Using BDI-Oriented Agents for Network Mangement

Morsy M. Cheikhrouhou
Corporate Communications Department
Institut Eurécom
BP 193 – 06904 Sophia-Antipolis Cédex – France
Morsy.Cheikhrouhou@eurecom.fr
Tel: +33 4 93 00 26 48

Abstract

This paper presents a novel approach to foster network management automation. Our approach is based on the use of intelligent agent technology as a means to provide feedback control of the managed network. The intelligent agents we use are BDI oriented. They are based on beliefs, capabilities, motivations, intentions and goals. Their mental model is first presented, then the way it can be applied to network management is discussed. We show how such agents are suitable to design network management solutions that automate network control based on monitoring results. This is achieved by considering the managed network as a control system, and the Intelligent Agent as a feedback controller.

Keywords Network Management, Intelligent Agents, Belief-Desire-Intention.

1 Introduction

Network Management can be divided into two major activities: monitoring and control [1]. However, current NMSs focus mainly on (and rarely go beyond) monitoring and problem reporting. Some systems provide analysis facilities which help the administrator to locate or to repair more easily network problems. At the level of management control operations, most of the management platforms provide facilities to perform control actions on the network. In general, they provide their own tools to perform such tasks as setting MIB (Management Information Base) variables using SNMP-Set primitives. For example, a firewall usually provides an interface by which the network administrator can control security parameters in his network.

However, these facilities are not enough to automate network control. The administrator still needs to launch himself the actions on the network. Actually, the most difficult task consists in finding what actions to take when the network monitoring reports problems. This task is a decision taking which relies on the analysis of the causes of the problem and which must consider the actual state of operation of the network. For instance, depending on the services installed on a machine, the action to be taken when this machine hangs may vary from nothing (suppose it is a stand-alone PC) to having to reboot it (in the case of server failure for example).

Moreover, the administrator cannot monitor everything in the managed network. What actually happens, is that the network users are those who alert the administrator of performance and fault problems. For this reason, some management platforms introduce the possibility to associate actions to some events by means of triggers or rules. However, the actual use of this possibility is to display alert messages on the console display, even though management operations could be ascribed to events.

In fact, linking monitoring results to control actions is what is missing to automate network control. Control operations cannot be planned and executed without considering monitoring data. Current NMSs do not offer efficient means to express this dependency. Of course, rules and triggers are only simple means and are not efficient to handle the complexity of automating network control. Alone, rules cannot satisfy the requirements needed to automate network management which will be described in section 2. In this paper, we propose an approach based on the concept of Intelligent Agent (IA). Since many IA philosophies exist in the literature [2], we specify in section 3 which agent philosophy we adopt and we present its concepts. In section 4 we detail the agent operational model we adopt for our approach to network management. Section 5 presents how our intelligent agents can be

used to automate network control when deployed as feedback system controllers. In section 6 we show the advantage and the challenges of our approach. Finally, we end the paper by concluding remarks and future work.

2 Requirements for Network Management Automation

We identify three requirements to establish the bridge between monitoring operations results and control operations. These requirements also explain the gap between network control and monitoring that is illustrated in Figure 1.

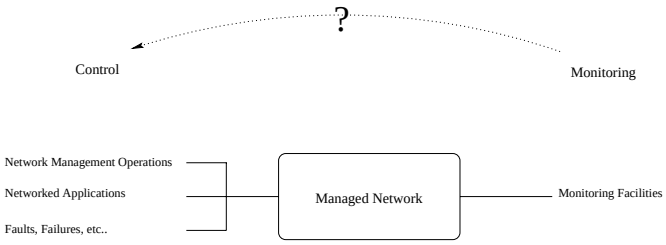

Figure 1: Current Network Management Systems

1. To help the NMS detecting problems, it is essential to provide it with information that makes it able to distinguish problems from the normal operation of the network. The human network administrator needs also to specify management requirements that the NMS has to achieve. The network management lacks *an explicit and complete formulation* of what is expected to be a good behavior from the network. Indeed, this information is only partly specified by the network administrator in the form of thresholds, minimum or maximum values of some management parameters. The problem with this situation is that these threshold values are spread throughout many parts of the network management system and information database and that they hardly can be maintained to evolve with the user requirements from the network. Therefore, the NMS should offer the possibility to express in an explicit and maintainable fashion what is expected from the network when it is behaving well.

2. Current efforts in standardization organizations are mostly directed toward the integration of management by providing integrated management information models (CIM, JMAPI, etc.) [3, 4]. However, since the NMS should have the responsibility to plan and perform management operations, it needs explicit formulation of the semantics of these operations. Currently, there are only few research efforts that implicitly make use of modelling of management operations. However, these works are mostly carried out for simulation and test generation purposes (Look for example at [5, 6]).

3. The high dynamic of the network (see section 5.1), the wide range of possible control operations and the complexity of the decision taking process make difficult the automation of network control. Therefore, we require a sophisticated paradigm to handle this complexity. This paradigm should offer powerful concepts that allow to describe complex automation problems.

Our approach to overcome this lack of automating network control is based on two converging ideas. The first idea consists in considering the network and the management system as a feedback control system. The managed network is the controlled system and the management system is the controller. The role of the controller will be to monitor the status of the managed network, to compare it to an explicited formulation of its normal behavior and to take the corrective actions when need be.

The second idea consists in using Intelligent Agent technology as a natural and powerful way towards the concretization of the first idea. In effect, one kind of agent architectures called BDI architectures provide powerful abstraction means to handle complex control problems. These abstraction means are called **mental categories**. Mental categories are used to describe the mental state of the agent which helps in modelling and interpreting its behavior. There are excellent references describing BDI agent architectures such as [7, 2]. The next section introduces relevant concepts of BDI theory for the purpose of the paper.

3 Mental Category Based Agents

Agents theories that are based on the mental categories has their origin from works in the Artificial Intelligence domain that focus on the understanding of the essence of actions in human behavior so as it can be mapped to intelligent software. One of the most interesting results of such works ([8]) lead to the introduction of *mental categories* (or *mental attitudes*) to describe, understand and analyze the state of the agent and its past and future behavior. Notions such as beliefs, desires, intentions, knowledge, commitments, etc. were used for such purposes. A mental attitude is a position taken by a human or

an agent towards a statement or an expression over the world. For example, an agent or a human can *believe* a statement, or *desire* to have that statement holding in the future.

Among the most popular agent architectures based on mental categories is the one called BDI (Belief, Desire, Intention) architecture [9, 8]. This architecture is based on the notions of **B**elief, **D**esire and **I**ntention. In fact, most of agent theories that are based on mental categories are called BDI-oriented or BDI-like theories. In general, BDI approaches are based on a set of mental categories with defined semantics and a control architecture that defines the agent's mental cycle. The **mental cycle** is the process that rationally selects its course of action based on these mental categories ([10] cited in [11]). We provide two simple examples showing how a mental cycle may be designed.

The first example is taken from [9]. The agent architecture is based on belief-desire-intention mental categories. The mental cycle is composed of three processes. The option generation process waits for events perceived as beliefs, determines which are the relevant according to the current desires the agent has and generates a set of options. The deliberation process selects the set of options that the agent believe to be pertinent for achievement. Finally, the selected options are considered as intentions and an execution process works out for their achievement.

The second example is Shoham's *agent-oriented programming* language Agent0 [12]. Agent0 is based on a set of mental categories composed of beliefs, commitments and capabilities. Capabilities are what the agent *can* potentially carry out, i.e. the set of actions the agent can perform. Commitments can be seen as the agent's obligations. The mental cycle is carried out with behavioral rules [13] that map agent beliefs to directly executable commitments that invoke its capabilities.

Several mental attitudes were defined by people adopting this approach in IAs. Müller [7] informally defined a set of these mental attitudes. In the following, we report the definitions that are most relevant to the remainder of the paper.

- **Belief**

  According to [7], beliefs are informally defined as "*the agent's expectations about the current state of the world and about the likelihood of a course of action achieving certain effects*". Therefore, agent's beliefs map its perception of the world. With a more general view, the beliefs of an agent must also include its perception of the state of possibly other agents and of its own state. Beliefs on its own state are essential to allow the agent to have control over itself, while the beliefs on the statuses of the other agents is required to allow the agent to reason about the opportunity of cooperating with some of them.

- **Motivation**

  The motivation attitude is introduced as a source of stimulation that lets the agent act on the world. In [14], the motivation is defined as a "*driving force that arouses and directs [agent] action toward the achievement of goals*".

- **Goal**

  A goal denotes a state that the agent wants to achieve through the execution of a certain plan of actions [14].

- **Commitment/Intention**

  Commitments describe what the agent committed or decided to do. When the agent commits to doing something, it decides how long it will persist in doing the corresponding action and under which circumstances it can be dropped [15].

- **Capability**

  The agent is an active software entity. The set of actions the agent is able to perform can be classified into action types. A capability defines a certain type of actions the agent may need to execute so as to let it know how and when it is appropriate to execute such actions. This definition adheres to and extends that adopted in [13].

There can be several other mental categories such wishes, dislikes, and human like emotions. However, we believe that such mental notions are not of a practical use in domains such as network management. (See [2] for more details and references on such human-like mental notions.)

4 An Operational Mental Model for Intelligent Agents

The agent community agrees that there is currently no widely adopted definition of an intelligent agent. Therefore, and for precision sake, we provide our own definition of what we consider an agent. "*An agent is a reactive software [16] that acts on the managed network autonomously in order to satisfy the management requirements specified by its user.*"

We insist on the fact that management requirements are expressed in a high-level fashion and do not specify what actions the agent has to do to satisfy them. Indeed, by 'autonomous

action' we mean that it's up to the agent to determine the sequence of actions to carry out without the direct help of the user.

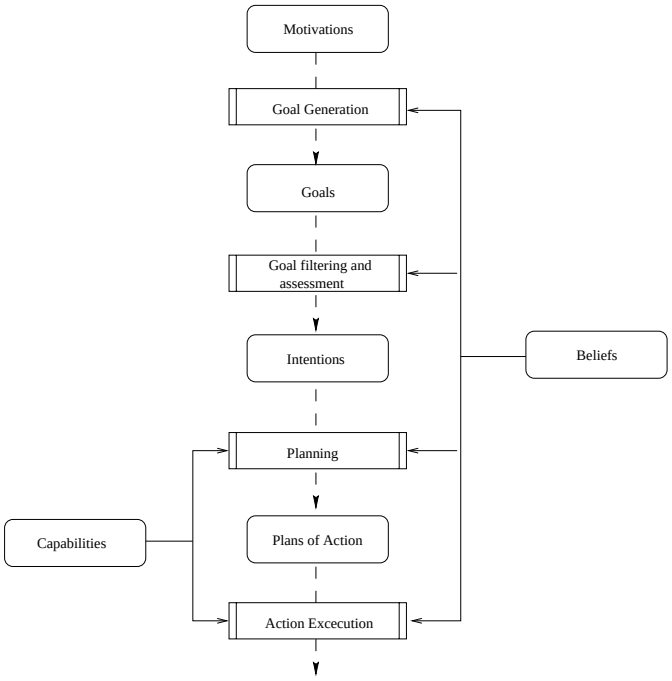

Figure 2: Agent's Mental Model

We present hereafter a mental model based on the following mental categories: beliefs, capabilities, motivations, goals and intentions. We refine the general definitions presented in the previous section and describe the way these mental categories can be applied to network management.

4.1 Beliefs

In the context of network management, beliefs can translate the network's configuration as well as its current state of operation and performance including bandwidth usage, servers' states, etc. Depending on the kind of problems that the agent is deployed to solve, its beliefs may range from simple performance measurements to complex expectations on the present and future behavior of the network.

4.2 Capabilities

In order for the agent to assess the applicability of a certain action, it requires the following information in its capability:

- The action's preconditions which express the domain of applicability as well as the consistency conditions that must be satisfied before the action can be executed. For example, the action of shutting down a machine requires that no user is connected to this machine and that there are no applications running on it.
- The action's effects which express the impact of applying it. The effects are used by the agent to determine the appropriateness of executing that action for a certain purpose.
- The action's cost in terms of CPU usage, execution time, bandwidth and preference. The preference indicates how good or bad using this action is. For example the `process kill` action is probably a bad action and should only be used in extreme cases.

Regarding Network Management, we have identified four kinds of capablities:

1. **Sensors**

   Sensors allow the agent to perceive some parameters of the world and provide the agent with the necessary beliefs. They are also used to maintain its beliefs coherent regarding the possible evolution of the world. Therefore, sensing actions correspond to the monitoring activity of an NMS. There is a first kind of sensors which can be qualified as persistent. The agent may have persistent sensors which continueously sense some parameters in the network. However, this kind of sensors can be used in two different ways. In one way, these sensors can be used asynchronously, i.e. to sense asynchronous changes in the network. A simple example for the network management activity is an SNMP Trap sensor which asynchronously receives traps from deployed SNMP agents and updates the intelligent agent's beliefs according to the traps. In a second way, a persistent sensor may be sollicited synchronously by the agent. Suppose that the agent has a special traffic measurement tool on some physical link. The measurement tool can be considered as a persistent agent sensor which communicates information only under the request of the agent. For example, one may imagine a particular sensor that computes the traffic rate generated by a certain application over time. Such sensor needs to run continuously in order to be able to compute the average rate. The agent may need only to get informed about the traffic rate value at random instants and not to get notified of each change in the value of this parameter.

   Persistent sensors may be configured by the agent on the fly while they are running. A typical example is to

configure a persistent sensor to a particular threshold on some parameter, beyond which a belief is updated or created.

The second type of sensors can be qualified as non persistent (or functional). An agent may temporarily activate a non-persistent sensor in order to know the value of some managed data from the network. A typical example of this kind of sensors is the SNMP-Get sensor, which is used by the agent to "sense" the value of an SNMP MIB variable.

2. **Effectors**

   Unlike sensors, there is a unique kind of effectors: non-persistent effectors. Indeed, effectors allow the agent to make changes on the configuration parameters of the network and hence to modify the behavior of the network.

   In fact, one may imagine one kind of persistent effectors that perform actions under some conditions. However, we choose to call them reactors rather than effectors as explained in the next section.

3. **Reactors**

   Reactors are simple means that allow the agent to perform actions when certain conditions or events occur. Such reactions are necessary for the agent to have prompt behaviors related to certain critical situations that may happen in the network. As an example, the agent may set a reactor that under the situation where an Ethernet card in a host continuously generates traffic on an Ethernet till it blocks the other hosts on the same link, turns off the corresponding slot in the hub to which it is attached.

4. **Calculators**

   This is a particular kind of actions that allow to compute new beliefs from other lower level beliefs. They can be seen as a function that takes low level beliefs as input, and provides more elaborated beliefs at the output level. Calculators are useful for network management purposes not least to perform usual computation like statistical calculus.

4.3 Motivations

Motivations are the agent's preferences about the state of the network and its behavior. The network administrator expresses the desired state of the network as motivations inside the agent. Then it's up to the agent to work out in a way to satisfy these motivations. This necessitates the creation of goals and determining which actions can achieve them and which is the appropriate time to execute these actions.

Concretely, a motivation consists of the desired expression on the network as well as of a priority and a 'valid-until' condition. We believe that three classes of priorities can be sufficient for NM purposes: *must* telling that the agent has no choice than satisfying this motivation; *good* telling that it would be better if such motivations are satisfied; and *if-possible* telling that the agent should try to satisfy these motivations only when all the motivations of the first two classes are satisfied.

4.4 Goals

During the process of 'goal generation', the agent considers the unsatisfied motivations and determines which goals that if achieved lead to satisfy these motivations. Therefore, the agent generates goals that direct its course of actions towards satisfying its motivations. The generation is asynchronous in the sense that as soon as the agent realizes that one or more of its motivations are no longer satisfied, the process of goal generation is activated.

When a goal is first generated, a priority and a deadline are attributed to its expression. The priority is function of the motivation(s) that caused its generation. The deadline indicates the date before which the goal must be achieved.

4.5 Intentions

The agent may have many goals to achieve at the same time. The number of goals may exceed the agent's own capacity. Furthermore, the set of goals the agent has to deal with may contain redundancy and contradiction. This is due to the fact that the motivations of the agent are settled and tuned-up independently. Therefore, the agent cannot look after each goal achievement directly after its generation. A process of goal assessment, optimization and scheduling is required. This process considers the newly generated goals and compares them with the set of the currently pursued goals. In case of a contradiction, goal priority is applied to discard contradicting goals. In case of redundancy, redundant expressions of the goals are combined together. Finally, priority and deadlines are used to sort the goals in case where the number of active goals exceeds the agent's capacity. The result of this goal filtering process is a set of intentions towards which the agent has committed for achievement.

Once an intention is adopted, the agent has to plan for its achievement. The planning process takes into account the agent's current beliefs and its capabilities, and produces a set

of actions which will accomplish the intention when they are executed.

Finally, the generated plans of action are ready to be executed. At this stage, the agent is acting on the managed network and produces the necessary regulations.

5 Network Management and Control Systems

5.1 The Managed Network as a Dynamic System

A computer or telecommunications network is a highly dynamic system. Its dynamics can be attributed to the fact that a network is composed of a huge number of more or less complex software and hardware components. Most of these components operate on very low-level data which transit in an increasingly high speed. If we consider as 'network inputs' any action or event that may affect the status of the network or its behavior, then we find three classes of inputs (see Figure 1):

1. There is a class of inputs that result from the usual operation of the network and its normal usage. For example, each connection request leads to the transmission of signalling data on the network thus leading to a change in its status. Considering the explosion of distributed applications and the proliferation of networked client-server applications, nearly any usage of such applications leads to data transmission and contributes to the evolution of the network status. Naturally, the responsible of such kind of inputs is composed of the whole set of network users and customers.

2. Another class of inputs comes from the configuration operations that can be performed on the network components. Each component of the network has configuration parameters which values affect its own behavior and therefore of the whole or part of the network. The usual responsible of such kind of inputs is the human network administrator or provider.

3. Finally, the last class of inputs consists of the failures and faults that may occur on network components. It is known that such failures may have random effects on the network and that these effects may seem too distant from the failure's nature or physical location.

Inputs from these three classes concurrently occur during the network operation and contribute to the perpetual changing of its status.

The network output is simply its own status and behavior. It can be more or less sufficiently represented using different parameters resulting from the monitoring operations such as performances parameters and fault reports.

5.2 Agents Performing Feedback Control Network Management

The network is a perfect candidate to be managed using a feedback controller. In effect, a feedback controller has to keep the regulated system in a set of desired states. It has to capture the current status of the controlled system using sensors. By comparing the current status to the desired status, it causes changes to the controlled system configuration so as to bring it back to a normal status. These changes are made via effectors. These principles are presented in Figure 3 and detailed in the following subsections.

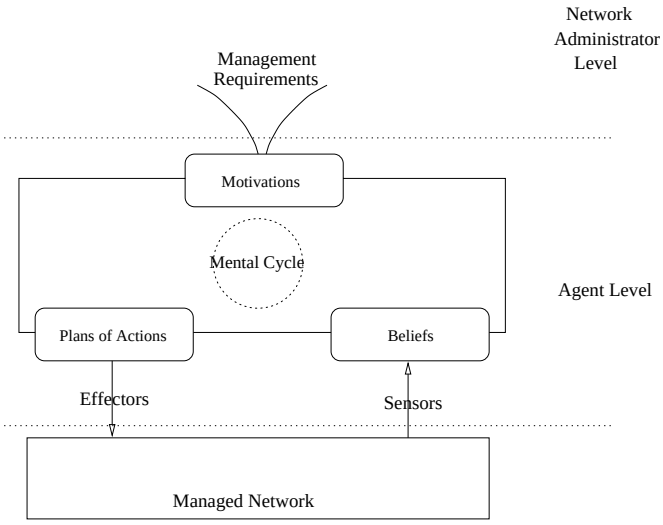

Figure 3: Network Management with Agents as Feedback System Controllers

5.2.1 Explicit Formulation of the Desired Network Behavior in Motivations

The NMS using IAs such as described in section 4 will regulate the network according to the network administrator requirements. The administrator explicitly specifies the way in

which the network is wanted to behave in terms of agent motivations.

According to what the agent is motivated to do, it instruments the necessary beliefs that allows it to check whether its motivations are actually satisfied or not. This implies that the agent will not instrument all the management information in the network all the time as current NMSs helplessly do.

5.2.2 Perceiving the Network Status in Beliefs

To instrument its beliefs, the agent activates and configures the necessary sensors. Sensors will be responsible for the asynchronous update and creation of the beliefs they are activated for. The agent also may use calculators to obtain more elaborated beliefs such as statistical computations.

5.2.3 Performing Regulations with Effectors and Reactors

By performing the mental cycle described in section 4, the agent generates the plans of actions that perform regulations on the network. When these plans are executed, they invoke effectors that change the configuration parameters in the network. They may also include the activation of reactors that allow to have immediate reaction to events that occur on the network.

6 Network Management with Intelligent Agents

Intelligent Agents are introduced in Network Management to provide a higher level of automation of management tasks. The deployment of agents for this task can be done in two phases which are actually dependent of each other. The first phase focuses on the management problem we want to tackle while the second phase puts emphasis on how to build the suitable agent for that problem.

6.1 The Network Management Problem

To provide an agent-based solution to a problem, four main tasks must be carried out.

1. The first task consists in analyzing the management problem. The aim is to define the information system required to solve this problem. The information system is then translated into a belief model made of belief templates.

2. Afterwards, the capabilities the agent requires to interact with the network must be developed. The set of sensors, calculators, effectors and reactors must be as complete as possible so as to allow the agent to perform management tasks in an efficient way.

3. At this stage, the agent can be launched. However, in most of the cases, it must be endowed with the initial beliefs required to start its operation. Such beliefs may include agent related information such as its name and location as well as some network related beliefs such as the network configuration.

4. Finally, during the agent operation, the human operator interacts with the agent mainly in terms of motivations which describe how the network is desired to be.

6.2 The Practical Deployment of the Agents

The mental cycle described in section 4 did not specify which goal generation algorithm or planning tool are used. In fact, in our sense, a unique algorithm for goal generation is not enough to ensure efficiency for all the network management problems. Instead, our approach is to provide a set of 'plug-and-play' algorithms that can be integrated into the agent's code. Therefore, for goal generation, the agent developer may be supplied with algorithms ranging from simple and reactive verification of the motivations to optimized and proactive goal generation tools that is based on beliefs predicted by the agent. Also, the planning tool may vary from a simple search engine in the capabilities database to an optimized tool that generates the optimal plan using dedicated heuristic functions.

Moreover, agent capabilities can be stored in a library and can be used on demand by the agent developer. For example, one may supply a set of capabilities that perform SNMP management operations. These capabilities can then be plugged into agents after ascribing the right preconditions and effects that are meaningful for the current management problem.

Finally, the problem of agent location should be addressed. The question is whether the agent should be on a separate management station, or be close the managed resources, etc. The answer to this question depends on many parameters including the agent programming language and the nature of the management network elements. If the agents are written

in Java, then they must run on a Java-enabled machine. In the near future, we may hope to find network elements directly supporting java programs and therefore, agents can be settled upon them.

7 Conclusion

In this paper, we have presented an approach to enhance network management automation. Current NMSs cannot go far in automating network control because they do not have the infrastructure and tools to allow for enhanced and high-level management. Namely, current NMSs do not include an explicit formulation of what is expected from the network and lack any model that describes the management operations. Our approach is mainly based on the use of intelligent agents with mental-based architecture. Mental categories allowed us to provide the agent with high capabilities of decision taking as well as a natural way to model its interaction with the network. Such agents can then be used as feedback system controllers that are able to link the monitoring results perceived by sensing operations to control operations performed by effecting operations. In this way, the task of the network administrator can be reduced to the tuning of the desired network behavior expressed in high-level abstract motivations.

Our future work will be directed towards the design and implementation of these ideas using intelligent agent technologies. Other team members inside the project are investigating agent cooperation techniques which could be very useful for network management scalability and distribution issues.

References

[1] William Stallings. *SNMP, SNMPv2 and RMON, Practical Network Management*. Addison-Wesley, USA, 1996.

[2] Michael Wooldridge and Nicholas R. Jennings. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

[3] Java management programmer's guide. http://java.sun.com/products/JavaManagement/document.html, February 1998.

[4] CIM specification v2.0. http://www.dmtf.org/cim/index.html, March 1998.

[5] Dominique Sidou. Precise semantics for a behavior model in the context of object based distributed systems. Technical Report TUM-I9725, technical university of Munich, Haim, 1997. Workshop on precise semantics for object-oriented modeling techniques.

[6] Rolf Eberhart, Sandro Mazziotta, and Dominique Sidou. Workshop on precise semantics for object-oriented modeling techniques. In *Integrated Network Management V : integrated management in a virtual world*, San Diego, California, USA, May 1997. IFIP, Chapman & Hall.

[7] Jörg P. Muller. *The Design of Intelligent Agents - A Layered Approach*. LNAI State-of-the-Art Survey. Springer, Berlin, Germany, 1996.

[8] M. E. Bratman. *Intentions, Plans, and Practical Reason*. Harvard University Press, 1987.

[9] Anand S. Rao and Michael P. Georgeff. Modelling agents within a bdi architecture. In R. Fikes and E. Sandewall, editors, *Second International Conference on Principles of Knowledge Representation and Reasoning*, pages 473–484, April, 1991.

[10] Anand S. Rao and Michael P. Georgeff. Bdi agents: from theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 321–319, S. Francisco, CA, June 1995.

[11] Raùl Filipe Teixeira de Oliveira. *Gestion des Réseaux avec Connaissance des Besoins : Utilisation des Agents Logiciels*. PhD thesis, École National Supérieure des Télécommunications, 1998.

[12] Yoav Shoham. Agent-oriented programming. *Artificial Intelligence*, (60):51–92, 1993.

[13] Agent builder: An integrated tollkit for constructing intelligent software agents. http://www.agentbuilder.com, September 1998.

[14] Timothy J., Norman, and Derek Long. Goal creation in motivated agents. In Michael Wooldridge and N. R. Jennings, editors, *Proceedings of the 1994 Workshop on Agent Theories, Architectures, and Languages*. Springer, 1995.

[15] Munindar P. Singh, Anand S. Rao, and Michael P. Georgeff. *??*, chapter Formal Methods in DAI: Logic-Based Representation and Reasoning. ?, 1998.

[16] Nicholas R. Jennings and Michael J. Wooldridge. *Agent Technology: Foundations, Applications and Markets*, chapter Application of Intelligent Agents. Springer Computer Science, February 1998.