GRID Services as Learning Agents: Steps Towards Induction of Communication Protocols

Michel Liquière, Stefano A. Cerri, Nik Nailah Binti Abdullah

GRID Services as Learning Agents:

Steps towards induction of communication protocols

Nik Nailah Binti Abdullah, Michel Liquiere and Stefano A. Cerri

LIRMM, CNRS and Universite Montpellier II

Montpellier, France

{binti, liquiere, cerri} @ lirmm.fr

Abstract. In this article, we outline some issues in agent interaction on the Web, which is the center point of supporting the needs of fully-realized learning GRID in the future. Of particular importance is conversation support, with its core element, communication protocols. We propose to construct communication protocols through concept learning of services generated on the Web. The proposed approach incorporates a machine learning model into a conversation environment for the induction of communication protocols.

1. Motivation and Scenarios

Real World Scenario. We unfold an important domain. [Clancey, 2004] presented a scenario of research collaborators, scientist who engage in a joint project. The author studies the collaboration between scientist at Haughton Crater in the High Canadian Artic. We have Z working with C on Devon Island, these people bring additional research capabilities to an effort. Each may be specialized in using particular instrument, or doing a particular kind of analysis. In defining joint research, collaborators often negotiate goals-such as (i.e. who will do what, how capabilities and efforts will leverage off of one another). They enter into a (usually informal) contract, or may write a research proposal to define roles and responsibilities. During

this work, collaborators sustain other commitments and participation. Collaborating scientists must negotiate because it is assumed that they retain their individual interests and their contributions will serve multiple, personal purposes. Because of the interests and intelligent capabilities of professional participants, successful collaboration requires negotiation of objectives, methods, roles and schedules [Clancey, 2003]. Handling thousands of large jobs for a big complex like NASA or SDSC computer centers led the managers to create control software and a network to connect scientists to a remote system leading to the GRID. GRID computing refers to computing in a distributed networked environment where computing and data resources are located throughout a network [Jiang et al, 2004].

Certain services must be identified during the course of collaboration to fulfill the needs of the collaborators and in turn maximizing what the GRID can provide. Knowing exactly what to provide to the collaborators as services is not a simple task, as computer users themselves do not potentially know what sort of services the computer systems can provide. One suggestion is to learn dynamically the sort of services that the learning GRID may provide [Cerri, 2003]. This can be initially achieved through tracking the conversational process [Clancey, 2004]; among the collaborators and/or among the communicating softwares.

A successful GRID is an incorporation of an Multi-Agent Systems (MAS) which organizes the collaboration between the participants. An MAS is a system consisting of artificial agents which interact with one another. An MAS can support distributed collaborative problem solving that is required by the GRID by agent collections that dynamically organize themselves having diversified capabilities and needs. Thus, enabling scientists to generate, analyze, share and discuss their insights, experiments and results in an effective manner on the GRID.

During the course of collaboration, interaction emerges and challenges of specifying and implementing agent communication protocols emerges as well [Paurobally et al, 2002]. Not only is learning services an important aspect in this

frame of work, the communication protocol aspects between the artificial agents↔artificial agents and human↔artificial agents must also be given equal attention. Communication protocols ensures smooth operation for the collaborating team. We use the term communication protocols instead of conversational protocols because it conveys a wider meaning, not only in the sense of conversational process but taken into consideration of interaction process.

In an open system that applies an MAS, softwares are called artificial "agents". By definition, an artificial agent is a software entity that is capable of carrying out some set of operations on behalf of the user or another program with some degree of autonomy having some level of intelligence, social ability, and cooperative. These agents resides in an environment working together in solving problems or subproblems interactively with their environment. Artificial agents are autonomous in behavior and autonomy encourages disregard for other agent's internal structure, implying a crucial need to model conversations [Mayfield et al, 1995]. These agents communicate with each other via an agent communication language (ACL).

Our proposed work focuses on two main aspect: 1) *agents' conversations* and 2) *construction of communication protocols inductively*. By definition, conversations are ongoing *sequences* of communicative acts, confirming to one or several protocols [Nzdis, 2000]. However, protocols are complex and dynamic interaction schemas. Normally, issues of interoperability arises for designing a standard interaction schemas.

There are 3 different approaches to modeling agent communication studies. 1) human agent ↔ human agent; 2) artificial agent ↔ artificial agent; and finally 3) human agent ↔ artificial agent. We are currently looking into 2). We hope to design an abstract and generalized system. These agents can serve as a purpose for modeling humans in a deliberately well-circumscribed context [Stefano A.Cerri,2003].

"How can computers help people or what help do people need ?"

-W J.Clancey, 2004

How do we identify services from tracking the conversational processes between collaborating artificial agents? Essentially, Machine Learning (ML) techniques are applied to learn concepts from examples, and we put this into correspondence to learning conversational protocols from examples. ML is a study of computer algorithms that improve automatically through experience [Mitchell, 1994]. For more information on an MAS from a machine learning perspective, please refer to [Peter Stone et al, 2000]. We address some other domains that require ML techniques in agent's communication:

1. According to the work of [Lemoisson, 2004] in a chemists collaboration environment, sharing knowledge of a certain organic chemistry structure requires *a certain protocol* that allows the involved parties to share a certain terminology or concept.
2. Analysis show that an artificial agent need not only be a collaborator, but an assistant-a *remote agent that logs, track, advises and monitors the work* [Grosz, 1994]. We consider a small fragment of interaction between a user and a system (human↔machine context).

   (1)User: We need to repair a connectivity problem between Mars and Saturn. Do a remote ping from Mars and Saturn.

   (2)System: I can't. Saturn seems to be down. *I'll take care of that first.*

   (3)<Systems reboot machine>

   (4) System: Okay, Saturn's back up and remote ping was successful.

   (5)User: Good. Verify. Mars' IP address for Saturn for me.

   (6) System: The entry for Saturn was wrong, but I corrected it.

   (7)User: Okay, good. We're done.

2. **Agents conversations in a society with social protocols**

Agent programs are designed to autonomously collaborate with each other in order to satisfy both their internal goals and the shared external demands generated by virtue of their participation in agent societies [Draa et al, 2002]. The balance between collaboration and fulfilling it's own goals is made by each agent individually and depending on the situation. Due to this autonomy of the agents the collaboration needs a sophisticated system of agent communication. An assumption is made that an Agent Communication Language (ACL) can best handle the issues of communication between agents.

As part of its program code, every agent must implement tractable decision procedures that *allow the agent to be able to select and produce ACL messages* that are appropriate to its intentions [Draa et al, 2002]. By engaging in pre-planned or stereotypical conversations, much of the search space of possible agent responses can be eliminated, while still being consistent with the ACL semantics.

Work on formal accounts of agent conversations remains in its infancy. The theory tries to find a middle way between completely fixed protocols and using high level rules that can generate protocols on the fly. Completely fixed protocols are usually too rigid to be used in an MAS environment or they get too complex (taking into account every possible exception that might occur). However, generating every next step in a protocol based on the present situation is highly computational intensive and therefore not practical for most agent implementations. It seems obvious that large-scale properties of agent conversations, such as overall information flow and the establishment of commitments, are a consequence of the individual meanings of the message that make up the conversation. There are several aspects that needs attention in agent conversations. A designed framework should be able to perform easy monitoring of the progress of a conversation. Also, encouraging the possibility of reusing the structures as *building blocks of complex conversations* [Draa et al, 2002].

When agents join in one or more roles in an environment, they acquire the commitments that go with their individual and social roles. The commitments of a role are restrictions on how agents playing that role must act and, in particular, communicate. *Such requirements requires communication protocols to ensure a non-dysfunctional system* . In figure 2, we have user Darel and Ray communicating at a distance on the world wide web. Behind those walls of interaction, the softwares behind the system is interacting as well, and conversations between these softwares takes place. By learning the conversations between these communicating agents, we can identify types of services to provide to Darel and Ray to further improve their collaboration.

Protocols need to be defined to give a guideline on *how agents should communicate with each other* and *to accommodate the kinds of exceptions* that arise in MAS. Specific protocols should be designed for societies of different applications such as e-learning, electronic commerce, travel applications, industrial control, logistics and student registration to function well. Current initiative to construct protocols are normally being predefined. It is very unlikely, that all protocols and their exceptions can be predefined without a formal definition and a centralized language. We suggest to tackle this problem by performing an induction learning of the exchange protocols; allowing them to know which specific protocols to adapt to during different context of conversations.

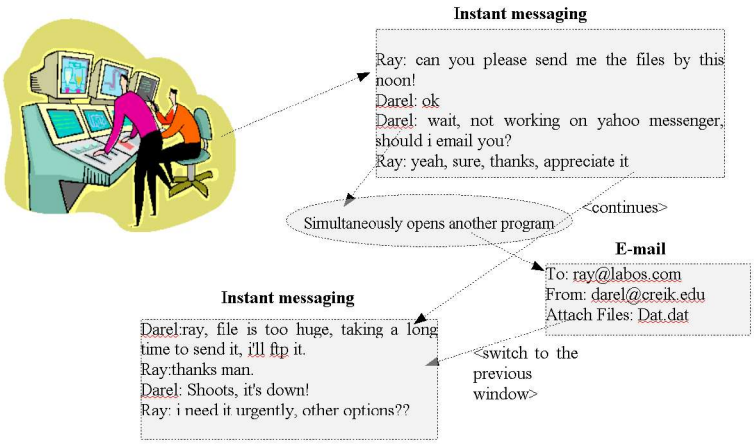

Figure 2: A chaotic virtual world

3. Concept learning of logs of message exchanges

MAS Scenario. Agent1 contacts Agent2 about defining job roles in a research group. They engage in a conversation about Agent1's preferences, Agent2's inventory, and so forth. For protocol, they agree to use a modified Agent Communication Language, in which each message contains one of a half-dozen standard performatives (i.e. denotes the type of the communicative act of the ACL message) to identify the intent of message, and message contents follow a standard define-your-role ontology. At some point, Agent1 volunteers the information that it would be willing to take up extra hours as a group motivator if being paid another extra 20 euros for each hour. This uses a non-standard perfomative. Agent2 cannot process the non-standard performatives, so it replies as "not understood". The negotiation continues as if nothing had been said. Some time later, Agent2 asks whether Agent1 wants to upload a one page cv. This term is not in the ontology Agent1 is using; so it contacts an ontology server to find out about the term, to be told that it relates to "documents" that

contains “professional activity”. Agent1 looks in its fact store that no information about cv, and no special overrides have been added for this negotiation, so it replies with “no”. The transaction continues and eventually completes to the satisfaction of both parties. Corresponding to the above scenario, in Figure 3, the Agent1 would reside at the client space and Agent2 would be at the service space. The scenario above has been taken and modified from [Hanson et al, 2002].

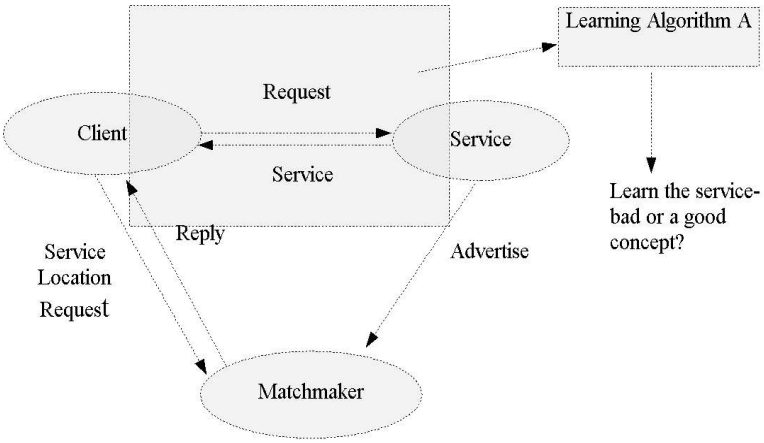

Figure 3: Prototype of grid computing services [Cybenko et al, 2002].

3.1 Learning the concepts : A model

We show fragments of possible XML messages exchange between the service and client which we made some modification from the work of [Hanson et al, 2002] and [Glushko, 1999]. These are however, well-defined fragment of messages.

```
<Roles>
   <Role>Client</Role>                                    1
</Roles>
- <Roles>
   <Role>Service</Role></Roles>
```

```
-<InitialState>                                                    2
    <IniStateName>Start</IniStateName>
</InitialState>
-<State StateId="Start">
   -<SendMessageTransition TransitionName="RequestConversation">
    <Target>ConversationRequested</Target>
   <Sender>Client</Sender>
   <Event>SendMessage</Event>
  -<Message>
      <service.op.name>Submit Job Role</service.op.name>
     <Schema>RequestConversation.xsd</Schema>
   </Message>
 </SendMessageTransition>
```

```
</State>
-<State StateId="ConversationRequested">                           3
  - <SendMessageTransition TransitionName="AcceptConversation">
      <Target>ConversationAccepted</Target>
      <Sender>Service</Sender>
     <Event>SendMessage</Event>
   -<Message>
       <Encoding>xml-document</Encoding>
            (ask-one
            :content  (DEFINE ROLE ?role)
            :reply-with role-definition
            :ontology  COLLABORATION-ROLE
```

```
        <Schema>AcceptConversation.xsd</Schema>
   </Message>
</SendMessageTransition>
```

```
</State>
 -<State StateId="ConversationAccepted">                    4
  -<LoadChild>
     <Sender>Client</Sender>
       <Policy>MetaConversation-2.1.xml</Policy>
          (request-one
           :content ( group motivator if receive 20 euros extra)
    <ChildReturn>Done</ChildReturn>
  </ChildReturnTransition>
```

```
</State>
 -<SendMessageTransition TransitionName="Refuse">           5
   <Target>ConversationOver</Target>
   <Sender>Service</Sender>
  <Event>SendMessage</Event>
  -<Message>
       <Encoding>xml-document</Encoding>
          (tell-one
          :reply-with not-understood
   </Message>
  </SendMessageTransition>
 </State>
-<State StateId="ConversationOver">
```

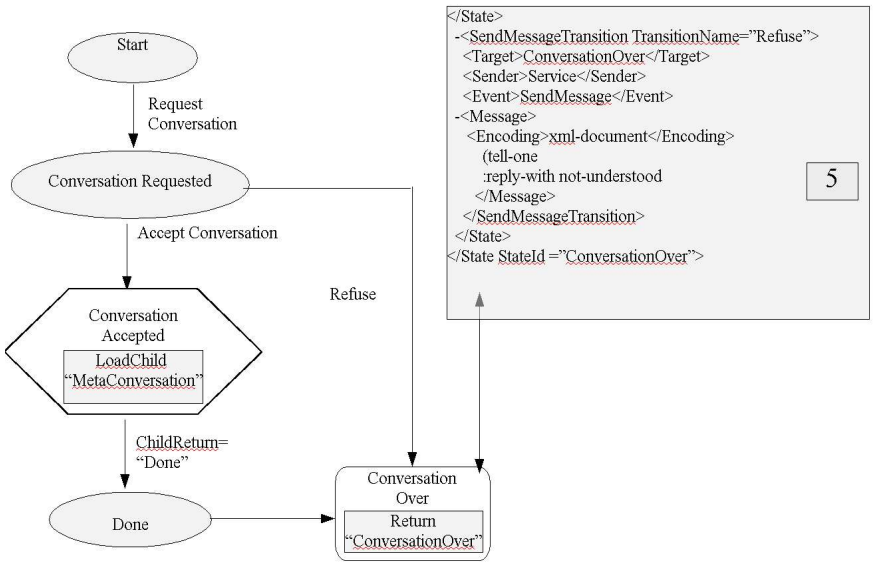

Figure 4. An example of a communication protocol corresponding to a fragment of XML message.

In figure 4, we demonstrate how a concept such as “conversation over” corresponds to the XML message labeled (5). Certain states in the communication protocols can be viewed as concepts; such as “conversation requested”, “conversation accepted”, “conversation over”. However, some performatives may not be recognized in these concepts. In our work, the XML messages will be less well-defined as compared to the above, omitting the stateId and Target.

In fragment (3) and (4), the client request something which was not defined in the standard-performatives and thus conversation during this particular context has terminate as if nothing took place. For example, our target concept could be “the context when the <service> respond with a message 'not understood' ”. This is based on the studies of concept learning [Mitchell, 1997]. Unexpected messages may turn to be valuable, because they may contain clues as to how they should be handled. By learning a “bad message”, we identify those not existing as a standard performatives

and validate, explain and if possible to update them. Of course there are many other negative performatives such as reject, refuse, and failure. When a message such as "not understood" is being received, we store it together with the content. Now, we shall have a set of a bad and good instances of reply (i.e. Reply- with not-understood).

Since concepts can be arbitrarily complex subsets of feature space, an important issue is the choice of the concept of description language. The language must have sufficient expressive power to describe large subsets succinctly and yet be able to capture irregularities. Our suggestion is using a structural description language [Liquiere et al, 1998].

4. Designing Protocols

The problem in designing protocols is in developing a practical, common sense set of rules that is efficient to use under circumstances and that allowed for a safe recovering from unexpected events.

Protocols require:

1. Precise format for valid messages (a syntax).
2. Procedure rules for data exchange (grammar).
3. Vocabulary of valid messages that can be exchanged, with their meaning (semantics) [Holzmann, 1991].

The grammar of the protocol must be logically consistent and complete; under all possible circumstances the rules should be prescribe in unambiguous terms what is allowed and what is forbidden in order to maximize the best performance in collaboration acts.

We summarize below some of the predominant methodologies used for the construction of communication protocols.

State of the Art	Pros	Cons
Statecharts [Harel et al, 1998]	1. Easiest to express protocols. 2. Less cluttered diagrams.	1.Difficult to show compound transitions for nested protocols and their results. 2.Undefined states and conflicts between states may arise.
Petri Nets [Cost et al, 1999]	1.Can detect conflicts and their properties. 2. Graphical modeling.	1.Redundancy in repeating the same parts of a protocol for different agents or roles. 2.Alternative actions such as agree or reject or both cannot be expressed.
AUML [Odell et al, 2000]	1.Visual representation along timelines. 2. Reuse of UML constructs.	1.Requires effort in expressing protocols for realistic complexity for developing, debugging and understanding.

Table 1: Comparison between different methodologies used for construction of communication protocols.

A detailed critique can be found in the work of [Paurobally, 2002]. [Huget, 2003] proposed a protocol model known as "interaction protocol engineering" which is based on a communication protocol engineering [Holzman, 1991]. The model allows the definition of protocol to be designed from the start. The authors use an informal document to define all the features that a protocol needs. Currently, there does not exist any algorithm nor methodology that help designers write the formal description of a protocol given its specifications [Huget et al, 2003]. Their work provide as a background for us to develop an inductive communication protocol. We consider the work of [Huget et al, 2003] because of the re-usability aspect of the micro-protocols and thus reducing effort of re-designing communication protocols.

The aim of their "interaction module" is to handle protocols and to manage interaction between agents. Several issues arise in the selection or "firing" of protocols, in their work, the authors look into the current state that an agent is in. They used a conjuction of first-order predicate that needs to be evaluated to true in order for the formula (i.e. protocols) to be used. The authors also ensures that the formula is deterministic. However, in any real-time case, since agents are autonomous, some actions and internal plans of agents are non-deterministic thus some protocols may cause exceptions.

4.1 Requirements of an agent communication protocols

We have briefly encountered the state of the art for the construction of communication protocols. Now, we shall define some of the important needs of an agent communication protocols; which are in our opinion:

1. Consistent
2. Unambiguous
3. Interactive
4. Adaptive
5. Capable of solving state of conflicts between protocols (i.e. Shifting and

firing protocols)

6. Explanatory

Referring to the example when the Agent2 responded with “not-understood” message to Agent1, this agent1 must be able to know and explain why his previous internal goal was not fulfilled? We term this as “explanatory”. Based on these requirements, we conclude to model a learning algorithm which consists some of these elements:

1) concept learning: ability to distinguish between the bad and the good set of messages. Are there any errors in the classification?
2) Adaptive computation: allows complex interaction parts and can adapt to the environment.
3) Grammatical Inference: providing the general framework of processing grammars
4) Shifting and firing mechanism: ability to know which concept of knowledge should be shuffled and then fired during a conflicting situation.

4.2 Randomly generating protocols by the rule of thumb

Continuing from defining the bad and good instances; we now enter into a stage where we inductively construct communication protocols based on the hypotheses which we can derive from the collected instances. In fact the hypotheses correspond to the construction of communication protocols. These hypotheses are created in an adaptive environment which can evolve and shall go through generation process.

The key question is how do we know which protocols to fire? In the previous section we discuss of our shifting and firing mechanism. Even though concepts are shuffled according to its' different level, some intersection of these concepts may occur, thus proposing us to also add a firing mechanism for precision. We propose to introduce a random component (firing order) during this process, which we borrow

from the idea of genetic algorithms [Goldberg, 2003]; and hope to incorporate this in our later stage of research studies.

5. Conclusions

We begin our study by analyzing people interacting in a collaborative environment and later relate it to artificial agents interacting on the Web. We suggest that conversation support is vital in any interactive environment that employs different artificial agents and each interacting to fulfill their own goals. In particular, we discuss issues that normally arises during interaction between a service and a client agent. Although, some communication protocols have been established for these purpose, none however focuses in improving the conversations itself among these agents. For example, communication protocols are generally to ensure that the agents abides to a certain "rule" during communication, however little attention is given to unexpected messages. As a consequence, the core issue of finding out why certain malfunction interaction erupts goes unstudied. We have suggested to use a machine learning technique that initially learns the conversational examples of interacting agents and identify which are the bad and good ones. Experimentation will be done in the near future once the tools are made available. Later, we shall hope that these findings provide us as a mean to construct communication protocols inductively.

References

Cerri, S.A. 2003. Open Learning Service Scenarios on GRIDs. 3rd International LeGE-WG Workshop: Grid Infrastructure to support future technology enhanced Learning, Berlin.

Cerri, S.A, M.Eisenstadt and C.Jonquet. 2003. Dynamic Learning Agents and Enhanced Presence on the GRID. 3rd International LeGE-WG Workshop: Grid

Infrastructure to support future technology enhanced Learning, Berlin.

Clancey,W.J. 2003. Agent Interaction with Human Systems in Complex Environments: Requirements for Automating the Function of CapCom in Apollo 17. AAAI Spring Symposium on Human Interaction with Autonomous Systems in Complex Environments, Stanford.

Clancey, W J. 2004. Roles for Agent Assistants in Field Science: Understanding Personal Projects and Collaboration. IEEE Transaction on Systems, Man and Cybernetics-Part C: Applications and Reviews. 2d eds. 34 vols.

Cost,R , Y.Chen, T.Finin, Y.Labrou and Y.Peng. 1999. Modeling agent conversations with colored petri nets. In Workshop on Specifying and Implementing Conversation Policies.59-66

Csuhaj-Varju, E., J. Dassow, J. Kelemen and G. Paun. 1994. Grammar Systems: A grammatical approach to distribution and cooperation. Gordon and Breach science publishers.

Cybenko, G., G.Jiang and D.Bilar. 1999. Machine Learning Applications in Grid Computing. 37th Annual Allerton onference on Communication. Control, and Computing. 348-357.

Draa, B.C, and F.Dignum. 2002. Trends in Agent Communication Language. Computational Intelligence. 2d eds. 5 vols. Blackwell Publishers.

Glushko,R.J, J.M. Tenenbaum and B.Meitzer. 1999. An XML framework for Agent-Based E-Commerce. Communications of the ACM, volume 42, no.3.

Goldberg, D.E. 2003. Genetic Algorithms in Search of Optimization and Machine Learning. Addison-Wesley.

Grosz, B.J. 1994. Collaborative Systems. AAAI-94 Presidential Address, AI Magazine.

Harel, D. and M.Politi, 1998, Modeling reactive systems with statecharts. McGraw Hill.

Hanover, V. 1999. Intelligent Agents and Multi-Agent Systems. A tutorial presented at the IEEE CEC.

Hanson, J.E, P. Nandi and D.W. Levine. 2002, Conversation-enabled Web Services for Agents and e-business. Proceedings of the International Conference on Internet Computing (IC-02), CSREA Press. 791-796.

Holzman, G.J. 1991. Design and Validation of Computer Protocols. Prentice-Hall.

Huget, M.P and J.L. Koning. 2003. Interaction Protocol Engineering. Communications in Multiagent Systems,LNAI 2650. Springer-Verlag.179-193.

Jiang, G. and G. Cybenko. 2004. Functional Validation in Grid Computing. Autonomous Agents and Multi-Agent Systems. Kluwer Academics Publishers.8:119-130.

Lemoisson,P., E.Untersteller, M.A Nunes, S.A. Cerri, A.Kreif and F.Paraguacu. 2004. Interactive Construction of EnCOrE (Learning by building and using and

Encyclopedia). technical report, EleGI.

Liquiere, M, and Jean Sallantin. 1998. Structural machine learning with Galois lattice and graphs. International Conference on Machine Learning (ICML).

Paurobally,S. and J.Cunningham. 2002. Achieving Common Interaction Protocols in Open Agent Environments. AAMAS, Melbourne, Australia.

Mayfield,J., Y.Labrou and T.Finin. 1995. Desiderata for Agent Communication Language. AAAI Symposium on Information Gathering from Heterogeneous, Distributed Environment. AAAI-95 Spring Symposium, Stanford University, Stanford, CA. 27-29.

Stone,P. and M.Veloso. 2000. Multiagent Systems: A Survey from a Machine Learning Perspective. Autonomous Robotics. 8th ed. 3 vols.

Mitchell, T. 1997. Machine Learning. McGraw Hill International Editions.

Moore, J.H and L.W.Hahn. 2003. Grammatical Evolution for the Discovery of Petri Net Models of Complex Genetic Systems. Genetic and Evolutionary Computation Conference. 2412-2413.

NZDIS team, 2002, http://nzdis.otago.ac.nz/download/slides/nzdc00-agents.pdf

Odell, J, H.Van D.Parunak and B.Bauer. 2000. Extending UML for Agents. Proc. of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence. Gerd Wagner, Yves Lesperance, and Eric Yu eds. Austin. TX. 3-17.

Pauw, G.D. 2003, Evolutionary Computing as a Tool for Grammar Development. Genetic and Evolutionary Computation Conference. 549-560.