We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

16

Principles of Agent-Oriented Programming

André Filipe de Moraes Batista, Maria das Graças Bruno Marietto, Wagner Tanaka Botelho, Guiou Kobayashi, Brunno dos Passos Alves, Sidney de Castro and Terry Lima Ruas
Centre of Mathematics, Computation and Cognition, Federal University of ABC (UFABC) Brazil

1. Introduction

In the early of 1970s the Artificial Intelligence (AI) was defined as a system based on the Von Neumann model (with a single control center) and the concepts of traditional Psychology. In the late of 1970s the idea of individual behavior was tested from several works that required a distributed control. For example, works related to blackboards (Fennel & Lesser, 1977) and actors (Hewitt, 1977) allowed the modeling of classical problems considering concepts such as cooperation, communication and distribution. Therefore, the researchers started to investigate the interaction between systems, trying to solve distributed problems in a more social perspective.

In order to find solutions for distributed systems, the Distributed Artificial Intelligence (DAI) started in the early of 1980s to be investigated. It combines the theoretical and practical concepts of AI and Distributed Systems (DS). The solution is also based on social behaviors where the cooperative behavior is utilized to solve a problem. The DAI is different of DS because (i) it is not based on the client-server model, and (ii) the DAI area does not address issues related to distributed processing, aiming to increase the efficiency of computation itself (transmission rate, bandwidth, etc). However it aims to develop technical cooperation between entities involved in a system. Also, the DAI differs from AI because it brings a new and broader perspectives on knowledge representation, planning, problem solving, coordination, communication, negotiation, etc.

The Multi-Agent Systems (MAS) is one of the research areas of DAI, and uses autonomous agents with their own actions and behaviors. The agents in a MAS are designed to act as experts in a particular area. The main characteristic is to control their own behaviors and, if necessary, to act without any intervention of humans or other systems. The focus of the designer is to develop agents working in an autonomous or social way, as well as systems of communication and cooperation/collaboration, so that the solution arises from the interactions. This bottom-up approach usually leads to an open architecture, where agents can be inserted, deleted, and reused. According to Sawyer (2003), the Internet is an example of MAS because it is constituted by thousands of independent computers, each on running autonomous software programs that are capable of communication with a program running on any other node in the network.

The term agent is used frequently in AI, but also outside its field, for example in connection with databases and manufacturing automation. When people in AI use the term, they are

referring to an entity that functions continuously and autonomously in an environment where other processes take place and other agents exist. The sense of autonomy is not precise, but the term is taken to mean that the agent activities do not requires constant human guidance or intervention (Shoham, 1993). There are a number of good reasons for supposing that agent technology will enhance the ability of software engineers to construct complex and distributed applications. It is a powerful and natural metaphor to conceptualize, design and implement many systems.

This chapter makes an overview of theoretical and technical concepts of Agent-Oriented Programming (AOP). Historically, the AOP appears after the Object-Oriented Programming (OOP). However the differences between them are not clear in the research and development community. Section 2 discusses the differences between objects and agents and also the evolution of the programming language paradigms. Section 3 presents the micro and the macro levels of a society of agents. The pitfalls of AOP are explained in Section 4. Two multi-agent systems platforms are presented in Section 5. In Section 6 two multi-agent applications are presented: a cognitive modeling of stock exchange and a military application of real-time tactical information management. Section 7 ends up the conclusions.

2. Programming with objects or agents: What are the main differences?

Procedural programs are typically intended to be executed discretely in a batch mode with a specific start and end (Huhns, 2004). However, the modular programming approach employs smaller units of code that could be reused under a variety of situations. The structured loops and subroutines are designed to have a high degree of local integrity (Odell, 1999). The concept of objects and agents are the key to understand the OOP and AOP, respectively.

2.1 Agents versus objects

In (Silva et al., 2003) an agent is defined as an extension of an object with additional features, because it extends the definition of state and behavior associated with objects. The mental states consist of its states and behaviors. The beliefs and goals, plans, actions are equivalent to the object's state and agent's behaviors, respectively. Moreover, the behavior of an agent extends the behavior of objects because the agents have freedom to control and change their behaviors. They also not require external stimuli to carry out their jobs. These make the agents active elements and objects passive ones.

Agents use some degree of unpredictable behavior. For example, the ants appear to be taking a random walk when they are trying to find food. Their behavior starts to become predictable when the pheromones or food are detected. Therefore, an agent can range from being totally predictable to completely unpredictable. On the other hand, the objects do not have to be completely predictable (Odell, 2002).

The agent has the ability to communicate with the environment and other entities. In MAS the agents are autonomous and has the characteristics to interact with the environments and other agents. However, the object messages have the most basic form of interaction. Also, it request via message only one operation formatted in a very exacting way. The oriented-object message broker has the job of matching each message to exactly one method invocation for exactly one object.

In the communication between agents in the MAS is allowed to use the method invocation of OOP. However, the demand of messages are greater than those used by objects technology. An agent message could consist of a character string whose form can vary, yet obeys a formal syntax, while the conventional object-oriented method must contain parameters whose

number and sequence are fixed. Agents may engage in multiple transactions concurrently, through the use of multiple threads or similar mechanisms. Conventional OOP have difficulty to support such requirements (Odell, 2002). But it is possible for the agents to employ objects for situations that require a little autonomous or interactive ability. In the MAS environment, an Agent Communication Language (ACL) is necessary to send a message to any agent. KQML (Group et al., 1992) and FIPA ACL (Buckle & Hadingham, 2000) are examples of the ACLs.

2.2 Object-Oriented Programming

Rentsch (Rentsch, 1982) predicted in 1982 that the OOP would be in 1980s what the structured programming was in the 1970s. The earliest precursor of MAS was OOP. The subsequent OOP evolution provide methods and techniques to identify the objects and their attributes needed to implement the software, to describe the associations between the identified objects, to define the behavior of the objects by describing the function implementations of each object, to refine objects and organize classes by using inheritance to share common structure are the challenges of object-oriented analysis and design (Wahono, 2001). In OOP, an object is a single computational process maintaining its own data structures and procedures (Sawyer, 2003). Also, it maintains the segments of code (methods) and gain local control over the variables manipulated by its methods. In traditional OOP, the objects are passive because their methods are invoked only when some external entity sends them a message. The basic element used in the OOP is the class. A class definition specifies the class variables of an object and the methods that the object accepts. One class inherits from another class such that the new class is an extension of the existing class, instances of two classes collaborates with each other by exchanging messages (Lind, 2000).

In Wahono (2001), the object is defined as the principal building blocks of OOP. Each object is a programming unit consisting of attributes (instance variables) and behaviors (instance methods). An object is a software bundle of variables and related methods. It is easy to see many examples of real-world objects. Also, it is possible to represent the real-world objects using software objects. For example, bicycles have attributes (gear, pedal cadence, two wheels) and behaviors (braking, accelerating, slowing down). A software object that modeled our real-world bicycle would have variables that indicate the bicycle's current attribute: its speed is 10 mph, its pedal cadence is 90 rpm, and its current gear is the 5^{th} gear. These variables and methods are formally known as instance variables and instance methods to distinguish them from class variables and class methods.

2.3 Agent-Oriented Programming

The MAS is considered as an object-oriented system that is associated to an intelligent meta-system. By this way, an agent is viewed as an object that has a layer of intelligence, comprising a number of capabilities such as uniform communication protocol, perception, reaction and deliberation, all of them not inherent to objects. However, the AOP has code, states and agent invocations. The agents also have individual rules and goals to make them appear like active objects with initiative. In AOP the class is replaced by role, state variable with belief/knowledge and method with message. The role definitions describe the agent capability and the information needed to desired results.

In order to the agents act with intelligence in their environment, the idea is to develop the complex entities and provide the agents with the knowledge and beliefs to be able to achieve their desires.

Framework	OOP	AOP
Basic unit	object	agent
Parameters defining state of basic unit	unconstrained	beliefs, commitments, capabilities, choices, ...
Process of computation	message passing and response methods	message passing and response methods
Types of message	unconstrained	inform, request, offer, promise, decline, ...
Constraints on methods	none	honesty, consistency, ...

Table 1. Relation between OOP and AOP (Shoham, 1993).

2.4 Differences between Object-Oriented Programming and Agent-Oriented Programming

Table 1 summarizes the major features of the relation between OOP and AOP. In short, AOP is seen as an extension of OOP. On the other hand, OOP can be viewed as a successor of structured programming. Wagner (2003) defines two main characteristics about the AOP. First, while the state of an object in OOP has no generic structure, the state of an agent in AOP consists of mental components such as beliefs and commitments. Second, while messages in OOP are code in an application-specific ad-hoc manner, a message in AOP is coded as a speech act according to the standard Agent Communication Language that is application-independent.

The autonomy and interaction are the key areas to differentiate the AOP from OOP. The following list describes some underlying concepts that agent-based systems employ (Odell, 2002):

- Decentralization: the objects are centrally organized, because the objects methods are invoked under the control of other components in the system. On the other hand, the agent has a centralized and decentralized processing;
- Multiple and dynamic classification: in OOP, objects are created by a class and, once created, may never change their class or become instances of multiple classes (except by inheritance). However, the agents provide a more flexible approach;
- Small in impact: the objects and agents can be described as small grained or large grained. Also, in comparison with the whole system the agent or object can be small. In an agent-based supply chain, if a supplier or a buyer is lost, the collective dynamics can still dominate. If an object is lost in a system, an exception is raised;
- Emergence: the ant colonies have emergent qualities where groups of agents behave as a single entity. Each consists of individual agents acting according to their own rules and even cooperating to some extent. In MAS, simple rules produce emergence. Since traditional objects do not interact without a higher-level thread of control, emergence does not usually occur. As more agents become decentralized, their interaction is subject to emergence.

3. Agents and agency in the Agent-Oriented Programming

3.1 Micro level

The micro level refers to the agent itself. The agent definition, reactive and cognitive agents and their architectures are considered in this level.

3.1.1 The agent definition

The term agent is increasingly used in diverse ways in DAI. However, it has become meaningless without reference to a particular notion of agenthood. Some notions are primarily intuitive, others quite formal. It is also called a software agent or an intelligent agent. The words intelligent and agent describe some of its characteristic features. Intelligent is used because the software has certain types of behavior. The intelligent behavior is the selection of actions based on knowledge and the agent tells something about the purpose of the software. When the researchers in DAI think about the term, they refer to an entity that functions continuously and autonomously in an environment in which other processes take place and other agents exist. The sense of autonomy is not precise, but the term is taken to mean that the agents' activities do not require constant human guidance or intervention (Shoham, 1993).

In the literature we find many discussions about what constitutes an agent. In Shoham (1993), an agent is an entity whose state is viewed as consisting of mental components such as beliefs, capabilities, choices, and commitments. Also, there are several approaches utilized to define agents. (Wagner, 2003) uses the software engineering and the mentalist approaches. In the first approach, it emphasizes the significance of application-independent high-level agent-to-agent communication as a basis for general software interoperability. For example, an entity is a software agent if and only if it communicates correctly in an agent communication language. In the second, the approach is based on the knowledge representation paradigm of AI, points out that the state of an agent consists of mental components such as beliefs, perceptions, memory, commitments, expectations, goals and intentions. Its behavior is the result of the concurrent operation of its perception system, its knowledge system and its action system.

According to Wooldridge & Jennings (1995), it is difficult to define an universal accepted definition of the term agent. But the autonomy is the central idea of the concept of an agent. The authors also explained that the difficulty to find the definition is because of the term is widely used by many researchers working in closely related areas. Therefore, they define two general usages of the term: the weak and stronger notions of agency. The weak notion considers a set of properties that a software and hardware represent to be considered an agent. The following properties are defined:

- Autonomy: the agents are able to decide their actions without the direct intervention of humans and others;
- Social Ability: the agents communicate using some kind of agent-communication language with other agents (humans or computational) in order to solve a problem;
- Reactivity: the agents perceive their environments (which may be the physical world, a user via a graphical user interface, other agents, etc) and respond to changes that occurs in them;
- Pro-activeness: the agents has initiative, they do not act only in response to their environment.

In the strong notion the weak notions are preserved, and in addition other properties are considered. These properties are more applied to human's characteristics, such as knowledge, belief, intention, obligation, emotion, anthropomorphism and etc.

Since the beginning of the DAI area, at least a consensus can be perceived in the scientific community: the division of the agents in reactive and cognitive ones. This binary view allows to focus the analysis on the key points of each of these classes. In the following sections these points are highlighted.

3.1.2 Reactive agents

Reactive agents are usually modeled by following the metaphor of biological and ethological organizations such as: hives, anthill, populations of insects, bacteria, antibodies, etc. Such systems provide evidence of emergent intelligence. Following this metaphor, reactive agents tend to be structurally simpler since they do not have an explicit representation of its environment, they are also not capable to perform sophisticated logic reasoning. Their behaviors are based only on the stimulus-response.

In a society of reactive agents the communication takes place indirectly, through the external environment. Also, their decisions are concerned on the current situation since no history actions are stored. Generally, these agents do not plan their future actions and also they do not communicate with other agents. Normally, they know the other agents actions by the changing of the environment. Thus, their intelligent behaviors are obtained through the interaction with the environment and other agents.

In Gottifredi et al. (2010), the reactive architecture is proposed for the implementation of a robot soccer team. In the reactive layer the basic actions are implemented. It includes the basic hardware and software supports that are provided by the league. This involves physical support, such as infrared transmitters, video camera, communication network, and common software.

3.1.3 Cognitive agents

Cognitive agents are inspired by human social organization, groups and hierarchies. They have explicit models of the external world and memory structures that allow the agents to have the history of past actions used to solve the current problems (Bittencourt, 2006). Also, they communicate with each other directly using their perceptual systems (to sense the environment) and the communication system (exchange messages).

The main characteristics about the cognitive agents are: (a) the societies have a few agents; (b) the communications are made directly; (c) they have explicit models of the external world, structures of memory which keep the history of the past actions and have the ability to make predictions about the future; (d) they are able to decide about their intentions and knowledge, create and execute their actions plans; (e) They have a cooperation and coordination system.

The studies of cognitive science and DAI techniques allow the agents to add the ability of thinking and learning. In fact, cognitive agents have a certain computational complexity and are characterized by an intelligent behavior in the agency or separately. There are several theories utilized to model the cognitive agents such as decisions theory (Parmigiani et al., 1997) and intentional systems (Dennett, 1987). It is possible to achieve the solution of many problems using knowledge and thinking based on the information that represents aspects of their world.

3.1.3.1 Intentional Systems

It is vital the development and application of theoretical models to structure and help reasoning about the individual and social agents´ behaviors. Among such theoretical models we cite the theory of intentional systems proposed in (Dennett, 1987). The philosopher Daniel Dennett (Dennett, 1987) uses the term intentional systems to describe systems that can be described and/or predicted through mental attributes such as beliefs, preferences, desires, intentions, free will, goals, etc. These attributes are called by (Wooldridge & Jennings, 1995) as intentional notions.

Therefore, what characterizes the intentional system is the possibility to be interpreted as a system with its intentional notions. The desires specify the preferences regarding to the future states of the world. They can be inconsistent and not reachable. The goals are the desire that an agent consider achievable in a certain moment. The intention has an associated commitment that directs and controls the future activities of the agent, so that it achieves its goals. The beliefs are the expression of the states of the world seen by the agent. Moreover, they also make the vision of how the world will change if the agent performs some actions.
The cognitive agents can be modeled as intentional systems with mental attributes that influence their actions. The differences in the model depend on the specification of the mental states used to describe the behaviors of each agent.

3.1.3.2 BDI Architecture

The architecture specifies the structure and the behavior of an agent. The BDI (Belief, Desire and Intention) model is probably the most known cognitive agent architecture. It was proposed by (Bratman, 1987) as a theory of human practical reasoning. Thus, the BDI are the mentalists attributes to define the agent state. The agents beliefs and goals correspond to the information that the agent has about the world and the intuitively correspond to the tasks allocated, respectively. (Rao & Georgeff, 1995) have adapted the model of Bratman changing to the formal theory and in a software agent model based on beliefs, goals and plans. The model is a BDI interpreter used as an inspiration for the BDIs systems utilized until now (de Nunes, 2007). In order to design an agent based on the BDI model, the beliefs and desires need to be specified. The agent is allowed to choice the intentions attributes, based on a self-analysis of the states initially available.
The BDIs applications emerge as a solution to various problems. For instance, Unmanned Aerial Vehicles (UVA) (Reichel et al., 2008), Real-time Scheduling (Paulussen et al., 2004) and others. In Section 6.1.2 we propose an architecture based on the general BDI and dMars models presented in (Wooldridge, 1999) and (d'Inverno et al., 1998b), respectively. The BDI dMars architecture contains four keys data structures: beliefs, goals, intentions and a plan library. The agents beliefs and goals correspond to the information that the agent has about the world and the intuitively correspond to the tasks allocated to it, respectively. Agents must choose some subset of available desires (i.e. intentions) and commit resources to achieve them. These chosen desires are intentions. Each agent has a plan library, which is a set of plans, or recipes, specifying courses of action that may be undertaken by an agent in order to achieve its intentions (d'Inverno et al., 1998b). Additionally to the dMars architecture, the Wooldridge generic BDI architecture has useful attributes (Wooldridge, 1999). For example, the filter, generators and revisions functions that makes the deliberation information.

3.2 Macro level

At the macro level, we investigate the agent communication languages, protocols communication between agents, coordination mechanism and negotiation.

3.2.1 Agent communication languages

The communication between members in any society is very important. It is not different in the society of agents that communicate between them to achieve their goals. The communication is a natural way to have interaction, cooperation and negotiation between agents in MASs. It is important for the agent to have the ability to perceive (receive messages)

and to act (send messages) in an effectively and efficiently ways in their environment. For that purpose, they need a shared language and communication protocol.
Agents can process and refer to the same object differently. Therefore, the language structure is needed to allow the integration of these representations. The Speech Acts is a linguistic approach utilized to model the language adaptivity (Marietto, 2000). The philosophers John Austin (Austin, 1962) and John Searle (Searle, 1969) developed the Speech Acts theory, that views human natural language as actions such as requests, suggestions, commitments, and replies (Huhns & Stephens, 1999). These philosophers contribute to the studies related to the development of the agent languages and the communication protocols. John Austin noticed that certain expressions were like "physical actions" that seemed to modify the state of the world. The theory considers three aspects (Huhns & Stephens, 1999):

- Locution: the physical utterance by the speaker. For example: "Can you make a coffee?"
- Illocution: intended by the utterance. For example: "He asked me to make coffee."
- Perlocution: the action that results from the locution. For example: "He made me make coffee."

[Albuquerque e Tedesco, 2004] pointed out that a communication language must have a predictable perlocutionary act with a locutionary act to make it possible to know what is the issuance perlocutionary act. In other words, the language must provide a mechanism where an agent knows the possible responses that another agent will answer in relation to the sender's message, predicting the possible reactions of the receiver.
The illocutionary concept is utilized to define the type of message in the Speech Scts theory. The intentions of the sender communication act are clearly defined, and the receiver has no doubt about the type of the sent message (Huhns & Stephens, 1999). The speech acts are referred as "performatives" in the context of agent communication. It is utilized to identify the illocutionary force of the special class of utterance. The verbs promise, report, convince, insist, tell, request, and demand are examples of performatives.
Based on the Speech Acts theory, it is possible to represent the interaction between agents as an exchange of knowledge. In the communication between agents, the messages have an intention (illocutionary act) and content (locutionary act). For instance, the message TELL(A, B, P) means that the agent A is telling to the agent B that he believes in P. The intention is to inform, and its content is P (Coutinho et al., 2009).

3.2.2 Communication protocols between agents

Protocols are defined as a set of rules to support the network communication. The protocols control the format, the content and the meaning of the sent and received messages. However, they are not restricted only in the communication tasks, but also to assist the negotiation process. They also guarantee the interactions between other agents (Marietto, 2000). In other words, when the agents have conflicting goals or are simple self-interested, the objective of the protocols is to maximize the payoffs of the agents (Huhns & Stephens, 1999). Three protocols are described in (Marietto, 2000):

- Announcement Protocols: The agent informs other agents in the society the services that are able to offer. The agent who is offering the service sends an advertisement (performative) with the :CONTENT (parameter) specifying the actions that is possible to perform. It receives a positive response message if the :CONTENT is validated. Otherwise, it receives a message of rejection.

- Ask-about Protocols: This protocol defines the sequence of two messages, allowing an agent to ask another agent. For example, a general question about the knowledge base of another agent, a request to identify an agent or a request to evaluate an affirmation.
- Task/Action Agreement Protocols: This protocol defines a sequence of messages, allowing an agent to engage a task of another agent.

3.2.3 Coordination

In sections 3.2.2 and 3.2.1 we analyzed how the agent shared the same environment and communicate with each other. However, in order to work together in a harmonious manner, it is necessary more than the ability to communicate with each other. In DAI systems the agents cooperate to achieve their local goals and the goals of the society they belong . Also, they perform a range of useful activities, have their own aims, goals and communicate with other agents (Jennings, 1995). (Reis, 2003) defines coordination as "the act of working together harmoniously to achieve an agreement or common purpose".

The coordination should be able to deal with agent systems with opposite goals or commons goals acting collaboratively or cooperatively. A collaboration is established when an agent has autonomy to perform a task. In order to speed up the execution time, the agent is allowed to accept or ask other agents for help. Therefore, the cooperation occurs when an agent is not able to realize a certain task (Marietto, 2000). In Jennings (1995) and Nwana et al. (1997) a few reasons related to the coordination between multiple agents are described:

- Prevent the chaos among the agents: the coordination is necessary or desirable when the MASs are decentralized. Thus, the chaos is established quite easily;
- Distribute information, resources and expertise: through the coordinated distribution of information, resources and expertise, complex problems are solved more efficiently;
- Increase the efficiency of the system: through the exchange of information or shared of information, the coordination increases the efficiency of the system. Even the agents are able to perform the tasks, they can exchange their tasks if they are able to perform more efficiently;
- Manage interdependencies between the actions of the agents: interdependence occurs when the actions needed to achieve the goals of the individual agents are related;
- No individual agent has the ability, resources or enough information to solve the proposed problem in a separate way: most of the problems require different knowledge to be solved, which can only be achieved by different agents. Therefore, different knowledge of the agents is combined to find the desired result.

To find the best solution in a distributed intelligent system the coordination process manages the behavior of the agents because they have local incomplete information about the problem. The management aims to avoid conflicts, redundant efforts, deadlocks, etc. The higher incidence of such situations, less coordinated is the system.

3.2.3.1 Coordination Mechanism

Jennings (1995) and Bond & Gasser (1998) describe three most common coordination mechanisms.

- Organizational Structures: it is the formalization of the types of interaction between individuals. It offers a constraint environment and expectations about the behavior of the agents (i.e. through a set of rules) that guides the decision-making and the actions of the agents. Therefore, the relations specified by the organizational structures give long-term information about the agents and the society as a whole;
- Meta-levels Information Exchange: they are directly concerned to the concept of abstraction. According to Bond & Gasser (1998), the agents communicate with other agents semantically related to a common problem. Thus, it is possible to establish an abstract structure that formalizes these semantic relationships. The exchange information in a meta-level allows an agent to reason about the past, present and future activities of other agent;
- Multi-agent Planning: the shared tasks between the agents help the coordination mechanisms if the agent's behaviors are related to the overall goals of the system. It is possible to know how the actions performed by the agent affects the behavior of the system if we use the planning technique in the coordination process. In this case, the planning should take into the consideration issues such as plans control, inconsistencies, conflicts, etc.

3.2.4 Negotiation between agents

The negotiation represents an important technique related to cooperative activities and human societies, allowing people to solve conflicts that may interfere in the cooperative behavior. In the DAI, the negotiation between agents has the same purpose.
In general, the goals of a negotiation process are (E.H. Durfee, 1989):

1. Modify the local plans of an agent, the interaction does not occur successfully;
2. Identify situations where potential interactions are possible;
3. Reduce inconsistencies and uncertainties about different points of views or plans in common, through an organized exchange of information.

The negotiation mechanism provides efficiency to the system allowing the agents to redistribute the tasks between them in order to minimize an individual agent's effort. Thus, the negotiation will provide stability to the agents if their tasks are distributed correctly in the group (Faraco, 2001).

4. Pitfalls of Agent-Oriented Programming

In this section we identify some of the main pitfalls that any development project based on agent system can expect to find. Agent's technologies are not a universal solution even if they have been used in a wide range of applications. In many applications, using OOP is more appropriate. If a problem is solved with equal quality solutions using agent and non-agent approach, the non-agent approach is preferred since it is better understood by the software engineers involved in the system development. There is a misunderstanding that agents are the right solution to every problem. Therefore, they are often developed for inappropriate problems.
The interactions between agents yield unpredictable collective behavior in the general case. The patterns and the effects of their interactions are uncertain because of the agents' autonomy. The agents decide which of their goals require interaction in a given context. In

order to realize these goals, the agents decide which knowledge they will interact, and also when these interactions will occur (Jennings & Wooldridge, 2000).
In (Jennings & Wooldridge, 2000; Wooldridge & Jennings, 1998), some pitfalls are identified and happened in some of the agent development projects:

- You get religious or dogmatic about agents: even the agents have been used in a wide range of applications, their technology are not the only solution. For example, the OOP is more appropriate in many applications. However, there is a danger of believing that agents are the right solution to every problem. Therefore, sometimes they are developed for inappropriate problems;
- You don't know why you want agents: sometimes the concept of agent technology is not clear. But it is utilized to solve some problems only if someone read optimistic information about the potential for agent technology;
- You don't know what your agents are good for: it is important to understand how and where the agent technology is usefully applied.

5. Multi-Agent Systems platforms

Due to the fact that many of the MAS characteristics are independent of the application, frameworks started to be utilized to facilitate the development of such systems. These frameworks provide the basic functionality of MAS, which allows the developers to concentrate in the development of the agents. The main goal of the Foundation for Intelligent Physical Agents (FIPA) is to develop the standard implementation of the open, heterogeneous and interoperable agents. The foundation was created in 1996s and is also responsible to define some standard agents that many developers use to ensure the interoperability between MAS developed with generic frameworks.
Based on the vision of interoperability between systems with different manufacturers and operators, FIPA released as reference the FIPA standard pattern. This pattern has a primary focus on the external behavior of system components, leaving open the implementation details and the architectures of the agents. In addition to this feature, it sets the reference model for an agent's platform, as well as the set of services to be offered by the platform. Among these services there are: Directory Facilitator (DF), Agent Management System (AMS), the Message Transport Service (MTS), and the Agent Communication Language (FIPA-ACL). FIPA does not formally implement any agent architecture because its open standards allow various ways to implement it, simply by following the recommendations and abstract mechanisms defined within. Among the generic frameworks that use the FIPA pattern it is possible to cite: JADE (Bellifemine et al., 2007) and FIPA-OS (Poslad et al., 2000). These platforms are described in the subsections 5.1 and 5.2, respectively.

5.1 JADE platform

The Java Agent Development Framework (JADE) is an environment for developing applications according to the FIPA patterns. It is implemented in Java and was developed at the University of Parma, Italy (Bellifemine et al., 2007). Some characteristics of this platform are listed below:

- Distributed platform of agents - JADE can be divided into multiple hosts or machines. The agents are implemented in Java threads and inserted into repositories of agents called containers, which provide all the support for the implementation;

- Graphical user interface (GUI) - JADE has a GUI interface that assists in managing agents and agent containers;
- Running multiple, parallel and concurrent activities of agents - JADE provides these features through their pre-defined models of agent's behavior. The structure of the agents behaviors using the JADE platform takes place via a scheduler that automatically manages the scheduling of these behaviors.

5.1.1 Platform's architecture

The JADE architecture is based on the coexistence of several Java Virtual Machines (JVMs) that can be distributed over multiple computers, independently of the operating system.
Figure 1 shows the distribution vision of the JADE platform in many hosts. Each host run the JADE agents that forms a container. These containers are registered in the main container of the platform. In each host has a JVM, indicating platform independence, and in each JVM has a container of agents that provides a complete running environment for these agents, in addition to allowing multiple agents to run concurrently on the same processor/host. The execution of the JADE platform occurs at the main container of a platform. The other hosts who own the remaining containers should only have the files needed to run the platform.

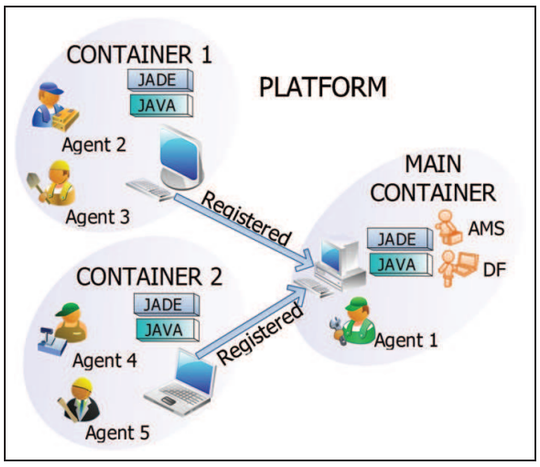


Fig. 1. Functional Architecture of the Jade Platform.

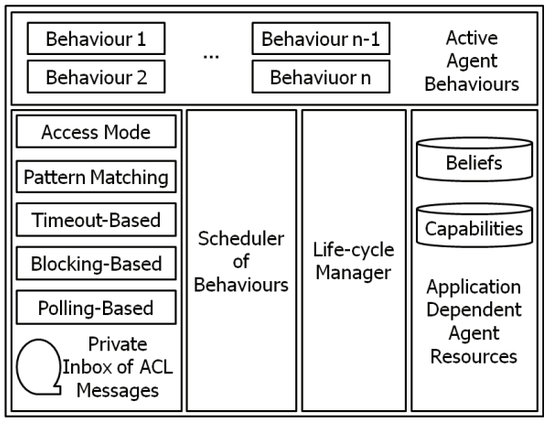


Fig. 2. Internal Architecture of an Agent in the Jade Platform (Bellifemine et al., 2007).

Figure 1 shows the main container where the AMS, the DF and the Remote Method Invocation (RMI) registry are located. The RMI is a name server used by Java to record and retrieve references of objects by name. With the RMI registry the JADE platform keeps references of other containers that connect to it.

5.1.2 Agents in the platform

In the JADE environment an agent is a process that is autonomous with an identity and requires communication with other agents to execute their functions. The JADE platform is neutral as regards of internal architecture of its agent. A JADE agent runs as a thread that employs multiple tasks or behaviors and simultaneous conversations.
Figure 2 shows the internal architecture of the JADE agent. At the top there are the active behaviors of the agent that represent actions that the agent can perform. The computational model of an agent in JADE is multitasking, where tasks (behaviors) are

performed concurrently. Each functionality provided by an agent must be implemented as one or more behaviors.
At the bottom of Figure 2 is possible to see a private messaging box of ACL. Every agent in JADE has this box, and can decide when to read incoming messages and which messages to read. In the center, there are the behavior scheduler and life cycle manager. The scheduler is responsible for scheduling the execution order of the behaviors. The life cycle manager is the controller of the current state of the agent. The agent is autonomous, it uses the life cycle manager to determine their current status (active, suspended, etc.). On the right side of Figure 2 there are the application-dependent capabilities of agents, where will be stored the beliefs and capacities that the agent acquires during the execution of the application.

5.2 FIPA-OS platform

The FIPA-OS platform was originally developed by Nortel Networks (Poslad et al., 2000) and currently is being developed by Emorphia Corporation. It is implemented in JAVA and its architecture has three types of components: mandatory, optional and switchable. The mandatory components are required for the execution of agents. The developers decide to use or not the optimal components. The switchable components have more than one implementation, which allows to choice the best implementation that adapt the necessity of the system. These components are illustrated in Figure 3 and will be discussed in the following sections.

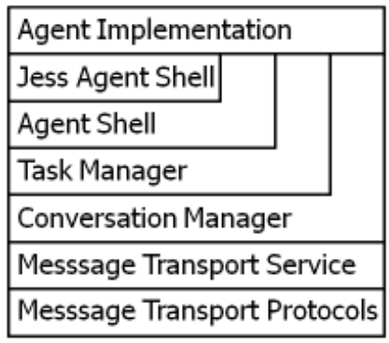


Fig. 3. Components of the FIPA-OS Platform (Poslad et al., 2000).

5.2.1 JESS Agent Shell

The Java Expert System Shell (JESS) is a tool for developing expert systems. The expert system is a set of rules drawn from a collection of facts, which are performed when a set of preconditions are satisfied. The structure of the JESS Agent provides an interface to the JESS, allowing agents to have a knowledge base and deliberative capacity (Buckle & Hadingham, 2000).

5.2.2 Agent Shell

The FIPA-OS platform provides a model called Agent Shell for the construction of agents. Agent Shell is a mandatory component that facilitates the implementation of FIPA-OS agents through an extended set of base classes. Through these base classes several features are provided such as transport, retrieval and buffering messages (Buckle & Hadingham, 2000). However, developers of agents does not necessarily need to use the Agent Shells, as FIPA-OS enables the independent development of agents.

5.2.3 Task Manager

Task Manager is a mandatory component that provides a way to split the functionality of an agent in small units of work, known as tasks. The purpose of this feature is to make the task as a piece of code that perform some functions and optionally returns some result (Buckle & Hadingham, 2000). Moreover, it must have the ability to send and receive messages, and should have a little or no dependence on the agent that is running it.

5.2.4 Conversation Manager

All messages exchanged between agents are part of a conversation. The Conversation Manager is a mandatory component that allows it to be kept a navigable history of conversations between agents (Buckle & Hadingham, 2000), as well as mechanisms for grouping messages of the same conversation. It is also the responsibility of this manager to ensure that both sides of a conversation use the same protocol.

5.2.5 Message Transport Service

Message Transport Service (MTS) is a mandatory component that provides capacity for sending and receiving messages between agents. The MTS of FIPA-OS platform is organized in a model of layer of services. Messages sent by an agent coming to MTS and are modified as they are moved from a higher to a lower layer, until they reach the Message Transport Protocol (see Subsection 5.2.6).

The protocol to be used to send the message depends on whether the message is for local or for a remote platform. If local, the message is sent to the destination agent passing through the layers of services in reverse order. If the message is intended for a remote platform, the service layers through which the message should go will depend on how the remote MTS is implemented.

5.2.6 Message Transport Protocol

Message Transport Protocol (MTP) is a switchable component that provides implementations of the protocols used by the MTS for sending and receiving messages (Buckle & Hadingham, 2000). The FIPA-OS platform divides these protocols into two types: internal and external. Internal protocols are responsible for carrying messages between agents in the same platform. For this task the system uses the FIPA-RMI. External protocols are responsible for carrying messages between agents in different platforms. For this task FIPA-OS uses the Internet Inter-ORB Protocol (IIOP). The IIOP is a universal standard for communication between distributed objects, having the advantage of being interoperable with any languages that support its specifications.

6. Multi-agent system applications

In this section we demonstrate in detail two multi-agent applications based on the JADE platform that were modeled and implemented by our Social and Artificial Intelligence research team at the Centre of Mathematics, Computation and Cognition (CMCC) of the Federal University of ABC (UFABC), Brazil. The two applications are: a cognitive modeling of stock exchange and a military application of real-time tactical information management.

6.1 Modeling cognitive Multi-Agent System for stock exchange

Stock exchanges play a major role in the global financial system. Among the most known stock exchanges we can mention: Dow Jones (New York, USA), Nikkei (Tokyo, Japan),

and BM&FBOVESPA (São Paulo, Brazil). The behavior of the stock market is complex and dynamic, which characterizes this system as unpredictable. Even with such unpredictability the stock market possesses unique characteristics and some traits in common that when viewed in general, it is possible to detect certain stability (Azevedo et al., 2008; Rutterford & Davison, 1993).

Among the observed patterns in stock market, beyond the fluctuation of prices and indexes, it is also noted social phenomena that emerge between its investors. Aiming to better understand the formation of behavior patterns in a stock exchange, this paper presents a social strategy to determine the behavior of the investor based on the theory of imitation ((Wei et al., 2003)). This theory is based on the concept of herding behavior, which says that the behavior of an agent is a function of imitation of the others attitudes. Thus, in this simulation investor behavior is modeled for his/her decision is based on the behavior of other investors in the same stock market environment. With the development of this work it will be possible to study the social behavior, improve techniques in the field of AI, and to analyze an investment strategy in details.

The analysis of the social strategy of imitation in the stock market, the operation of a stock exchange was modeled encompassing elements such as investors, stock brokerage and financial market. As theoretical and technical basis will be used the concepts of the field of DAI, specifically MAS.

6.1.1 Theoretical basis of stock exchange

According to (Fama, 1965) stock exchanges are organized markets which negotiate equity securities denominated shares of publicly traded companies. Trading in the securities occurs when an investor passes his order to buy or sell shares to the market. If there is another transaction of equivalent value in the opposite direction (to sell or buy), then the deal is done. Thus, the share price is set by supply and demand of each share. To make deals at the stock market there are companies that perform the necessary procedures. They are called stock brokers and act as an interface between the investor and the stock exchange for buying and selling shares (Rutterford & Davison, 1993).

The market index is estimated from the shares prices with greater trading volume. This index serves as a mean indicator of market behavior, and is used as a benchmark by the investors to follow its behavior over the time. The first index calculated on the day is called open index and closing index is the last. The minimum and maximum index are, respectively, the lowest and highest value recorded on the day. When the closing index of the day is superior to the previous day's then it is said that the stock market is high. If it is lower, then it is said that the stock market is down. The market is considered stable when these values are equal (Rutterford & Davison, 1993).

6.1.2 Modeling a stock exchange with cognitive Multi-Agent System

The MAS modeled in this section simulates the cognitive environment of a stock exchange, which contains the main elements shown in Figure 4.

It consists of a MARKET agent and several STOCK BROKER MASs. The interaction between these entities occurs when the MARKET agent sends data (such as stock index and companies shares) to the STOCK BROKER MASs. The brokers communicate with each other to know what decision (buy, sell or stay with the shares) the other brokers took. In the process of decision making each STOCK BROKER MAS uses the theory of imitation to define what will be its behavior. With the decision, the purchase or sale of shares is authorized between the

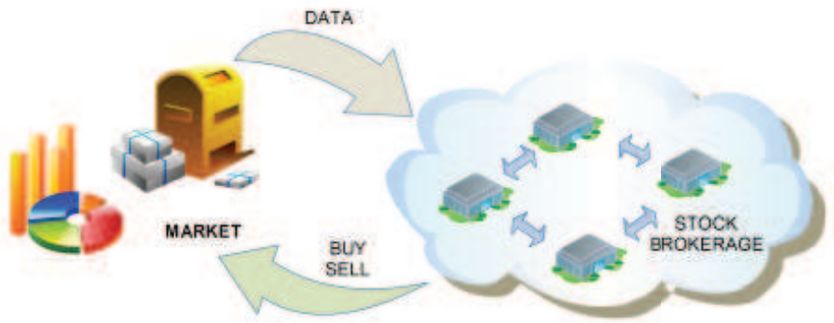


Fig. 4. The Main Elements of Stock Exchange MAS.

brokers. Each STOCK BROKER MAS is composed of five agents: COMMUNICATOR, MANAGER, DECISION MAKER, BUYER and SELLER (see Figure 5). This distribution aims to compose the STOCK BROKER with expert agents, each with a feature. This division facilitates future expansion of the simulation, for example, to adopt a different strategy for making decisions it is necessary just replace the agent responsible for this process.

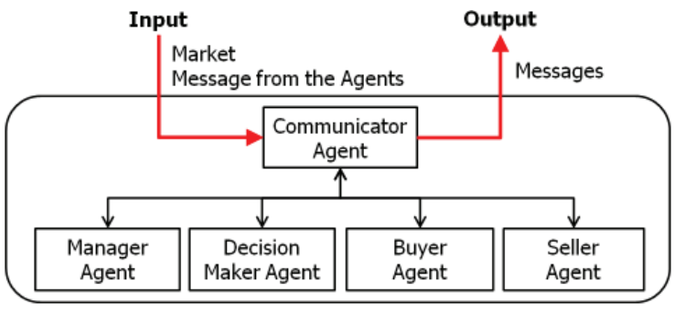


Fig. 5. STOCK BROKER MAS Architecture.

6.1.2.1 Agents of the STOCK BROKER MAS

The modeling of STOCK BROKER´s agents was based on BDI architecture of (d'Inverno et al., 1998a) and on general BDI architecture presented in (Wooldridge, 1999). These architectures are described in Section 3.1.3.2 and they are combined to join the functionality of sub-goals of the dMars architecture and the filters, generators and reviewers of Wooldrigde architecture. As can be seen in Figure 6, the proposed architecture performs the following cycle:

- INPUT MODULE is responsible for the agent perceives the world and receives messages from other agents. The information is sent to the INFORMATION MODULE for analysis by the INFORMATION FILTER;
- The INFORMATION FILTERS classify information according to two criteria: (i) Belief - information that the agent does not have absolute certainty, both its content or source are questionable, (ii) Knowledge: information that the agent believes to be true;
- GOALS MANAGER accesses the INFORMATION MODULE to define what goals are achievable by the agent in a given time. The goals selected by the GOALS FILTER are sent to the INTENTIONS MODULE for further analysis;

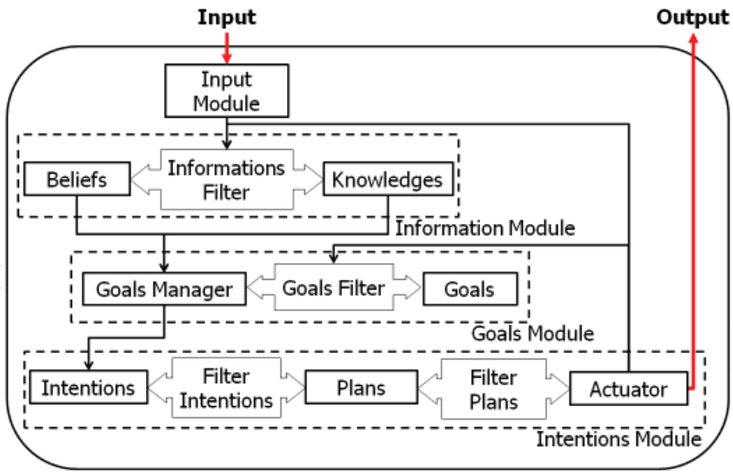


Fig. 6. Proposed BDI Architecture.

- The intentions are arranged in a row considering the priority order to be executed. Thus, the highest priority intention is the one the agent will try to achieve first;
- In the element PLANS there are the steps necessary to the agent perform and accomplish its intentions. These plans are defined in advance by the developer;
- ACTUATOR is responsible for the agent's action on the environment. It works as output to the world as it executes the plans for the intentions are met. During the execution of the plans, the ACTUATOR may: (i) insert new goals (subgoals), (ii) insert, remove and/or prioritize the intentions and/or information.

Now the features of each agents of the STOCK BROKER MAS are presented.

COMMUNICATOR Agent

The COMMUNICATOR agent is responsible for all the communications of STOCK BROKER MAS, acting as an interface with the outside world (other STOCK BROKER MAS and the MARKET agent) and its internal structure (formed by MANAGER, DECISION MAKER, BUYER and SELLER). Besides acting as an e-mail server, the COMMUNICATOR agent also assumes the functionality of general secretary because it knows who is responsible for each sector of the company.
This agent filters incoming messages and define them as part of their goals. Later, adopt a message as an intention and sends it to correct recipients. For that stores, a message queue that can be used for further processing, redirecting them or not when it seems necessary. Part of COMMUNICATOR agent's knowledge are (i) the messages, whether received or to be sent to COMMUNICATOR agents of other STOCK BROKER MAS, (ii) and MARKET agent.

MANAGEMENT Agent

The MANAGER agent stores global data relating to STOCK BROKER MAS. Information such as capital, stock portfolio, the final decision, how influenced the broker is, etc. It is also the responsibility of the MANAGER agent to send these information to other internal agents when required. The exchange of messages between the MANAGER and other agents is mediated by the broker's COMMUNICATOR agent. The MANAGER agent architecture is based on BDI architecture proposed in Figure 6. However, it lacks the GOALS MODULE because its functionality always provide data when requested.

DECISION MAKER Agent

In a stock brokerage firm, there are several types of analysts that define how to invest in the stock market: graphic analyzers, specialists in a niche market or specialists in high and moderate risk of investments. The DECISION MAKER agent implements the functionality analysis of the STOCK BROKER, deciding whether the broker will buy, sell or hold the shares it owns. In this work, the strategy adopted by the DECISION MAKER agent is imitation, so its behavior is based on the decision taken by the other STOCK BROKER MASs. The modeling of this strategy is based on the work of (Wei et al., 2003).
Investment preferences reflect the behavior of the investor's imitation, which is influenced by (i) macro factors (MF) represented by the stock index (ii) and a probability P to mimic the other investors, characterizing the investor's permittivity (degree of imitation).
P is a random number given to each agent when it is instantiated. The MF is a normalized value between 0 and 1 taking into the consideration the higher and lower stock index. The objective of MF is to quantify the economic situation of the market. With the MF above 0.5 agents tend to buy stocks, whereas the opposite trend is to sell. Given the state of the MF and the highest number of one type of behavior of other investors (buy, sell or hold), the agent in question defines its behavior.
Thus, the knowledge base of DECISION MAKER agent are composed by the coefficient of permissiveness of the investor, the stock exchange current index, the current MF and the most common decision taken by other STOCK BROKER MASs. To determine the most common decision taken, the DECISION MAKER agent requests to the COMMUNICATOR agent to send a broadcast message to other STOCK BROKER MASs. Until receiving all the answers from the brokers, the decision is suspended. After accounting the most frequent behavior, the DECISION MAKER agent decides its behavior.
With the goal chosen to be promoted to the intention (to buy, sell or hold shares), the DECISION MAKER agent performs a plan. If the intention is to purchase or sell the BUYER or SELLER agent, respectively, is triggered via the COMMUNICATOR agent to start in executing their tasks. If the decision is to keep the shares it owns, nothing is done.

BUYER Agent

In distribution of areas inside the corporation, there is a clear distinction between buying and selling areas. For STOCK BROKER MAS modeling this distinction was also used. The BUYER agent has the function of executing, purchase transactions of shares with the MARKET agent when the broker makes this decision. The purchase decision is made by the DECISION MAKER agent, which informs the BUYER agent. The MANAGER agent, through the COMMUNICATOR agent, informs the BUYER agent the current value of broker's capital.
The BUYER agent's knowledge consists of the shares it may acquire and their prices. After determining that it will buy shares and own the necessary capital for the purchase, it will make a purchase offer to MARKET agent. The proposed purchase may be accepted or not. If there is confirmation of the transaction, the capital employed will be charged to STOCK BROKER's capital and the shares purchased will be added to its portfolio.

SELLER Agent

As there is an agent to make the purchase, also there is an agent to conduct the sale of shares that the STOCK BROKER MAS has. The SELLER agent has the role of conduct sales transactions with the MARKET agent when the broker makes this decision. The actions executed for sale are similar to the purchase, but the SELLER agent, instead of capital, requests the MANAGER agent the shares portfolio of STOCK BROKER MAS.

The SELLER agent's knowledge consists of the shares and their respective prices. After informed that it must sell the shares, it will make the selling proposal to MARKET Agent. The proposed sale may be accepted or not. If there is confirmation of the transaction the capital from the sale will be credited to the capital of the STOCK BROKER, and the respective shares will be removed from its portfolio.

6.1.3 Case study

The STOCK EXCHANGE MAS was implemented using the JADE platform. The system was run using actual values of the BM&FBOVESPA index of 2004. In this simulation is established a communication between the agents that make up a STOCK BROKER, MARKET agent and other broker's COMMUNICATOR agents. At the start of the simulation the MARKET agent sends to all COMMUNICATOR agents (which are registered on the platform) a message informing the stock index value, characterizing the opening of trading. Thereafter every STOCK BROKER shall perform its flow to make decisions and make deals with the MARKET.

During the execution of the simulation it is possible to see that agents gradually converge to similar behaviors. In early trading, when the index is down, some agents hold the majority shares and start selling the shares it owns. In the middle of the year with the onset of high index, the agents start to buy shares in small numbers, increasing the number of agents with this behavior as the index increases at each interaction. It is perceived that agents with low permittivity are less influenced by others. The agents with high permittivity are more susceptible to the decision of others, thereby mimicking the behavior of majority of the agents assigning low relevance for the stock index.

6.1.4 Future works

Among the possible extensions of this work, there is the possibility of modeling the concept of reputation for each STOCK BROKER MAS. With this concept, it will be possible that each broker would have a perception about the "reliability" of the other brokers. This information can assist the process of decision making regarding to the investment, because the imitation strategy can now be adjusted to take into consideration the status of the reputation of each broker's information.

Another proposed improvement for the system is to adapt the DECISION MAKER agent to make more specific decisions, based on individual performance data of the companies participating in the stock market. For example, in the same round this agent may choose to sell a company's share and buy another if the first company is performing poorly while the second is on the rise. Or in a more complex reasoning, where two companies are on the rise, but one company is more promising than another. So the agent chooses to sell shares of a company to raise capital, with the intention to purchase shares of the company that has a higher profit outlook.

6.2 Multi-Agent Systems for real-time tactical information management

With technological advances implemented in today's battlefields, arises the need for the command of military operations enhances the ability of decision making by responding quickly and efficiently to the events in the field. Dealing with a growing number of variables and the continuous increase in the complexity of actions, there is a need for increased speed and quality of information processing. Within this context the armed forces fight today with a new perspective, using advanced technologies to improve their decision quality and,

consequently, the chance of victory (Cil & Mala, 2010), (Xuan et al., 2001),(Bonabeau, 2002), (Barbuceanu & Fox, 1995), (Wijngaards et al., 2002).
Following this new perspective, the current processes of decision making in combat use pre-processing technologies and information distribution that are based on symbols. These symbols represent important information for the participants of a specific mission. This simplified representation of the environmental elements and the dynamics of action allow a clearer interpretation of the direction of activities and improve the prospect for a global action in real time (Panzarasa et al., 2002).
To aid the process of decision making in areas of conflict, this work proposes a MAS where soldier agents interact with cyber agents. From this interaction will be possible to map and keep under surveillance a particular area. This work will also allow analysis of how this collaboration influences the quality of decision making, and what are the most significant characteristics that emerge from such human-machine interaction.

6.2.1 Multi-Agent System modeling tactical information in combat situations

To aid the process of decision making on the battlefield, this work proposes a MAS composed of cyber agents (radar and helmets) interacting with human soldiers and commanders. Figure 7 shows SOLDIER agents on patrol, whose mission is to map and keep under surveillance a particular area. SOLDIER agent carries RADAR agents, which are left over from the walk aiming to maintain surveillance of these areas after his/her passage. The CONTROL agent receives the information from all agents (SOLDIER, RADAR), processes and distributes to the soldiers a simplified representation of the information of the environment (projected on the visor of his/her helmet), also showing for each soldier the direction to follow.
The environment in which the action takes place consists of several obstacles: lakes, roads, forests, enemies, among others. The decision to fight or flee from obstacles is coordinated by the CONTROL agent who evaluates the possibilities in order to achieve tactical superiority in action.

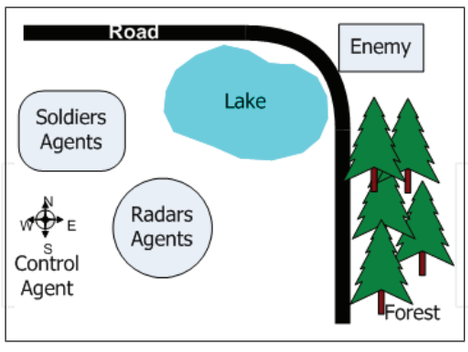


Fig. 7. Overview of a Combat Situation.

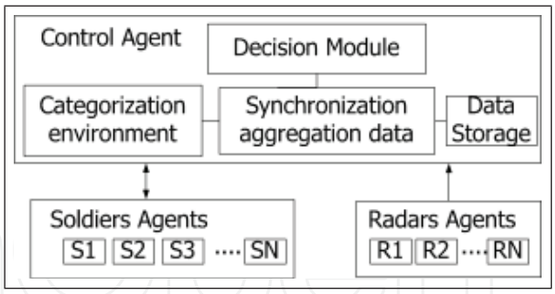


Fig. 8. General Architecture of MAS of Real-Time Tactical Information in Combat Situations.

The MAS proposed in this work to represent a war situation is illustrated in Fig 8. It consists of the following agents: RADAR, SOLDIER and CONTROL (presented in Subsection 6.2.1.1). This MAS is immersed in an environment that is described in Subsection 6.2.2.

6.2.1.1 Agents of MAS

The following are presented the features of the proposed MAS.

RADAR Agent

As the soldiers move on the ground they install radars, under the control and command of the CONTROL agent, aiming to maintain surveillance after leaving a particular area. After installed, the RADAR agent, continuously sends the CONTROL agent images that are in its scanning spectrum (within the region that is at your scope). In addition to send this data to the CONTROL agent, the RADAR agent also broadcast its data to the allied soldiers who are in the range.

SOLDIER Agent

Each human soldier has a helmet that receives and sends information in real time to the allied soldiers and their commanders. Both human soldier and helmet are represented in this work by the SOLDIER agent. Figure 9 represents the integrated architecture of the SOLDIER agent.

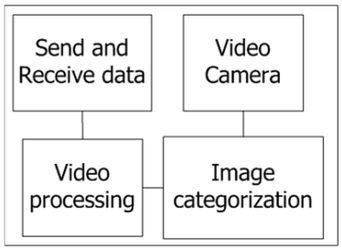


Fig. 9. Architecture of SOLDIER Agent.

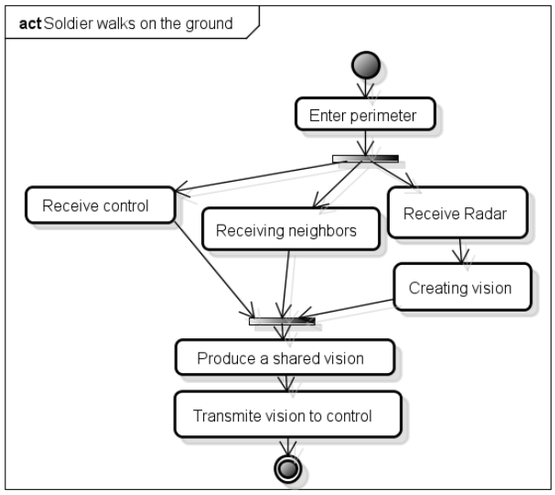


Fig. 10. Behavior of SOLDIER Agent.

The VIDEO CAMARA module of helmet captures images that are within eyesight of the soldier. The IMAGE CATEGORIZATION module recognizes environmental patterns warning the soldier about the position of weapons or recognizable obstacles, present in its active range. The SEND AND RECEIVE DATA module continuously sends the vision rendered in the display of the SOLDIER agent to the CONTROL agent, as well as for the other allied soldiers who are in their range. This module also receives the images of the CONTROL agent and displacement orders. Such data are analyzed by the SOLDIER agent, together with its own information.
In conflict region may be necessary to perform the mapping of obstacles and enemies, keeping the region under surveillance. For combat situations is defined one perimeter for mapping and surveillance. When the SOLDIER agent enters this perimeter, it starts the process of continuous exchange of messages between him, the CONTROL agent and other agents. Figure 10 illustrates this process. Messages from CONTROL agent are shown with priority on the display of the SOLDIER agent, which can orient his moves over the terrain to cover the region and scan for weapons and enemies.

CONTROL Agent

The CONTROL agent is responsible for receiving all messages from SOLDIER agent and RADAR agent, summarizing them in a format of matrix of symbols. Figure 11 describes the behavior of

the CONTROL agent for two types of possible inputs: data transmitted by the SOLDIER agent, and by the RADAR agent.

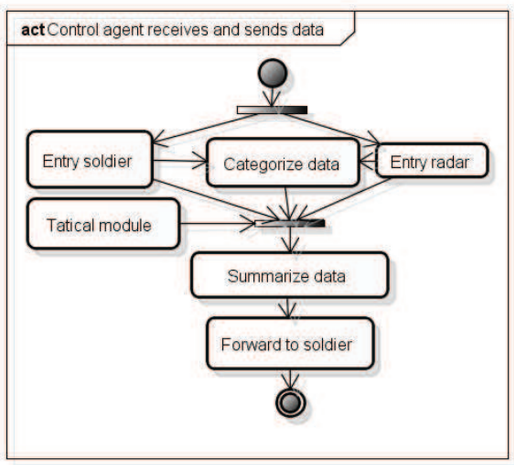


Fig. 11. Behavior of CONTROL Agent.

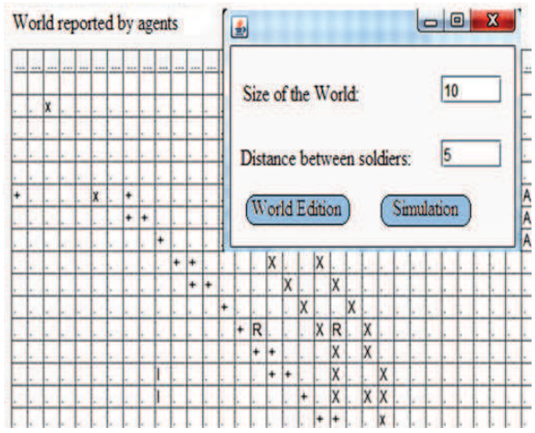


Fig. 12. Symbols Representation in the Matrix of the Environment.

The signal transmitted by the RADAR agent must pass through the IDENTIFICATION AND IMAGE CATEGORIZATION OF THE ENVIRONMENT module to transform the images in standard symbols. If the image comes from the SOLDIER agent, the categorization is not necessary because the images were already been processed in symbols by SOLDIER agent's equipment. The categorization is the identification of elements in the environment. In general the environment contains two types of obstacles: static (trees, rocks, rivers) and mobile (allied soldiers and enemies). When an element is identified by the categorization process, a specific symbol is used to represent this object, aiming to represent it with best level of information. Throughout this process the CONTROL agent builds up an overview of the battlefield and the dynamics of movement that drives the various components of the environment. This process is called aggregation.

In parallel, the TACTICAL module tries to recognize the direction of movement of allied soldiers and enemies, creating a tactical map with the different stored views. At the end of this process, the CONTROL agent builds an overview of the battlefield at every stage of combat. Based on this information, and with the objective to help the decision-making process of the soldiers, CONTROL agent executes two actions: (i) transmits to every soldier a partial view of the battlefield - this vision is related to the action and vision range of every soldier, and (ii) defines progress strategies for each soldier (tactical moves), and sends them individually.

6.2.2 The environment

The environment to be mapped and kept under surveillance can be quite complex in the real world, with natural obstacles such as rivers, lakes, depressions in the ground, buildings and vegetation. In this work, the environment where the combat situation occurs might contain the following elements: soldiers (allies and enemies), radar, trees, rocks and rivers. These elements will be displayed on the visor of each SOLDIER agent and on the interface of CONTROL agent, following the pattern presented in Figure 12.

The ground of the conflict area is represented as a matrix, and in each cell can be allocated an agent in every moment of combat. One of the advantages of using this representation is ease of manipulation for electronic systems. Since the main objective is to improve the speed of

decision making, a lighter processing has many advantages over the use of complex imaging systems. In the Figure 12 the elements of the environment are represented with the following symbols:

- SOLDIER agent is represented by the character (S);
- Enemy soldiers are represented by the character (I) and may be of two types: those who walk and those that keep a fixed position;
- RADAR agents are represented by the character (R) and are inserted into the environment as the soldiers walk around. RADARS do not move;
- Trees are represented by the character (A);
- River banks are represented by the character (X);
- Stones are represented by the character (P) and can be isolated or grouped within the environment representing a rise or other natural obstacle;
- The symbol (+) represents the footprints (or the route) left by an enemy.

It can be seen in Figure 12 the initial screen of the simulation process where the size of the matrix and the distance between SOLDIER agents are chosen. The number of soldiers is calculated by the system as the size of the matrix divided by the distance between SOLDIER agents. As the simulation runs the partial visions of soldiers and radars are analyzed, compiled and aggregated by the CONTROL agent, who later returns for each SOLDIER agent only corresponding the part to his field of vision.

6.2.3 Future works

The application presented in this section describes a society of agents able to map and keep surveillance on any area or region. Therefore, in future works it is possible to extend this research to situations where the control flow of vehicles or persons should be evaluated for safety or improvement of the flow. A good example is the use of radars installed on highways as agents that send information about the average flow of vehicles, and the CONTROL agent with the ability to change the timings of traffic lights.

The implementation of the system using intelligent agents can be economically more attractive because it leverages the current infrastructure and has the characteristic of being resistant to failures. A radar system that does not send information for any reason would not invalidate the control operation of traffic signals, since for evaluating the mean flow of the traffic for decision making, is sufficient to collect the information from the other agents. Other advantages are the fact that the system can anticipate the arrival of a stream of cars and take a preventive action, and detect other abnormal behavior of the traffic.

7. Conclusions

We know that the hardware and software are constantly improving in performance and reliability. The programmers are also worry to improve their ability. For instance, many programmers try to minimize memory usage and to maximize throughput or processing speeds especially in the distributed applications. However, the low cost of the hardware make this problem almost disappear. The multi-agent technique tends to span many applications in different domains. The autonomy in the MAS allows the agents to learn the behaviors of the system to decide automatically, for example, the communication protocols and strategies, without the human intervention. The autonomy is constantly under improvement to make

the agents form and reform dynamic coalitions to pursue goals. In addition, the behaviors, the coordination between heterogeneous agents, faulty agents, languages, protocols may be improved.

The agent community has the interest to broaden the application of the agent technology and also the acceptance of the new paradigm among the researchers who are interesting to develop intelligent systems. The AOP has the potential to provide the best solution in different scenarios of abundance and opportunities, with the ability to create MAS that can work naturally on parallel and distributed environments. Agents that can collaborate and distribute tasks, and that can be built based on human characteristics like belief, desire and intention. The systems will be able to handle large numbers of agentes in the society. Also, the technology is easy to customize to provide new components and services. That is a technology to be followed closely.

8. References

Austin, J. L. (1962). *How to Do Things with Words*, Oxford University Press, London.

Azevedo, S., Teixeira, M., Costa, A., Borsato, B., Soares, F., Lucena, C. & de Janeiro-RJ-Brasil, R. (2008). Multi-agent system for stock exchange simulation-masses, *XXII Fourth Workshop on Software Engineering for Agent-oriented Systems (SEAS) of the Brazilian Symposium of Software Engineer (October 2008)*.

Barbuceanu, M. & Fox, M. (1995). COOL: A language for describing coordination in multi agent systems, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, Citeseer, pp. 17–24.

Bellifemine, F. L., Caire, G. & Greenwood, D. (2007). Developing multi-agent systems with jade (wiley series in agent technology), *John Wiley and Sons*.

Bittencourt, G. (2006). Distributed artificial intelligence.

Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems, *Proceedings of the National Academy of Sciences* 99: 7280–7287.

Bond, A. H. & Gasser, L. (1998). Readings in distributed artificial intelligence, Morgan Kaufmann Publishers, pp. 3–35.

Bratman, M. E. (1987). *Intention, Plans, and Practical Reason*, Harvard University Press, Cambridge, Massachusetts and London.

Buckle, P. & Hadingham, R. (2000). Fipa and fipa-os overview, *Invited talk, Joint Holonic Manufacturing Systems and FIPA Workshop, London*.

Cil, I. & Mala, M. (2010). A multi-agent architecture for modelling and simulation of small military unit combat in asymmetric warfare, *Expert Systems with Applications* 37(2): 1331 – 1343.

Coutinho, L. R., Brandao, A. A. F., Sichman, J. S. & Hubner, J. F. (2009). A model-based architecture for organizational interoperability in open multiagent systems, *Proc. of the International Workshop on Coordination, Organization, Institutions and Norms in Agent Systems*, pp. 128–140.

de Nunes, I. O. (2007). Implementation of the bdi waho architecture model.

Dennett, D. (1987). *The Intentional Stance*, MIT Press.

d'Inverno, M., Kinny, D., Luck, M. & Wooldridge, M. (1998a). A formal specification of dmars, *Intelligent Agent IV: 4th International Workshop on Agent Theories* . Architectures and Languages.

d'Inverno, M., Kinny, D., Luck, M. & Wooldridge, M. (1998b). A formal specification of dmars. intelligent agent, *4th Int. Workshop on Agent Theories*.

E.H. Durfee, V. L. (1989). Trends in cooperative distributed problem solving, 11(1): 63–83.
Fama, E. (1965). The behavior of stock-market prices, *Journal of business* 38(1).
Faraco, R. A. (2001). *Uma arquitetura de agentes para a negociação dentro do domínio do comércio eletrônico*, Master's thesis, Universidade Federal de Santa Catarina - UFSC, Brasil.
Fennel, R. D. & Lesser, V. (1977). Parallelism in artificial intelligence problem solving: A case study of gearsay ii, 26(2): 98–111.
Gottifredi, S., Tucat, M., Corbatta, D., Garcia, A. J. & Simari, G. R. (2010). A bdi architecture for high level robot deliberation.
URL: *http://dialnet.unirioja.es/servlet/oaiart?codigo=3214348*
Group, K. A., Finin, T., Mckay, D. & Fritzson, R. (1992). An overview of kqml: A knowledge query and manipulation language.
Hewitt, C. (1977). Viewing control structures as patterns of passing messages., 8: 323–364.
Huhns, M. N. (2004). Software development with objects, agents, and services, *Proc. of Third Int. Workshop on Agent-Oriented Methodologies, OOPSLA*.
Huhns, M. N. & Stephens, L. M. (1999). Multiagent systems and societies of agents, pp. 79–120.
Jennings, H. (1995). Controlling cooperative problem solving in industrial multi-agent systems using joint intention, 2(75): 195–240.
Jennings, N. R. & Wooldridge, M. (2000). Agent-oriented software engineering, *Artificial Intelligence* 117: 277–296.
Lind, J. (2000). Issues in agent-oriented software engineering, *Agent-Oriented Software Engineering III*, Springer Verlag, pp. 45–58.
Marietto, M. G. B. (2000). *Definição Dinâmica de Estratégias Instrucionais em Sistemas de Tutoria Inteligente: Uma Abordagem Multiagentes na WWW*, PhD thesis, Instituto Tecnológico da Aeronáutica - ITA.
Nwana, H., Lee, L. & Jennings, N. (1997). Coordination in software agent systems, p. http://www.labs.bt.com/projects/agents/publish/papers/coorination.html.
Odell, J. (1999). Objects and agents: How do they differ? (draft 2.2), *url: http://www.jamesodell.com*.
Odell, J. (2002). Objects and agents compared, *J. of Object Technology* 1(1): 41–53.
Panzarasa, P., Jennings, N. & Norman, T. (2002). Formalizing collaborative decision-making and practical reasoning in multi-agent systems, *Journal of logic and computation* 12(1): 55.
Parmigiani, G., Inoue, L. Y. T. & Lopes, H. (1997). *Decision Theory: Principles and Approaches*, John Wiley and Sons.
Paulussen, T. O., Zöller, A., Heinzl, A., Braubach, L., Pokahr, A. & Lamersdorf, W. (2004). Dynamic patient scheduling in hospitals. coordination and agent technology in value networks.
Poslad, S., Buckle, P. & Hadingham, R. (2000). The fipa-os agent platform: Open source for open standards, *Proceedings of the 5th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents*, Vol. 355, Citeseer, p. 368.
Rao, A. S. & Georgeff, M. P. (1995). Bdi agents: From theory to practice, *Proc. of the First Intl. Conf. Multiagent Systems*.
Reichel, K., Hochgeschwender, N. & Voos, H. (2008). Opcog: An industrial development approach for cognitive agent systems in military uav applications, *Proc. of the 7th int. Joint Conf. on Autonomous Agents and Multiagent Systems: Industrial Track*, pp. 97–100.
Reis, L. P. (2003). *Coordenação em Sistemas Multi-Agentes: Aplicações na Gestão Universitária e Futebol de Robôs*, PhD thesis, FEUP - Portugal.

Rentsch, T. (1982). Object oriented programming, *SIGPLAN Notices* 17(12): 51.

Rutterford, J. & Davison, M. (1993). *An Introduction to Stock Exchange Investment*, MacMillan.

Sawyer, R. K. (2003). Artificial societies: Multiagent systems and the micro-macro link in sociological theory, *Sociological Methods and Research* 31: 325–363.

Searle, J. R. (1969). *Speech Acts: An Essay in the Philosophy of Language*, Cambridge University Press.

Shoham, Y. (1993). Agent oriented programming, *Artif. Intell.* 60(1): 51–92.

Silva, V., Garcia, A., Brandão, A., Chavez, C., Lucena, C. & Alencar, P. (2003). Taming agents and objects in software engineering, *Software Engineering for Large-Scale Multi-Agent Systems: Research Issues and Practical Applications, volume LNCS 2603*, Springer-Verlag, pp. 1—25.

Wagner, G. (2003). The agent-object-relationship meta-model: Towards a unified view of state and behavior, *Information Systems* 5(28): 475–504.

Wahono, R. S. (2001). *Intelligent agents for object model creation process in object-oriented analysis and design*, Master's thesis, Department of Information and Computer Sciences, Graduate School of Science and Engineering, Saitama University, Saitama - Japan.

Wei, Y.-M., jun Ying, S., Fan, Y. & Wang, B.-H. (2003). The cellular automaton model of investment behavior in the stock market, *Physica A: Statistical Mechanics and its Applications* 325: 507–516.

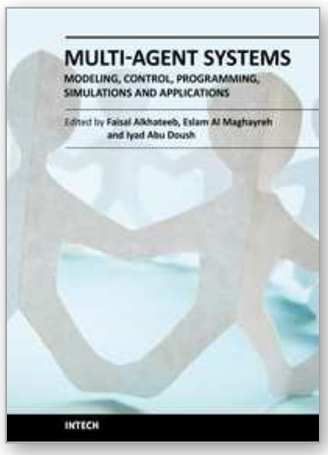
Wijngaards, N., Overeinder, B., van Steen, M. & Brazier, F. (2002). Supporting Internet-scale multi-agent systems, *Data & Knowledge Engineering* 41(2-3): 229–245.

Wooldridge, J. M. (1999). Intelligent agents, *Multiagent Systems: a Modern Approach to Distributed Artificial Intelligence* pp. 27–77.

Wooldridge, M. & Jennings, N. R. (1995). Intelligent agents: Theory and practice, *The Knowledge Engineering Review* 10(2): 51.

Wooldridge, M. & Jennings, N. R. (1998). Pitfalls of agent-oriented development, *Proc. of the Second Int. Conf. on Autonomous Agents (Agents 98*, ACM Press, pp. 385–391.

Xuan, P., Lesser, V. & Zilberstein, S. (2001). Communication decisions in multi-agent cooperation: Model and experiments, *Proceedings of the fifth international conference on Autonomous agents*, ACM, pp. 616–623.



Multi-Agent Systems - Modeling, Control, Programming, Simulations and Applications

Edited by Dr. Faisal Alkhateeb

ISBN 978-953-307-174-9
Hard cover, 522 pages
Publisher InTech
Published online 01, April, 2011
Published in print edition April, 2011

A multi-agent system (MAS) is a system composed of multiple interacting intelligent agents. Multi-agent systems can be used to solve problems which are difficult or impossible for an individual agent or monolithic system to solve. Agent systems are open and extensible systems that allow for the deployment of autonomous and proactive software components. Multi-agent systems have been brought up and used in several application domains.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

André Filipe de Moraes Batista, Maria das Graças Bruno Marietto, Wagner Tanaka Botelho, Guiou Kobayashi, Brunno dos Passos Alves, Sidney de Castro and Terry Lima Ruas (2011). Principles of Agent-Oriented Programming, Multi-Agent Systems - Modeling, Control, Programming, Simulations and Applications, Dr. Faisal Alkhateeb (Ed.), ISBN: 978-953-307-174-9, InTech, Available from: http://www.intechopen.com/books/multi-agent-systems-modeling-control-programming-simulations-and-applications/principles-of-agent-oriented-programming

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen