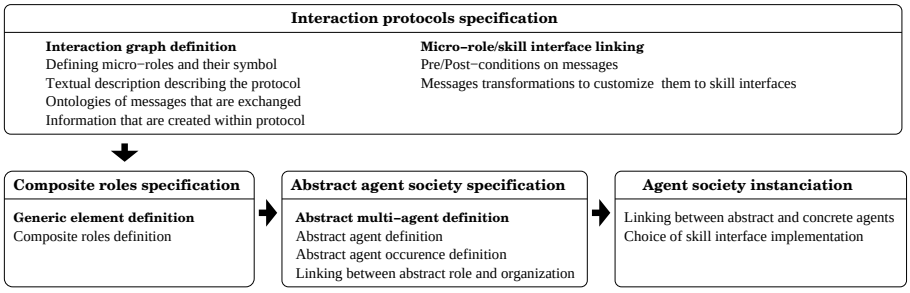


Fig. 1. The stages of the Rio methodology

3.1 Interaction protocols specification.

The first stage consists in identifying the involved interactions and roles. Then, it is necessary to determine the granularity of these interactions. Indeed, for reasons of re-use of existing protocols, it is significant to find a balance between protocols using too many roles, these protocols becoming thus too specific, and protocols where there are only two roles, in this case the view of the interaction is no more global.

The purpose of these runnable interaction protocol specifications, is the checking and the deployment of interaction protocols within multi-agent systems. On all these stages, the designer has to define the specification, the other stages being automated. For that, we define a formalism representing the global view of an interaction protocol, and a projection mechanism which transforms this global view into a set of local views dedicated to each role. The interaction protocols are regarded here as social laws within the meaning of Shoham[16], that means that agents lose part of their autonomy (conversational rules are static), but the system gains in determinism (because conversations are precisely defined) and in reliability (because the code needed to handle the flow of messages is generated).

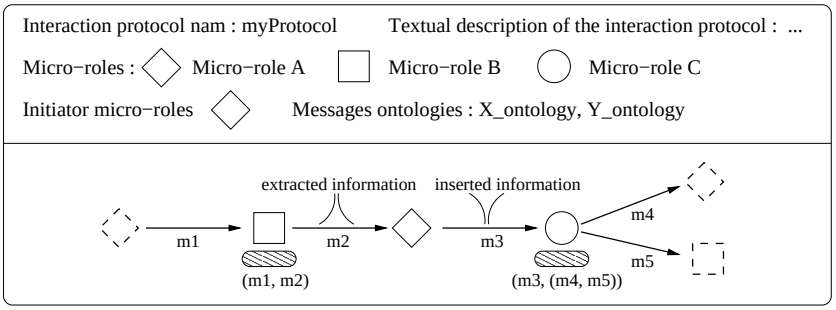
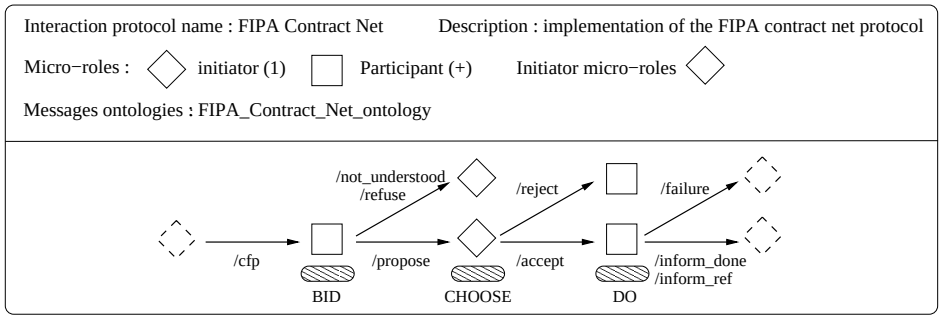


Fig. 2. Cartouche that specifies interaction protocols

The specification of the interaction protocols can then be done in three ways : either *ex-nihilo*, by specialization, or by composition. Creation *ex-nihilo* consists in specifying the interaction protocol by detailing its cartouche (figure 2). Specialization makes it possible to annotate an existing cartouche. Thus, it is possible to specify the cartouche of an interaction protocol such as FIPA ContractNet, its specialization will consist in changing *micro-roles* names to adapt them to the application, to refine message patterns, and if required to modify insertions/extractions of information. Finally, the composition consists in assembling existing protocols by specifying associations of *micro-roles* and information transfers. The figure 3 illustrates a specification of the FIPA Contract Net Protocol, that is represented by an XML encoding reproduced below.




```
<interaction>
  <protocol name="FIPA Contract Net">
    <roles>
      <role name="Initiator" occurence="1"/>
      <role name="Participant" occurence="+"/>
    </roles>
    <skills>
      <skill name="bid">
        <input type="/cfp"/>
        <output type="/propose"/>
        <output type="/refuse"/>
        <output type="/not_understood"/>
      </skill>
      <skill name="choose">
        <input type="/propose"/>
        <output type="/reject"/>
        <output type="/accept"/>
      </skill>
      <skill name="do">
        <input type="/accept"/>
        <output type="/failure"/>
        <output type="/inform_done"/>
        <output type="/inform_ref"/>
      </skill>
    </skills>
```

```
    <net>
      <nodes>
        <place role="Initiator" initial="1"
               ids="1,3,4,7,8"/>
        <place role="Participant" ids="2,5,6"/>
      </nodes>
      <arcs>
        <arc from="1" to="2" type="/cfp"/>
        <arc from="2" to="3" type="/refuse"/>
        <arc from="2" to="3"
               type="/not_understood"/>
        <arc from="2" to="4" type="/propose"/>
        <arc from="4" to="5" type="/reject"/>
        <arc from="4" to="6" type="/accept"/>
        <arc from="6" to="7" type="/failure"/>
        <arc from="6" to="8"
               type="/inform_done"/>
        <arc from="6" to="8"
               type="/inform_ref"/>
      </arcs>
    </net>
  </protocol>
</interaction>
```

Fig. 3. The FIPA Contract Net interaction protocol and its XML encoding

Now that we have seen the formalism representing interaction protocols, we will explain the transformation making it possible to obtain a *runnable* specification. The specification of interaction protocols gives to the designer a global view of the interaction. Our objective is to generate for each *micro-role* a local view starting from this global view, this one could then be distributed dynamically to the agents of the system.

The projection mechanism transforms the specification into a set of automata. More precisely, an automata is created for each *micro-role* (figure 4). This automata manages the course of the protocol : coherence of the protocol (messages scheduling), messages types, side effects (skill invocation). The implementation of this mechanism is carried out by the generation of Colored Petri Nets[7]. Indeed, we use the color of tokens to represent messages patterns, in addition we have a library facilitating the interactions between generated networks and the agent skills. On the basis of the interaction graph, we create a description of Colored Petri Net for each *micro-role*, and we transform this textual description to a JAVA class. This class is then integrated within a skill, which is used by conversation manager skill.

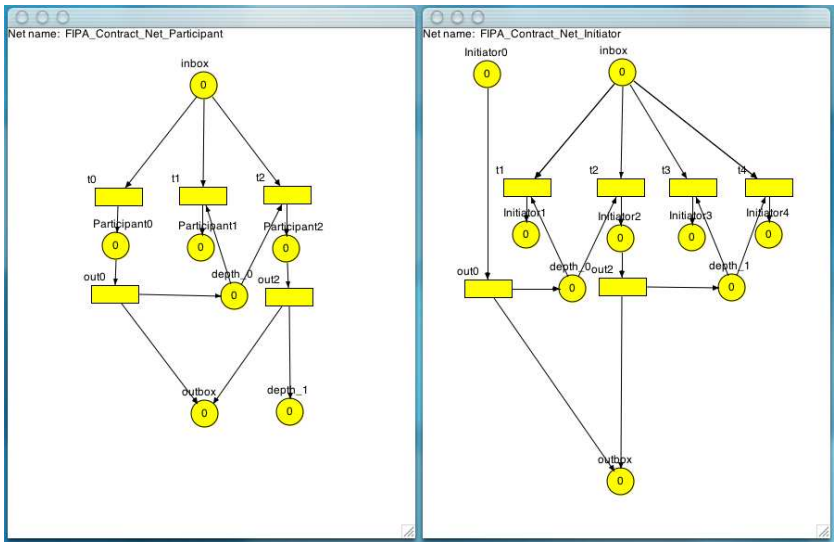


Fig. 4. Generated Colored Petri Nets for FIPA Contract Net protocol.

The interest of this approach is that the designer graphically specifies the global view of the interaction, the projection mechanism generates the skill needed to the management of this interaction. Moreover, thanks to the dynamic skill acquisition, it is possible to add new interaction protocols to running agents of the system.

At the end of this stage, the designer has defined a set of interaction protocols. He can then go to the next stage : the description of *composite roles*, which will allow the aggregation of *micro-roles* that are involved in complementary interactions.

3.2 Composite roles specification.

This second stage specifies role models. These models are abstract reusable descriptions, which correspond to a logical gathering of *micro-roles*. These patterns define *abstract* roles, which gather a set of consistent interaction protocols. For example, a composite role SUPPLIES MANAGEMENT will gather the *micro-role* BUYER within the PROVIDERS SEEKING interaction protocol and the *micro-role* STOREKEEPER of the interaction SUPPLIES DELIVERY. Indeed, a role is gener-

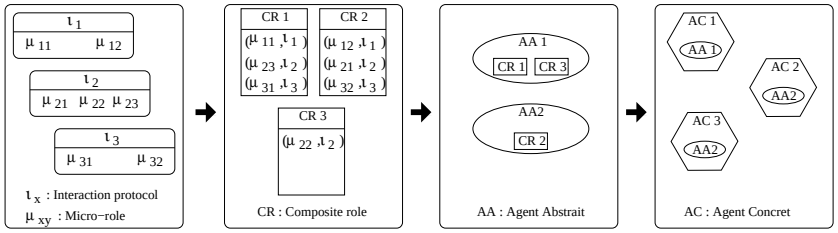


Fig. 5. Synthetic illustration of RIO stages

ally composed of a set of tasks which can be, or which must be carried out by the agent playing this role. Each one of these tasks can be broken up and be designed as a set of interactions with other roles. The concept of *composite role* is thus used to give a logical coherence between the *micro-role* representing the many facets of a role.

3.3 Agent societies specification.

This third stage can be regarded as a specification of an abstract agent society, i.e. a description of *abstract agents* and their occurrence, as well as the link between *composite roles* and organizations. Once the set of *composite roles* is created, it is possible to define the *abstract agents*, which are defined by a set of *composite roles*. *Abstract agents* can seen as agent patterns, or agent templates : they define a type of agent relying on interactions that the agent can be involved in. For the designer, these *abstract agents* describe applicative agent models. Because these models introduce strong dependencies between composite roles, they are specific to the targeted applications and cannot therefore hardly be reused.

For example, the OFFICE STATIONERY DELIVERY *composite role* could be associated with the TRAVELLING EXPENSES MANAGEMENT *composite role* to

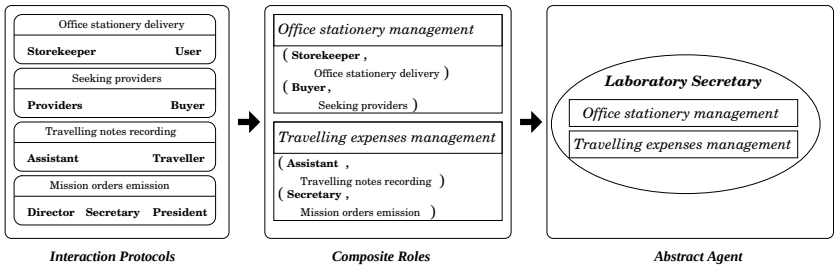


Fig. 6. The Secretary example

characterize a LABORATORY SECRETARY *abstract agent* (fig 6). Once *abstract agents* are defined, it is necessary to specify their occurrence in the system. This means that each *abstract agent* has an associated cardinality constraint that specifies the number of *instances* that could be created in the system (exactly N agents of this type, 1 or more, 0 or more, or [m..n]).

The second part of this stage consists in specifying for each *abstract agent*, and even for the *composite roles* of these agents, which organization should be used to find their acquaintances. Indeed, when agents are running, they have to initiate interaction protocols, but in order to do that they initially have to find their interlocutors. Organizations are used as a media for this search. The concept of organization is necessary to structure interactions that intervene between entities of the system. This concept brings some significant benefits : a means to logically organize the agents, a communication network per defect and a media to locate agents, roles or skills. Moreover, its reification provides a back door in the system, making it possible to visualize and to improve interactions between agents[11]. So, in this stage, the designer has to associate each interaction with the organization that should be used to locate acquaintances. It is thus possible to use different organizations within each *abstract agent*.

3.4 Instantiating an agent society in a multi-agent system.

This last stage specifies deployment rules of the *abstract agents* on the running agents of a system. We have a complete specification of the agent society, that can be mapped on the concrete agents of the multi-agent system. For that, it is necessary to indicate the assignments from *abstract agents* to concrete ones. Then, the connection between a skill interface and its implementation is carried out. The designer indeed must, according to criteria that are applicative or specific to the hosting platform, bind the implementation with skill interfaces.

It is during deployment that the use of our generic agent model[10] is justified as a support to the dynamic acquisition of new skills related with the interaction. Indeed, the interaction, once transformed by the projection mechanism, is represented for each *micro-role* by a Colored Petri Net (figure 4) and its associated skills. All these information are sent to the agent, which adds applicative skills and delegates the CPN to the conversation manager.

When an agent receives a message, the conversation manager checks if this message is part of a conversation in progress (thanks to the conversation identifier included in the KQML message), if it is the case, it delegates the message processing to the concerned CPN instance, if not it seeks the message pattern matching for the received message and instantiates the associated CPN. If it does not find any, the message will have to be treated by the agent model.

4 Conclusion

Multi-agent systems are now becoming a new design paradigm. Many agent models have been proposed, as well as many multi-agent toolkits or framework. This diversity poses the problem of multi-agent systems interoperability. In this paper, we have presented the Rio methodology, which is based on the notions of role, interaction and organization. We use these notions to propose an engineering of interaction protocols and to enable interoperability for open multi-agent systems.

The heart of our proposal is the idea of *runnable* specification of interaction protocols. This specification, that can naturally be represented graphically, gives a global view of a conversation between several roles. These *micro-roles* are characterized by the messages that they can handle and by the skills they need in order to fulfill their function. Then, we have a projection algorithm that generate Colored Petri Nets for each *micro-role*, that represent the local vision that each participant has of the interaction. By using these specifications, it is possible to create abstractions that characterize the various roles and agents of a multi-agent system : the *composite roles*, which gather a set of *micro-roles*, and *abstract agents*, which gather a set of *composite* roles. These specifications are described in an XML encoding, making it possible for other agent/platform designers to implement interaction protocol management within their agent/platform.

We are working on an agent platform and its associated tools to support the Rio methodology. The technologies that we are using are Colored Petri Nets to manage conversations, Owl (formerly Daml+Oil) for messages ontologies and knowledge representation [14], and an OSGi framework as component model [3] to implement agent skills.

References

1. M. Barbuceanu and M. S. Fox. Cool: A language for describing coordination in multiagent systems. In *Proceedings of the First International Conference oil Multi-Agent Systems (ICMAS-95)*, pages 17–24, San Francisco, CA, 1995.
2. F. M. T. Brazier, B. M. Dunin-Keplicz, N. R. Jennings, and J. Treur. DESIRE: Modelling multi-agent systems in a compositional formal framework. *Int Journal of Cooperative Information Systems*, 6(1):67–94, 1997.
3. H. Cervantes and R.S Hall. Beanome : A component model for the osgi framework. In *Workshop on Software Infrastructures for Component-Based Applications on Consumer Devices, held in Lausanne in September 2002.*

4. J. Ferber and O. Gutknecht. Operational semantics of a role-based agent architecture. In *Proceedings of ATAL'99*, jan 1999.
5. T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an Agent Communication Language. In *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463, Gaithersburg, Maryland, 1994. ACM Press.
6. C. Hewitt. Viewing control structures as patterns of passing messages. In *Artificial Intelligence: An MIT Perspective*. MIT Press, Cambridge, Massachusetts, 1979.
7. Kurt Jensen. Coloured petri nets - basic concepts, analysis methods and practical use, vol. 1: Basic concepts. In *EATCS Monographs on Theoretical Computer Science*, pages 1–234. Springer-Verlag: Berlin, Germany, 1992.
8. Nobuyashu Osato Kazuhiro Kuwabara, Toru Ishida. Agentalk : Describing multiagent coordination protocols with inheritance. In *Proc. 7th International Conference on Tools with Artificial Intelligence (ICTAI'95)*, pages pp. 460–465, 1995.
9. E. A. Kendall, M. T. Malkoun, and C. H. Jiang. A methodology for developing agent based systems. In Chengqi Zhang and Dickson Lukose, editors, *First Australian Workshop on Distributed Artificial Intelligence*, Canberra, Australia, 1995.
10. P. Mathieu, J.C. Routier, and Y. Secq. Dynamic skill learning: A support to agent evolution. In *Proceedings of the AISB'01 Symposium on Adaptive Agents and Multi-Agent Systems*, pages 25–32, 2001.
11. P. Mathieu, J.C. Routier, and Y. Secq. Principles for dynamic multi-agent organisations. In *Proceedings of Fifth Pacific Rim International Workshop on Multi-Agents (PRIMA2002)*, August 2002.
12. Frank G. McCabe and Keith L. Clark. April – agent process interaction language. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents: Theories, Architectures, and Languages (LNAI volume 890)*, pages 324–340. Springer-Verlag: Heidelberg, Germany, 1995.
13. J. Odell, H. Parunak, and B. Bauer. Extending uml for agents, 2000.
14. Filip Perich, Lalana Kagal, Harry Chen, ovrin STolia, Youyong Zou, Tim Finin, Anupam Joshi, Yun Peng, R. Scott Cost, and Charles Nicholas. ITTALKS: An application of agents in the Semantic Web. In *Engineering Societies in the Agents World II*, volume 2203 of *LNAI*, pages 175–193. Springer-Verlag, December 2001. 2nd International Workshop (ESAW'01), Prague, Czech Republic, 7 July 2001, Revised Papers.
15. Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, 1993.
16. Yoav Shoham and Moshe Tennenholtz. On social laws for artificial agent societies: Off-line design. *Artificial Intelligence*, 73(1-2):231–252, 1995.
17. Munindar P. Singh. Toward interaction-oriented programming. Technical Report TR-96-15, 16, 1996.
18. M. Wooldridge, NR. Jennings, and D. Kinny. The GAIA methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*, 2000.