REACTIVE AND DELIBERATIVE AGENTS APPLIED TO SIMULATION OF SOCIO-ECONOMICAL AND BIOLOGICAL SYSTEMS

MARCO REMONDINO

Lagrange Project for Complex Systems, University of Turin, Italy
E-mail:remond@di.unito.it

Abstract: An important open issue in the Agent Based Simulation field is the lack of an univocal definition of the term "agent" and of the paradigms and methodology used to build models**;** in software-engineering, a system of independent programs is considered a multi-agent system, although there is no clarity about what the term exactly defines. The term "agent", deriving from the Latin "*agens*", identifies someone (or something) who acts; the same word can also be used to define a mean through which some action is made or caused. The term is used in many different fields and disciplines, such as economics, physics, natural sciences, sociology and many others. In this paper an overview of different kinds of agents, that could all be applied to simulation of complex systems, will be presented; in particular, some practical examples will be given of models pertaining Game Theory, Biology and Social Organizations. The innovative contribution of the article is the use of declared agent paradigms, derived from computer science and Artificial Intelligence, for simulating complex systems.

Keywords: Social Simulation, Intelligent Agents, Complex Systems, Biology, Economics

1. INTRODUCTION

In the field of Agent Based Simulation (ABS), the concept of agent is often fuzzy and – sometimes – even abused. In Drogoul et al. (2002) the authors argue that, ABS, despite its name, is in fact rarely based on computational agents. This is because the semantics associated differ considerably from one model to another, or from one implementation to another. As a matter of fact, in most ABS research works, the languages used by domain experts, modellers and computer scientists, while syntactically coherent, often hide important semantic disparities. Sometimes, in order to simplify the implementation, the "agent" concept is only present at the modelling stage, while it is lost at the implementation step, where it is substituted by simpler "objects", according to the Object Oriented languages terminology. In this article two paradigms of software agents will be examined from the theoretical point of view: the reactive ones, i.e. very simple entities, embedded with the ability to sense the environment and to respond to the stimuli coming from it and from other agents, and the deliberative ones, endowed with a logic-based mind and with the capacity of creating autonomous plans to reach their design objectives. Both these paradigms, even if very different from each other, can be successfully employed to build simulations of complex systems; in order to show this, after the theoretical introduction, some practical examples will be given of different situations, coming from different fields (Game Theory, Computer Science, Biology, Economics), studied and modelled with these different approaches.

2. BACKGROUND ON AGENT BASED SIMULATION

Agent Based Modelling is the most interesting and advanced approach for simulating a complex system: in a social context, the single parts and the whole are often very hard to describe in detail. There are agent based formalisms which allow to study the emergency of social behaviour, through the development and use of models, known as artificial societies. Thanks to the ever increasing computational power, it has been possible to use such models to create software, based on intelligent agents, whose aggregate behaviour is complex and difficult to predict, and can be used in open and distributed systems.

The concept of software agent originates in the early fifties with J. McCarthy, while the term has been coined by O.G. Selfridge some years later, when both of them were working at the Massachusetts Institute of Technology. Their original project was to build a system which, given a goal, could be able to accomplish it, looking for human help when the necessary information is not available. In practice, an agent was considered a software robot that lives and acts in a virtual world. In (Wooldridge and Jennings 1995): "... a hardware or (more usually) software-based computer system that enjoys the following properties:

• autonomy: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;
• social ability: agents interact with other agents (and possibly humans) via some kind of agent-communication language;
• reactivity: agents perceive their environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the internet, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it;
• pro-activeness: agents do not simply act in response to their environment, they are able to exhibit goal-directed behaviour by taking the initiative." The Wooldridge and Jennings definition, in addition to spelling out autonomy, sensing and acting, allows for a broad, but finite, range of environments. They further add a communications requirement.

Franklin and Graesser (1997) also try to find the typical features of agency, deriving them from the word itself: an "agent" is 1) one who acts, or who can act, and 2) one who acts in place of another with her/his permission. Since "one who acts in place of " acts, the second usage requires the first. Humans act, as do most other animals. Also, some autonomous mobile robots act, for example Brooks' Herbert (Brooks 1990; Franklin 1995). All of these are real world agents. Software agents "live" in computer operating systems, databases, networks, MUDs, etc.
Finally, artificial life agents "live" in artificial environments on a computer screen or in its memory (Langton 1989, Franklin 1995).
Each is situated in, and is a part of some environment. Each senses its environment and act autonomously upon it. No other entity is required to feed it input, or to interpret and use its output. Each one acts in pursuit of its own agenda, whether satisfying evolved drives as in humans and animals, or pursuing goals designed in by some other agent, as in software agents. (Artificial life agents may be of either variety.) Each acts so that its current actions may effect its later sensing, that is its actions effect its environment. Finally, each acts continuously over a specified period of time. A software agent, once invoked, typically runs until it decides not to. An artificial life agent often runs until it is eaten or otherwise dies. Of course, some human can pull the plug, but this is not always the case. These requirements constitute for sure the essence of being an agent, hence the definition by Franklin and Graesser (1997):

An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it will sense in the future.

And the very general, yet comprehensive one by Jennings (1996):

…the term is usually applied to describe self-contained programs which can control their own actions based on their perceptions of their operating environment.

3. DIFFERENT KINDS OF AGENTS AND MAS

Agents themselves have traditionally been categorized into one of the following types (Woolridge and Jennings, 1995):

• Reactive
• Collaborative/Deliberative
• Hybrid

When designing any agent-based system, it is important to determine how sophisticated the agents' reasoning will be. Reactive agents simply retrieve pre-set behaviours similar to reflexes without maintaining any internal state. On the other hand, deliberative agents behave more like they are thinking, by searching through a space of behaviours, maintaining internal state, and predicting the effects of actions. Although the line between reactive and deliberative agents can be somewhat blurry, an agent with no internal state is certainly reactive, and one which bases its actions on the predicted actions of other agents is deliberative.
In Mataric (1995) we read that reactive agents maintain no internal model of how to predict future states of the world. They choose actions by using the current world state as an index into a table of actions, where the purpose of the indexing function is to map known situations to appropriate actions. These types of agents are sufficient for limited environments where every possible situation can be mapped to an action or set of actions.
The major drawback of the purely reactive agent is its lack of adaptability. This type of agent cannot generate an appropriate plan if the current world state was not considered a priori. In domains that cannot be completely mapped, using reactive agents can be too restrictive.
Different from reactive agents are the deliberative ones. The key component of a deliberative agent is a central reasoning system (Ginsberg, 1989) that constitutes the intelligence of the agent. Deliberative agents generate plans to accomplish their goals. A world model may be used in a deliberative agent, increasing the agent's ability to generate a plan that is successful in achieving its goals even in unforeseen situations. This ability to adapt is desirable in a dynamic environment.
The main problem with a purely deliberative agent when dealing with real-time systems is reaction time. For simple, well known situations, reasoning

may not be required at all. In some real-time domains, such as robotic soccer, minimizing the delays between changes in world state and reactions is important.
Hybrid agents, when designed correctly, use both approaches to get the best properties of each (Bensaid and Mathieu, 1997). Specifically, hybrid agents aim to have the quick response time of reactive agents for well known situations, yet also have the ability to generate new plans for unforeseen situations.

3.1 Multi Agent Systems

A Multi Agent System (MAS) can be thought of as a group of interacting agents working together to achieve a set of goals. MAS are used to describe several agents that interact with each other (positively, but also negatively). Positive interaction is usually known as cooperation, collaboration is used as a more elaborate word for interaction, and competitive settings usually describe systems where negative interaction takes place.

To maximize the efficiency of the system, each agent must be able to reason about other agents' actions in addition to its own. A dynamic and unpredictable environment creates a need for an agent to employ flexible strategies. The more flexible the strategies however, the more difficult it becomes to predict what the other agents are going to do. For this reason, coordination mechanisms have been developed to help the agents interact when performing complex actions requiring teamwork. These mechanisms must ensure that the plans of individual agents do not conflict, while guiding the agents in pursuit of the goals of the system.

The following property dimensions are an extension of the dimensions reported in Sridharan (1986):

- System model: individual, team, society
- Granularity: fine grained, coarse grained
- Number of agents: small, medium, large
- Adaptability of agents and the whole MAS: fixed, programmable, able to learn, autodidactic
- Control distribution: being controlled, dependent, independent
- Resources: limited, rich, unlimited
- Interaction scheme: simple, complex
- Solution strategy: synthesis, analysis
- Degree of cooperation between agents: cooperative and selfless, competitive and hostile

MAS are perfect for social simulation, because they are configurable and they can represent the real system to any degree of granularity. Besides, complex systems can also be modelled using MAS, since they are the composition of many parts interacting with one another.

4. COMPLEX SYSTEMS AND SIMULATION

There are many accepted definition for the word “complexity”, when applied to a social system, i.e.: a system in which the single parts interact among them. The first and most straightforward one is the following Pavard and Dugdale, (2000):

A complex system is a system for which it is difficult, if not impossible to restrict its description to a limited number of parameters or characterizing variables without losing its essential global functional properties.

Formally, a system starts to have complex behaviours (non-predictability and emergence etc.) the moment it consists of parts interacting in a non-linear fashion. According to this, a complex system is defined as:

…the interaction of many parts, giving rise to difficulties in linear or reductionist analysis due to the nonlinearity of circular causation and feedback effects (Calresco Glossary).

It is thus appropriate to differentiate between complicated systems (such as a plane or computer) and complex systems (such as ecological or economic systems). The former are composed of many functionally distinct parts but are in fact predictable, whereas the latter interact non-linearly with their environment and their components have properties of self-organization which make them non-predictable beyond a certain temporal window.
A truly complex system would be completely irreducible. This means that it would be impossible to derive a model for the behaviour of her/his system (i.e. a representation simpler than reality) without losing some of its relevant properties. However, in reality different levels of complexity obviously exist. If situations which are highly structured and governed by stable laws are to be modelled, then it is possible to represent and model the system by simplification, without loosing too many of the system’s properties. Thus, the essential question is to know to what extent the properties of the analyzed and designed socio-technical systems fall into one or the other of these situations. In other words, to what extent an abstraction of microscopic interactions can be made in order to understand macroscopic behaviours. In what measure microscopic interactions are linked in a non-reducible way with the laws that govern more structured behaviours and, finally, it is necessary to check if it is possible to explain the most structured behaviour using rules which control the microscopic behaviour. This last

question is important from an epistemological and methodological point of view: if theoretical economy is considered, it can be preferable to generate the structural property of a system using knowledge of its microscopic properties (emergence), rather than suggest its macroscopic properties and only validate them with an analytical process.
The reduction of complexity is an essential stage in the traditional scientific and experimental methodology (also known as analytic). After reducing the number of variables (deemed most relevant), this approach allows for systems to be studied in a controlled way, i.e. with the necessary replication of results. This approach in itself need not be questioned. However, when considering complex socio-technical systems it is appropriate to analyze precisely the limits of the approach.

4.1 Simulating Complex Systems Using Agents

Modelling is a way of solving problems that occur in the real world. It is applied when prototyping or experimenting with the real system is expensive or impossible. Modelling allows to optimize systems prior to implementation. It includes the process of mapping the problem from the real world to its model in the world of models, – the process of abstraction, – model analysis and optimization, and mapping the solution back to the real system. A distinction can be made between analytical and simulation models. In analytical, or static, model the result functionally depends on the input (a number of parametersin simple cases, it is possible to implement such a model in a spreadsheet.

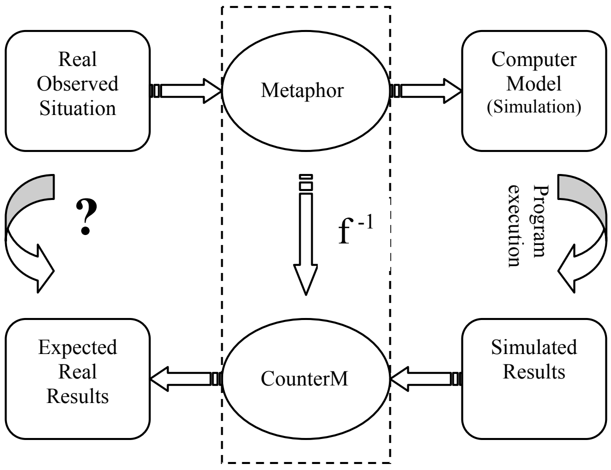

Figure 1: from the Real Situation to the Model

However, an analytical solution does not always exist, or may be very hard to find. Then simulation, or dynamic, modelling may be applied. A simulation model may be considered as a set of rules (e.g. equations, flowcharts, state machines, cellular automata) that define how the system being modelled will change in the future, given its present state. Simulation is the process of model "execution" that takes the model through (discrete or continuous) state changes over time. In general, for complex problems where time dynamics is important, simulation modelling is a better answer. In Figure 1, the metaphor based approach (Remondino, 2003) is shown, illustrating how to step from a real observed situation (problem in the real world) to a computer model and hence to a simulation in order to obtain results that can be scaled back and applied to the original problem.
The metaphor layer is a conversion one, and works like a function, which maps a real situation onto a computer program, that can be executed by a machine. The results obtained by the simulation built with this approach, do not necessarily apply one-to-one to the real situation. Therefore, an inverse function is required, which makes them suitable for the observed reality; this inverse function, called counter-metaphor, has to be directly derived from the metaphor used to translate the observed system into the simulated model. This counter-metaphor will allow to go back from the results obtained from the model to others that can be compared to the real data.

In an AB model, there is not a place where the global system behaviour (dynamics) would be defined. Instead, the modeller defines behaviour at the individual level, and the global behaviour emerges as a result of many (tens, hundreds, thousands, millions) individuals, each following its own behaviour rules, living together in some environment and communicating with each other and with the environment. That is why AB modelling is also one type of bottom-up modelling.
The agent-based view takes a different approach to modelling. Instead of creating a simple mathematical model, the underlying model is (based on) a system (comprised) of various interacting agents. Therefore, its structure and behaviour have the potential to resemble the actual economic theory and reality better than simple mathematical models. Especially, when the underlying real relationships are complex.
In (Bonabeau, 2002), we read that AB paradigm can be used successfully to model different situations, like flows, markets, organizations, social diffusion of phenomena, and all these are indeed complex systems.

5. PRACTICAL EXAMPLES OF ABS

In the following sub-sections some practical examples will be given about the use of reactive and deliberative agents applied to the simulation and modelling of complex systems in different fields; in particular it will be shown how they can be used in Game Theory, an interdisciplinary field belonging to Experimental Economics; in Biology, through the simulation of colonies of periodical insects; in Organizational Sociology, through a practical example of a group of people accessing a database, and a theoretical example of an enterprise at a macro level, modelled using deliberative structured agents. All the models shown have been created by using the JAS library (http://jaslibrary.sourceforge.net/).

5.1 Simulating a Minority Game

The Minority Game (MG) is a simple, generalized framework, belonging to the Game Theory field, which represents the collective behavior of agents in an ideal situation where they have to compete through adaptation for some finite resource.
While the MG is born as the mathematical formulation of "El Farol Bar" (EFB) problem considered by (Arthur, 1994), it goes way beyond this one, since it generalizes the study of how many individuals may reach a collective solution to a problem by adapting to each other's expectations about the future. The original formulation of EFB problem is as follows: N people, at every step, take an individual decision among two possibilities. Number one is to stay at home; number two is to go to a bar. Since the space in the bar is limited (finite resource), the time there is enjoyable if and only if the number of the people there is less than a fixed threshold (aN, where a<1). Every agent has its own expectation on the number of people in the bar and according to its forecast decides whether to go or not: if there were an obvious model that all agents could use to forecast attendance and base their decisions on, then a deductive solution would be possible. But this is not the case here. Given the numbers attending in the recent past, a large number of expectational models might be reasonable and defensible. Thus, not knowing which model other agents might choose, a reference agent cannot choose her/his in a well-defined way. The only information available to the agents is the number of people going to the bar in the recent past; this means that there is no deductively rational solution to this problem, but there can be a multiplicity of models trying to infer the future number according to the past ones.
The EFB problem has been applied to some proto-market models: at each time step agents can buy (go to the bar) or sell an asset and after each time step, the price of the asset is determined by a simple supply-demand rule. The MG has been first described in (Challet and Zhang, 1997) as a mathematical formalization and generalization of EFB problem. It is assumed that an odd number of players take a decision at each step of the simulation; the agents that take the minority decision win, while the others loose.
The EFB problem, as well as the MG in its original formulation state that there is no communication among the agents involved in the simulation; the idea in this example is to introduce in the model a sort of a social network (Remondino and Cappellini, 2004), in order to see how the links among certain agents can change the results of the simulation. A social network is defined as "a set of nodes - e.g. persons, organizations - linked by a set of social relationship - e.g. friendship, transfer of funds, overlapping membership - of a specific type" (Laumann, et al., 1978). This example shows that even using simple, reactive agents, realistic results can be obtained from the simulation of a complex social phenomenon.

5.2 Simulation Framework and Results

The simulation framework can be described as follows: in the model, reactive agents are used, linked into a social network (Remondino and Cappellini, 2005). During the setup, a simple world populated by N agents is created. These agents can be considered as the vertices of a social network and the links among them (relations) as the edges. The network is oriented and every arc is composed by two edges with opposite directions. Every agent has a list of F (friends) other agents (called friendsList) (to) whom it can communicate with. This list is composed by the neighbours, i.e. the vertices linked to the (examined) vertex under consideration (the agent).
(Here follows) A brief description of the simulation process is provided in the following:

- At the beginning of each simulation step, every agent has its own forecast. The forecast is absolutely random between two choices –1 and +1.
- The decision taken by each agent (before communicating with others) is denoted with a "certainty index" equal to 1 (100%).
- Now an agent is randomly chosen. It starts asking (to) the first in the list; if this one has the same expectation, then the certainty index is increased by a value of 1/F, while if the prevision is different, than the certainty index is lowered by 1/F (notice that the index can be greater than 100%; this is done for modelling reasons, even if in practice it woudn't make any sense)
- After having asked to all the friends in its list, the agent takes the final decision: if the certainty index is equal or greater than 1, then the decision will be the original one. If it is smaller than 1, then the decision will be the other possible one
- Another agent is then randomly chosen, and so on (the same agent can't be chosen twice during the same decision step). Note that an agent that has been asked can still change its mind, basing on the agents it will in turn ask.

In the output graph the time can be read on *x-axis* (1000 iterations of the game), and two variables are plotted: the lower one depicts the number of changed decisions while the higher one is for unchanged decisions. On the *y-axis* the number of decisions is represented (changed or not); the scale (10^1, 10^2, 10^3) depends on the number of agents. As standard example a world of 100 agents and 500 relations is shown (Figure 2), in which 65 out 100 preserve their original decisions. In a second run a different situation is imagined, in which the agents have many more relations among them: an average of fifty for every individual (Figure 3). A simple common sense rule states that the larger the number of relations, the higher is the probability that expectations are changed.

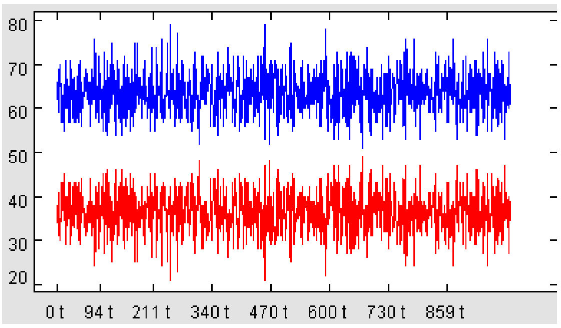

Figure 2: 100 Agents and 500 Links

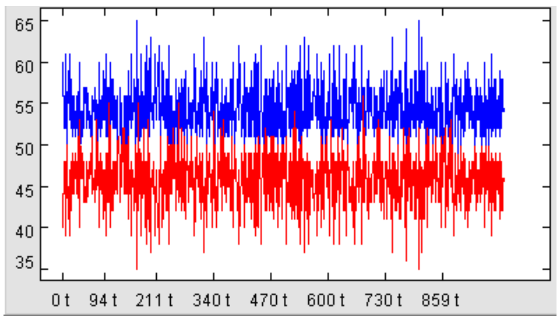

Figure 3: 100 Agents and 5000 Links

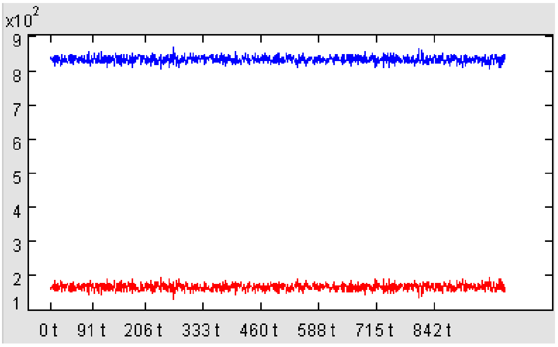

Figure 4: 1000 Agents and 500 Links

This example shows the rule to be right and the model to be consistent with real world results; a counter example is now shown, i.e. a poor relations world, as the one in Figure 4; one thousand individuals with a total of just five hundred relations. This very simple example shows that reactive agents, i.e. software entities simply endowed with the capability of sensing the environment (in this case, using a social network to know what others will do) and with a simple function (the one used to take the decision) can present a realistic aggregate behavior. In this case, with some very simple rules a community of agents is simulated and a rule of thumb emerges: the more relations in a community of agents, the more the probability of changing the original opinion. This can apply, for instance, to marketing studies or even to political campaigns.

5.3 ABS in Biology: Insects' Dynamics

Biology is an important area of reference for agent based modelling, both as a source of examples/applications and as methodological inspiration. Many important results in Artificial Life and Computer Science are based on biological metaphors such as the *ALife* concept itself and, in detail *Game of Life* by John Conway, Holland's *Genetic Algorithms*, *Neural Networks* and many others. Also the simple ants or "*vants*" (virtual ants) are worth mentioning, with their simple rules and their simulated pheromones that are present in a large number of models and algorithms, since the Langton's ant.

One of the most interesting fields to which ABS can be applied is the behaviour of simple insects and the predators/pray dynamics. In the following a real phenomenon is explored using an agent based model; there are two species of cicada, called Magicicada Septendecim and Magicicada Tredecim, which have a life cyle of 17 and 13 years respectively. These are among the longest living insects in the world; they display a unique living behaviour, since they remain underground for all but their last few weeks of life, when they emerge "en masse" from the ground into the forest where they sing, mate, eat, lay eggs and then die. The nymphs of the periodical cicadas live underground, at depths of 30 cm (one foot) or more, feeding on the the juices of plant roots. They remain still and go through five development stages before creating an exit tunnel in the spring of their 13th or 17th year. Adult periodical cicadas live only for a few weeks: by mid-July, they will all be gone. Their short life has one sole purpose: reproduction. After mating, the male weakens and dies. The female lives a little longer in order to lay eggs: it makes between six and 20 V-shaped slits in the bark of young twigs and deposits up to 600 eggs there. Shortly thereafter, the female also dies. After about six to ten weeks, the eggs hatch and the newborn nymphs drop to the ground, where they burrow and begin another 13 or 17 year cycle.

The fact that they have both evolved prime number life cycles is thought to be key to their survival. Many are the hypotheses as to why these insects display these life cycle lengths. Two are the most interesting ones, and both of them imply an adaptation by evolution of these species. The first one is about limited resources, that must be shared by the insects once out. By evolving life cycles of 17 and 13 years, the two species only have to share the forest floor every 221 years, that's 13 times 17. Resource limitations can then be considered as an upper limit for the cicadas: no more than a threshold number could survive with the available food, so the fewer insects that are out at the same time, the better.

The second hypothesis, actually interacting with the first one, is somewhat opposite to it; it is called the predator satiation hypothesis and moves the focus from the insects to their main predators: dogs, cats, birds, squirrels, deer, raccoons, mice, ants, wasps, and even humans make a meal of the cicadas. Predator satiation is when a species can survive because its abundance is so great that predators do not have a large enough impact to effect the species' survival. In order to prove that predator satiation is occurring in a certain situation one must prove that above a certain prey density, the frequency of predation does not increase as the prey density increases (Williams et al.,1993).

In the case of magicicadas this has been proven to occur. When the first cicadas emerge from the soil there is a very high predation rate, especially from avian predators. However, the predation rates decline over the next couple of days as predators have indulged in all the food they needed, or "satiated". Then, by the time that the satiation of the predators has worn off and foraging activities increase again, the density of adult cicada's has begun to decline after they have mated. This creates a situation where only a small portion of the adult population is consumed by predators. This is indeed a sort of lower bound for the number of cicadas that can be out at the same time; in short, the more cicadas are out at the same time, the least the possibility of being decimated by predators. Also according to this hypothesis, the prime numbers have a motivation: the prime number cycles were selected (for) because they were least likely to emerge with other cycles. For periodical cicadas emerging with other cycles of cicadas would mean hybridization, which would split up populations, shift adult emergences, and create lower densities below the critical size (Yoshimura, 1996). This would have made it harder for the hybridized broods to survive predation. Thus, prime number cycles which emerged with other cycles the least, would grow in population size over time because they would have the highest survival rates.

Under there two working hypotheses it can be assumed that there has been a process of selection among the species through many generations or, better a "tacit communication by evolution". This way, the prime number cycles can be seen a very interesting emergent natural phenomenon, where the conclusions (results) were not embedded in any way into the initial data. For this reason, Agent Based Simulation (ABS) was used to model this phenomenon, i.e. a tool allowing to capture emergent behaviour arising from complex systems. Using an agent-based model of the cicadas' life cycle, the world in which they live is simplyfied and reduced to just few parameters, essentially the limited resources and the predators. The cicadas have a reproduction rate, and so do the predators and the food; will these parameters be enough for prime number life cycles to emerge?

5.4 The Model

An agent based model was created in order to simulate different situations in which many species of cicadas compete for finite resources and are threatened by some predators. The only difference among the species, in the model, is the duration of their life cycle; population #1 will have a one year long life cycle, while population #20 a twenty years life cycle and so on. This section describes the model as it has been implemented; variable names are in italic.

A world is defined, with a fixed amount of resources (resources), that can be consumed by cicadas, a fixed amount of predators (*predatorsNumber*) and a probability to survive at wake up (*chanceToSurviveAtBirthRate*). Except for the costrain represented by food (resources), the other (*chanceToSurviveAtBirth* and predators) can be switched on and off, through the parameters window.

The population of cicadas is characterized by the number of members (*magicicadasNumber*), randomly distributed among the different classes and by a growth rate (*reproductionRate*).

A fixed number of predators is used in order to underline that their population is not influenced by cicadas; In fact these insects represent only an alternative food, among the many present in nature.

At the beginning of the simulation the cicadas are uniformly randomly distributed among *C* classes/sub-species. In this metaphorical world, each class of cicadas has a different life cycle length, so that they wait underground for a different number of years, from a minimum of 1 to *maxSleepingTime*. 20 was used as a maximum number of years a cicada can live, since in nature the *magicicada septendecim* is already the longest living insect and this way unrealistic results won't be produced. However, the model supports whatever number as a maximum life cycle.

Each simulation step represents one year; at any step the model inquires every cicada in order to update/reduce the number of years left for it to stay underground, or if this time is over, wake it up.

The probability *chanceToSurviveAtBirthRate* also increases every year they spent underground. This is done according the biological theory that a longer life cycle is usually a good achievement. In particular, for the cicadas, Yoshimura (1996) points out that during the climate cooling of the Glacial period, growth and development of cicada nymphs was slowed down by lowering soil temperature. He supports this by pointing out that it is well known that cumulative temperature is very critical in insect development. Also, the fact that the cooling climate slowed down the development of host plants from which cicadas get their nutrient may have also slowed their development. Because cicadas needed proper nutrients at each stage of their developmental cycle, and they were provided less because of the cooling temperatures, their life cycle was extended to larger range of years. So in the model the *chanceToSurviveAtBirthRate* is increased at any simulation step, to reproduce this natural phenomenon. When the cicadas have to go outside (i.e.: when their time underground is over), they perform a first check according to this probability.

As soon as they go outside, the cicadas must eat and reproduce themselves. Since the resources are limited, in the model only a certain number of cicadas can survive at each year/tic; this is a strong constraint existing also in the real world; if there are *n* resources in the world, at time *t*, only *n* cicadas will survive.

Then predators can also eat cicadas, and reduce the population. This is the second constraint, present in the real world. In order to fulfil the "predator satiation" hypothesis in the model the predators are satiated after eating a fixed number of cicadas, and so they let the other cicadas live and reproduce.

Finally the surviving cicadas can reproduce and die; the cicadas are then cloned according to *reproductionRate*: the new cicadas have the same properties of their parents, and start a new life cycle underground, according to their duration.

Obviously, before each step the list of cicadas who are still alive is shuffled, in order to randomise the process.

5.5 Simple Model, Realistic Results

In the following some interesting results coming from the simulation model are presented; many exercises can be done by changing the parameters to unrealistic ones, but here the more realistic ones are examined, in order to show that ABS can be a very effective scaled representation of the real world. In particular, you'll see that realistic results have been obtained just by considering the predator/prey/food dynamics and the simple rule of the real life, which states that, whenever possible, "longer is better".

These are the parameters used for this experiment:

- magicicadasNumber: 10000
- reproductionRate: 6.0
- chanceToSurviveAtBirth: true
- chanceToSurviveAtBirthRate: 0.15
- predators: true
- predatorsNumber: 190
- resources: 1000
- maxSleepingTime: 20

During first 20 steps (figure 5) the cicadas with the shortest life cycles died, according to their low *chanceToSurviveAtBirthRate*. At this time nothing can be said about the trend, yet, since most of the cicadas have been out for just one time and the food and predators dynamics have not yet influenced the results.

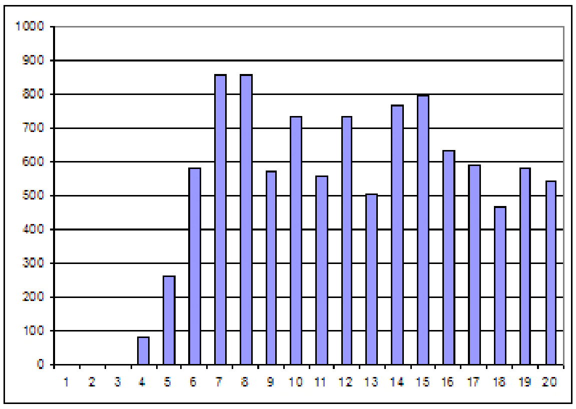

Figure 5: Classes Still Alive after 20 Tics

Normally, the even classes are the least likely to survive, since they have many divisors. This causes many cicadas to be out at the same time,which means that they starve to death according to the limited resources constraint. As it can be observed in figure 6, all the even classes are gone, but class #20. This is because it is the longer living one (and in the model "longer is better") and because it is out less then others. Notice that at this time (after 1000 years) four out of six classes which still exist have a prime number based life cycle.

In the absence of the *chanceToSurviveAtBirthRate*, which increases at each time step the probability of a class to survive and go out, a different effect would emerge: in fact the populations with longest sleeping time are not competitive as the others, since they don't take advantage from the reproduction rate, being out less often. This would be highly unrealistic: the classes with a short life cycle (one to three years) will have much more chances to go out and reproduce, when compared to those with longer cycles. As said before, nature often prefers longer life cycles, when possible, so the model holds.

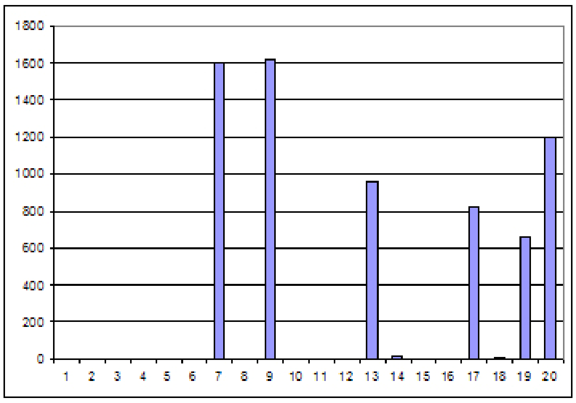

Figure 6: Classes Still Alive after 1000 Tics

In figures 7 and 8 the typical trend of all the experiments can be observed, with a stable population concentrated in few classes.

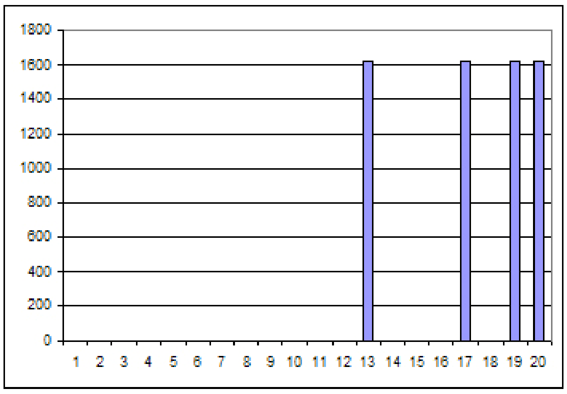

Figure 7: Classes Still Alive after 5000 Tics

In particular, with the parameters listed above, after 5000 steps there remain three high prime numbers (13, 17 and 19) and the longest possible class (20), while after 20000 cycles the highest reproduction rate of the lower classes wins over the longest living one, and the only three remaining ones are represented by the three highest prime numbers lower than 20, which are 19, 17 and 13.

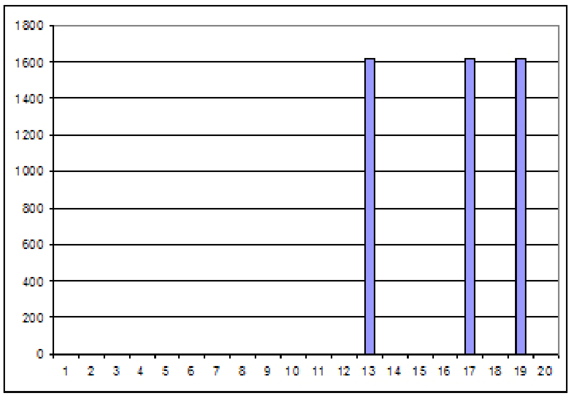

Figure 8: Classes Still Alive after 20000 Tics

While in nature a cicada with a life cycle of 19 years doesn't exist, probably because it wouldn't be possible for such insect to live so long, in the simulated world that is considered feasible. If the classes were just up to 18, the real situation would have been exactly reproduced, that is one in which just two species emerge and survive: the *magicicada tredecim* and the *magicicada septendecim*. In this experiment, then, it can be observed that the populations of *magicicadas* confirm the "myth" of being biological prime number generator.

5.6 DataBase Security and Social Relationships

In this paragraph an operative framework for the creation of a model of a generic database is examined (Remondino, 2004), to simulate the security rules applied to it, and to verify the various effects they have on efficiency, time and data corruption, by modifying some core parameters. A database is always a very complex object, managed by many different rules, hence the idea of simulating the security environment using a Multi Agent Based Model. Besides, the need of granting a certain security level for data access often compromises the efficiency of data retrieval, and thus the optimal balance between the two is a difficult task to accomplish. The creation of an Agent Based Simulation, which models a database security environment, can give answers to what-if scenarios, as rules and parameters are changed, without interfering with the security policy in the real environment.

A Data Base Management System (DBMS) is defined as a software package, designed to store and manage databases, which are very large, integrated collections of data. A DBMS allows to reach the following objectives:

- Data independence and efficient access
- Reduced application development time
- Data integrity and security
- Uniform data administration

• Concurrent access, recovery from crashes

It is then obvious that one of the fundamental goals of a DBMS is to reach a security level which could prevent users with no specific grants to read data. It's also very important to reach a satisfying level for data integrity, by preventing unauthorized users to modify them. On the other side, it is necessary to reach (an) high efficiency of data retrieval, when the users have the required rights. A security policy applied to a database must specify who is authorized to do what, and a security mechanism allows to enforce a chosen security policy. There are two main mechanisms at the DBMS level: Discretionary Access Control (DAC) and Mandatory Access Control (MAC). The former is based on the concept of access rights or privileges for objects (i.e. tables and views), and mechanisms for giving and revoking users privileges; in this model, the creator of a table or a view automatically gets all privileges on it. The DMBS keeps track of who subsequently gains and loses privileges, and ensures that only requests from authorized users at the time the request is issued, are allowed. The fundamental command, in this paradigm, is GRANT:

GRANT privileges ON object TO users [WITH GRANT OPTION]

This way, the specified users get the privileges on the object belonging to the DB; usually, the privileges are the following ones:

• SELECT: Can read all columns (including those added later via ALTER TABLE command).
• INSERT(col-name): Can insert tuples with non-null or non-default values in this column. Similarly, UPDATE.
• INSERT means same right with respect to all columns.
• DELETE: Can delete tuples.
• REFERENCES (col-name): Can define foreign keys (in other tables) that refer to this column.

If a user has a privilege with the GRANT OPTION, she/he can pass it on to other users, in turn with or without passing also the GRANT OPTION. Privileges can of course be lost, through the REVOKE command; if a user looses her/his privileges on an object, also the ones who had them from him will lose them. A user can receive the same privileges from different subjects and, in this case, she/he would loose them only if all these users loose those privileges on the object.

5.7 An Agent Based Model of DB Security

A set of agents will be modelled, which are the users of a database, organized into a hierarchy; in general, a community of agents which can access data according to specific rules. When the single agent needs a datum, she/he first looks for it and, if she/he can't access it directly, she/he asks other agents, who have the specific permission on it. Hierarchy and proximity relations are defined among the agents: there are n levels, organized into a pyramid. Level 1 is the uppermost, while Level n is the lowest. Besides, on the same level, proximity relations can be defined: a list could also exist, called Project Colleagues, containing the IDs of the agents working on the same project, so that if one of those creates a table or an object, the other agents will automatically have the access to them. Data inside the database are modelled as very simple objects, which have: a unique number, so that they can be called and retrieved; an identifier, signalling if the datum is corrupt or fine; the ID of their creator. Besides, there is a variable, associated with the datum, which signals who accessed it for the last time; this is used to keep track of who damaged it, if the datum is not fine anymore. Each user agent has its own unique ID, representing the name of the subject; a number, identifying the level to which it belongs and a list of the privileges on data; each element of the list is an array whit the following elements:

1) datum number (code)
2) read privilege (yes/no)
3) ID of the subject that granted the read privilege
4) write privilege (yes/no)
5) ID of the subject that granted the write privilege
6) delete privilege (yes/no)
7) ID of the subject that granted the delete privilege

Besides, each agent has another list, containing the privileges it granted to others. Again, each element of the list is an array, with these elements:

1) datum number (code)
2) IDs of the subjects to whom read permission has been granted
3) IDs of the subjects to whom write permission has been granted
4) IDs of the subjects to whom delete permission has been granted

Obviously, an agent can't grant a privilege on a datum if it does not have it in the first place. When a subject creates an object, an array is automatically inserted in it list, with the new datum number (code) and all the privileges on it. Besides, the Colleagues List described above could be implemented and in this case, when an object is created, all the subjects in this list will automatically have the privileges on it. Each user has also an unreliability index, which is increased each time she/he damages data, after a write operation. When an agent must complete an operation on a certain datum, she/he tries to access it directly, by looking in its list if she/he has the needed privileges on it. If she/he has the privileges, she/he access the datum in a single time unit, t.

If it has not got the needed privileges, she/he asks the datum who its creator is; this requires a time equal to that of retrieving the datum directly, that is t; as a reply, the creator ID is returned. According to the adopted security policy, which in the simulation can be changed by the user, the agent will ask for the grant directly to the creator or, in the most inflexible case, she/he will have to move up in the hierarchy, asking for the grant to an user at the lever which is immediately above, and so on, till when she/he meets one that has the required privileges. In the worst case she/he will need to go all the way up to the creator; obviously, each request will take some time, which can be considered equal to t/2. When the user meets an agent with the required privileges, she/he asks him to pass them to himself, consuming again a time t/2. According to the inflexibility of the security policy, specified before the simulation starts, and according to the unreliability index of the subject, the privileges will or won't be granted.

If they are granted, the user will be able to access the datum, in a time t; again, according to the security policy, the privileges can be kept by the user or can be immediately revoked after the operation has been completed. If the privileges are not granted, the user which owns them will access the datum on behalf of the requesting agent in a time 2t. Of course, if the same user has to access the datum again, she/he will have to pass again through all these steps.

When an agent has the write privilege on a datum, there is a probability function which determines the possibility that this operation compromises its integrity. During the write operation, the user could modify the flag variable of the datum from 1 to 0.

The next time this datum will be needed, the problem will emerge: this will increase the index of system unreliability and will waste a time 2t, to restore the datum. The user who damaged it, whose ID is stored in the Last Access variable of the datum, will have its personal unreliability index increased and, according to the security policy adopted, will loose or not the privileges on that datum. When a user looses her/his privileges on a certain datum, also the agents who had the privileges from him will loose them. It could be possible that an agent had received the same privileges from more than one user: in this case, she/he will keep the privileges, unless all the granting agents loose them. The need of the users to access data is controlled by random functions, and so it the probability function for data corruption.

This probability increases with the growth of the delta between the level of the creator and that of the user accessing the data. During the execution of the simulation, two real-time graphs will be created: one will represent the average time for data retrieval; the other one will represent the general unreliability index of the system, derived from the average of corrupted data. By varying the security policy, through the initial parameters, it will be possible to compare different situations, after the same number of simulation steps and with the same random seed. The security policy affects the probability for an agent to grant the privileges to another user, on certain data. A probability equal to 0 means that no agent will receive the privileges, so that only the owners can access the data they created. In this case, the general unreliability index will be very low, but the time for data retrieval will reasonably be very high.

A probability equal to 1 means that the privileges are always granted, no matter who asks for them: of course this will bring to an opposite situation. The intermediate cases, i.e. a probability between 0 and 1, are the most interesting and difficult to predict, but also the most useful to model real situations. Also the creation of new data is managed in a random way, and at the beginning of the simulation some steps will be dedicated to this activity. A number of data is modelled, with an inverse proportion in respect of the level or, in alternative, the exact situation that is observed in the organization we want to model can be put in the simulation. The last case that needs to be considered is the request of privileges on certain data by an user who is at an upper level than their creator; there could be an automatic grant or, more realistically, the request could be addressed directly to the creator, without needing to move down in the hierarchy, but using the rules defined in the security policy, that consider the personal unreliability of the requesting agent.

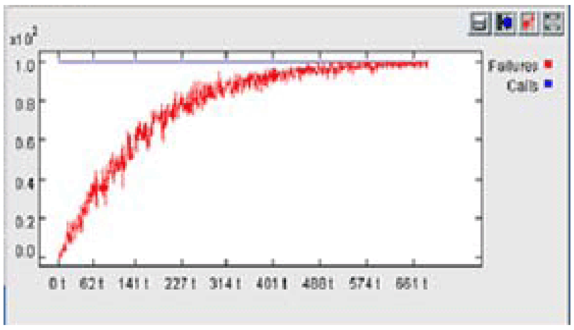

Figure 9: DB with no Security Rules

In Figure 9 the simplest case is shown; no security rules exist, so data are soon corrupted at 100%. This is just to show that the model can be realistic and thus can be used for "what-if" analysis and parameters correlation check.

This simple example shows how reactive agents (here endowed with basic capabilities) can be used to create models of very complex realities, such as a database.

A very different use of software agents is done with deliberative agents, i.e. those endowed with the ability of reasoning about actions and consequences. In the next paragraph and example is shown of a theoretical model employing BDI agents, a subset of deliberative ones.

6. BDI AGENTS IN ENTERPRISE MODELING

According to (Kinny et al., 1996), the BDI paradigm provides a strong notion of agency; agents are viewed as having certain mental attitudes, Beliefs, Desires and Intentions, which represent, respectively, their informational, motivational and deliberative states. In the BDI architecture an agent can be completely specified by the events that it can perceive, the actions it may perform, the beliefs it may hold, the goals it may adopt, and the plans that give rise to its intentions.

In the following the enterprise is considered as a BDI meta-agent (Remondino, 2005), that's an agent grouping other ones, constituted by the functional areas. In this way the newest contribution of this approach is stressed: rather than a component based or object based approach to the architecture of the enterprise the different levels are modelled as agents, not real agents, but agents which are real only as long as all the ordinary (human or software) agents believe they are. The section is organized as a description of the two following schemes.

6.1 Macro Level

The whole enterprise can be considered as a meta-agent, grouping a series of other agents, which will be discussed later, sharing some common goals, desires and beliefs. The environment in which the enterprise operates is shared with other subjects, mainly competitors, customers and supplier, which have a direct influence on the crude facts (f), also connected with the enterprise itself and representing the repeated activities.

These other subjects are also agents, but it is not necessary to give them a real BDI structure, since that is not directly visible from the modelled enterprise; though, the enterprise can have a representation (through its own beliefs) of what is supposed to be the structure of the outer agents. For example, it will know for sure that the competitors will try to overcome it in terms of market share, while the customers need to be satisfied and will try to get the best possible deals in terms of goods/services for the least money. Other external agents can exist: for example a normative system, created "a priori".

According to (Boella, 2003) this is defined as a social constructed entity, to which the other agents in the world attribute the status of agent; in particular the agents assign regulative norms to it. Since this is not the main focus of the model, the normative agent is simplified from the one proposed in (Boella, 2003) and acts just as a regulator for the actions of the agents involved.
There are not constitutive norms, and hence no beliefs are present in the normative agent. This agent is useful to avoid improper actions, like treacherous competition, prices out of the logical range and so forth. Thus the normative system will just feature goals (rules), a sub-set of which is derived representing what is considered a violation (V). By observing the facts and considering its beliefs, the enterprise has its own representtation of the state of the world (s). This is schematically represented in Figure 10.

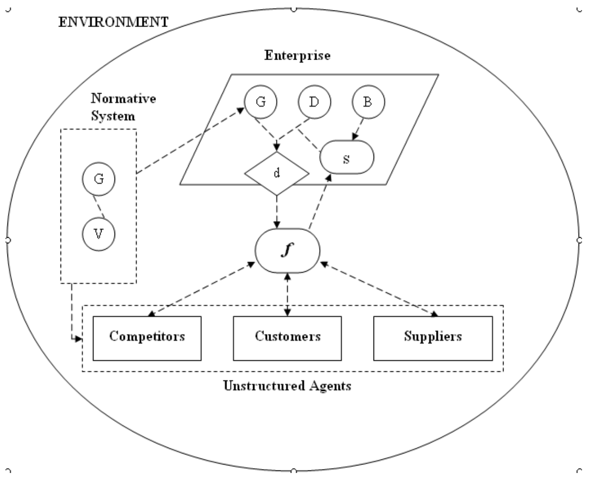

Figure 10: Enterprise as BDI Agents - Macro

6.2 Enterprise Level

The enterprise meta-agent is divided into several sub-agents, each of which has its own beliefs, desires and goals. These are strictly linked to the general ones, belonging to the whole enterprise (B,D, G), in the sense that the attributes of the single functional area contribute to defining the general ones, but these, in turn, influence the ones of the various areas. This construction is realistic, since the areas constituting the enterprise must share some knowledge and objectives with it in order to make it work, but at the same time there could be some attributes which are not shared by all the subjects or that could not be of any interest for some of them, or can't be pursued by the enterprise as a whole. Besides, in the model an indirect link among the areas is provided by the super-agent, which is the enterprise itself; it does not directly affect the behaviour of the sub-agents but at the same time both influences and is influenced by them. For example, the enterprise seen as a BDI agent will pursue the highest possible profit, the best efficiency, the highest customer satisfaction and so on; some of these goals – or part of each - can flow to all the sub-agents, while others should be shared just by some of them. Besides, since the model should be realistic - depicting also the human factor – it must be assumed that the functional areas share some of the beliefs of the enterprise as a whole, but not all of these are surely correct when considered at a macro level. For instance, the chief of a production unit could believe that building her/his product is strategic for the enterprise, while at a macro level that's not so important and hence that sector can be sacrificed to drive more resources to another unit or area. It's thus very important to introduce two levels of views, one corresponding to the whole enterprise and one to the single areas.

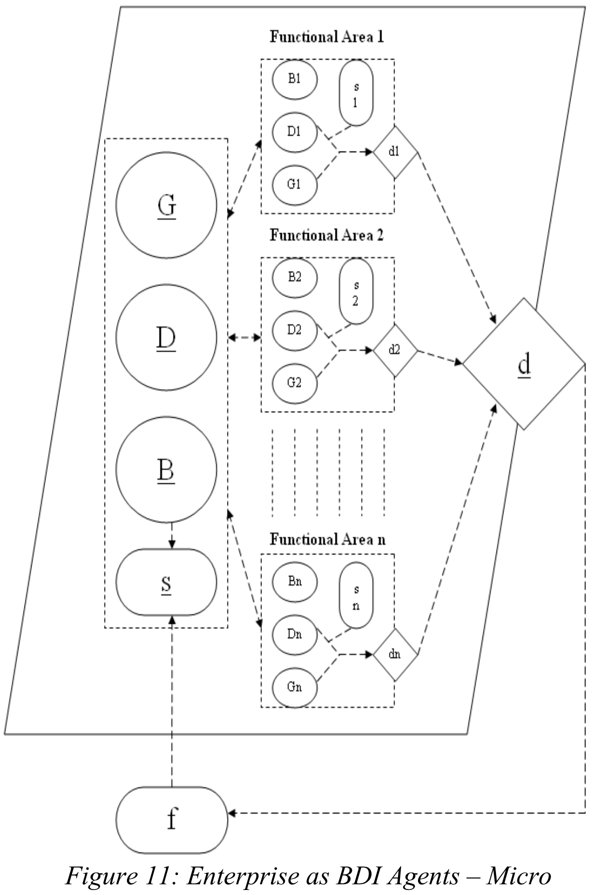

Figure 11: Enterprise as BDI Agents – Micro

Each functional area can then perform actions (d1, d2,…,dn) according to its own goals and desires. At a macro level, the set of these actions creates those (d) performed by the enterprise in the environment. These macro-actions are the only ones which control some of the facts (f), while the ones performed by the functional areas are not directly affecting the external environment.

It is important to determine how the various actions performed by every area affect the global actions and performance of the enterprise. The mechanism, in the real world, is complex and certainly not deterministic. The beliefs about the state of the world are also different if considered at the macro level or at the micro level; the enterprise derives its own beliefs about the state of the world (s), from the brute facts it observes in the external environment (f) and from its own general beliefs (B). On the other hand, the beliefs about the state of the world belonging to the functional areas (s1, s2,…, sn) are derived from what they observe in their own environment, which is the enterprise itself, and/or from the external facts (f) filtered by the enterprise (i.e. by s) and always with the influence of their own beliefs (B1, B2,…, Bn). This is a rather closed and unrealistic vision: in the real world, the people belonging to a functional area have their personal beliefs about the state of things from the outside, with no mediation by the enterprise. Though, since the focus is on the structure of the enterprise itself, it would be useless and time consuming to model different – and more complex - relations. In this model, the single beings are not relevant and exist just as members of the functional area, that's in turn formally and politically linked to the whole enterprise, being a part of it and loosing its function (and reason of existence) outside of it.

7. CONCLUSION

The "agent" concept is very blurry and often abused; two quite different agent paradigms were examined in the article and both of them can be practically used to create simulations and models of complex systems. On one side, the reactive paradigm is easier to implement and probably more flexible: the single agents are very simple and do not feature a simulated mind. They can only receive stimuli from the environment in which they act and respond to them according to their inner program. Even if they may look so simple, the aggregation of many can bring to complex situations and unpredictable results. In the real world there are many examples that can be modelled and simulated by using reactive agents; we can think of insects colonies, but also to markets, enterprises and everything which needs social contacts among the entities involved. Of course, sometimes simplifications can be necessary, but the important properties of the system must be preserved. The output of the model can be quantitative or qualitative, but is often about a social trend, that's a way in which the agents aggregate and

behave when linked to others by some sort of situation.
Very different from the reactive agents are the deliberative ones; while more complicated and endowed with a simulated mind, able to "reason" on the actions to take basing on the situations in which they operate, they are much more difficult to implement practically. An example is shown, in which an enterprise – which is of course a complex social system – is theoretically modelled using BDI agents, subset of the deliberative ones. Although very comprehensive and complete, this scheme becomes difficult to implement on a computer and computationally heavy. Anyway, some deliberative agents can have reactive parts, simplifying their implementation and speeding their execution on a machine.

8. FUTURE WORK

An important issue that remains open in the ABS field is that of *model validation*, which is "substantiation that a computerized model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model" (Schlesinger et al. 1979), while m*odel verification* is often defined as "ensuring that the computer program of the computerized model and its implementation are correct" and *model accreditation* determines if a model satisfies specified criteria according to a specified process. The task of model validation in ABS is particularly hard for the aggregate dynamics described in this article.

The innovative idea, which will be described in a future article, is to use data-mining to tune the initial parameters of the model, and to study the sensitivity of the results. This is expected to supply satisfactory ranges to the agent based models.

REFERENCES

Arthur B. (1994). Inductive Reasoning and Bounded Rationality, American Economic Review (Papers and Proceedings) pp.84,406-411

Boella G. & Van Der Torre L. (2003). Normative Systems as Agents, working paper

Brooks, R. A. (1990). Elephants Don't Play Chess, Designing Autonomous Agents, Pattie Maes, ed. Cambridge, MA: MIT Press
Bensaid N., & Mathieu P. (1997), A hybrid architecture for hierarchical agents

Bonabeau E., (2002). Agent-based modeling: Methods and techniques for simulating human systems, PNAS 99 Suppl. 3 (pp. 7280-7287)

Challet D. & Zhang Y.C. (1997). Emergence of cooperation and organization in an evolutionary game, Physica A246

Drogoul A., Vanbergue D., Meurisse T. (2002). Multi-Agent Based Simulation: Where are the Agents?, Proceedings of MABS'02 (Multi-Agent Based Simulation, Bologna, Italy, July 2002), LNCS, Springer-Verlag

Franklin, S. (1995). Artificial Minds, Cambridge, MA: MIT Press

Franklin, S., & Graesser, A. (1997). Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents, Proceedings of the Agent Theories, Architectures, and Languages Workshop, Berlin (pp. 193-206). Springer Verlag

Ginsberg M. (1989). Universal planning: An (almost) universally bad idea. AI Magazine 10(4) pp. 40-44

Henize, J. (1984), Critical issues in evaluating socio-economic models. Tuncer I. Oren, Bernard P. Zeigler, and Maurice S. Elzas, editors, Simulation and Model-Based Methodologies: An Integrative View, NATO Advanced Science Institutes Series, Series F: Computer and Systems Science, vol. 10, (pp. 557-590). Springer, Berlin, Heidelberg, New York, Tokyo.

Jennings, N. R. (1996). Software agents. IEEE Review (pp. 17-20)

Kinny D., et al. (1996). Modelling and Design of Multi-Agent Systems, Intelligent Agents III: Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages (ATAL-96)

Langton, C. (1989). Artificial Life, Redwood City, CA: Addison-Wesley

Mataric, M. (1995). Issues and approaches in the design of collective autonomous agents. Robotics and Autonomous Systems 16 (pp.321-331)

Ostrom T. (1988). Computer simulation: The third symbol system. Journal of Experimental Social Psychology, 24 (pp. 381-392)

Pavard B. & Dugdale J. (2000). The contribution of complexity theory to the study of socio-technical systems, Proceedings of Third International Conference on Complex Systems. New Hampshire.

Remondino, M. (2003). Agent Based Process Simulation and Metaphors Based Approach for Enterprise and Social Modeling, ABS 4 Proceedings (pp.93-97). SCS Europ. Publish. House – ISBN 3-936-150-25-7

Remondino M. & Cappellini A. (2004). Minority Game with Communication: an Agent Based Model, Simulation in Industry 2004, SCS Europ. Publish. House – ISBN 1-56555-286-5

Troitzsch, K.G. (1996). Chaotic behaviour in social systems. In Rainer Hegselmann and Heinz-Otto Peitgen, editors, Modelle sozialer Dynamiken. Ordnung, Chaos und Komplexitat, (pp. 162-186). Holder-Pichler-Tempsky, Wien

Williams, Kathy S., Smith, Kimberly G. and Stephen, Frederick M. 1993. " Emergence of 13- Yr Periodical Cicadas (Cicadidae: Magicicada): Phenology, Mortality, and Predator Satiation." Ecology Vol.74. No.4 1143-1152

Woolridge, M., & Jennings, N. (1995). Intelligent agents: Theory and practice. Knowledge Engineering Review 10(2) pp.115-152

Yoshimura, Jin. 1996. " The Evolutionary Origins of Periodical Cicadas During Ice Ages" The American Naturalist Vol. 149, No. 1 112-124

BIOGRAPHY:

Dr. Marco Remondino was awarded a master degree (cum Laude et Menzione) in Economics at the University of Turin in 2001. In 2005, he was awarded a PhD in Computer Science. He is now a temporary researcher and holds a grant from the ISI Foundation, for the Lagrange Project on Complex Systems, at the University of Turin, Italy. His current research interests include social simulation and modelling, agent based and process simulation and enterprise modelling languages. For the Lagrange Project he is collaborating with AEM (the Electrical Company of Turin) for developing models of the electric market and diffusion of technical innovations.