EMERGENCE OF SELF ORGANIZATION AND SEARCH FOR OPTIMAL ENTERPRISE STRUCTURE: AI EVOLUTIONARY METHODS APPLIED TO AGENT BASED PROCESS SIMULATION

Marco Remondino
Department of Computer Science
University of Turin
C.so Svizzera 185
10149 Turin, Italy
E-mail: remond@di.unito.it

KEYWORDS

simulation, model, intelligent agent, genetic algorithm, classifier system, complex behaviour, process

ABSTRACT

Enterprise simulation allows what-if analysis and helps in business process re-engineering. There are mainly two approaches to simulation: process based, which is strictly deterministic and generally used to model well known parts of enterprises or mechanical/electronic systems and agent based, which allows to study the emergence of aggregate behaviour, through the creation of models, known as artificial societies. In order to simulate enterprises where the environment and human factor are relevant, a hybrid formalism is proposed: Agent Based Process Simulation. Colonies of intelligent agents, modelled using evolutionary methods derived from the AI field, are put side by side with the representation of processes, modelled as symbolic and deterministic agents built using paradigms derived from Propositional and Modal Logic. The main goal of this work is to study how this approach can allow the emergence of aggregate behaviour and, by using some performance parameters, can help to find the optimal organization for an enterprise. In particular, agents built using Genetic Algorithms and Classifier Systems can evolve to find local maximum of functions representing situations whose rules are not entirely known a priori, such as the ones behind the optimal organization of an enterprise.

INTRODUCTION

In (Ostrom 1988), simulation is described as a third way to represent social models, being a powerful alternative to other two symbol systems: the verbal argumentation and the mathematical one. The former, which uses natural language, is a non computable way of modelling though a highly descriptive one; in the latter, while everything can be done with equations, the complexity of differential systems rises exponentially as the complexity of behaviour grows, so that describing complex individual behaviour with equations often becomes an intractable task. Simulation has some advantages over the other two: it can easily be run on a computer, through a program or a particular tool; besides it has a highly descriptive power, since it is usually built using a high level computer language, and, with few efforts, can even represent non-linear relationships, which are tough problems for the mathematical approach. According to (Gilbert, Terna 1999):

"The logic of developing models using computer simulation is not very different from the logic used for the more familiar statistical models. In either case, there is some phenomenon that the researchers want to understand better, that is the target, and so a model is built, through a theoretically motivated process of abstraction. The model can be a set of mathematical equations, a statistical equation, such as a regression equation, or a computer program. The behaviour of the model is then observed, and compared with observations of the real world; this is used as evidence in favour of the validity of the model or its rejection"

Computer programs can be used to model either quantitative theories or qualitative ones; simulation has been successfully applied to many fields, and in particular to social sciences, where it allows to verify theories and create virtual societies. In particular, the simulation of an enterprise can give very good results, regarding case studies, what-if analysis and business process re-engineering. It is possible to identify two different approaches to computer simulation, both of which lead to the creation of a computational model of a social or complex system, starting from a very different point: Process Simulation and Agent Based Simulation. Both of them can be used to model enterprises or firms, but with some fundamental differences, which will be discussed in detail. Agent Based Modeling 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

Proceedings 15th European Simulation Symposium
Alexander Verbraeck, Vlatka Hlupic (Eds.)
(c) SCS European Council / SCS Europe BVBA, 2003
ISBN 3-936150-28-1 (book) / 3-936150-29-X (CD)

hard to describe in detail. For this reason, process simulation is not the ideal tool to model these complex environments; besides, there are agent based formalisms which allow to study the emergency of social behaviour with the creation and study of models, known as artificial societies. Thanks to the ever increasing computational power, it's been possible to use such models to create software, based on intelligent agents, which aggregate behaviour is complex and difficult to predict, and can be used in open and distributed systems. A software agent can be described as a flexible system, capable of dynamic, autonomous actions, in order to meet its design objectives, that is situated in some environment. The main features for a software agent are: situatedness, that is ability to perform actions according to a particular input received from outside, which can, in turn, change the environment itself; autonomy in performing actions, without intervention of humans; flexibility and adaptability. Some particular agents can also be proactive, which means they are goal-directed, and social, in the way they can interact with other artificial agents, robots, and humans. Such an intelligent agent can be referred to as a Belief-Desire-Intention (BDI) one. There are many agent based paradigms that can be applied to computer simulation:

- *Symbolic*: highly structured agents, described through expressions of Propositional and Modal Logic. This is perfect when there is a single agent, which must interact with the environment, but it's not versatile when used to simulate big communities
- *Sub-symbolic*: simple agents, which can be described through metaphors. A multi-agent context of this kind allows the emergency of complex behaviour and self-organization. Intelligent behaviour is a product of the interaction among agents and environment, and of the interaction among many simple behaviours. It can be really hard to describe the real world under every aspect: some fundamental macro-actions can thus be defined on single agents, which allow cooperation with the environment and with other agents. The concept of *Multi Agent System* for social simulations is thus introduced: the single agents have a very simple structure. Only few details and actions are described for the entities: the behaviour of the whole system is a consequence of those of the single agents, but it's not necessarily the sum of them. This can bring to unpredictable results, when the simulated system is studied.
- *Hybrid Architectures*: at the lower levels, we find reactive agents, like the ones described above, while at the upper levels there are more complex and structured agents. In this way, we can combine reactive capabilities with planning.

The approach proposed in this work can be considered as a hybrid architecture, since it uses symbolic agents as process based blocks, and sub-symbolic agents to model those parts of an enterprise which are not fully known, part of the environment, or even the human beings involved in the organization.

SIMULATION: TWO DIFFERENT APPROACHES

Both process simulation and agent based simulation are powerful approaches for creating models of enterprises and complex systems, but they also have some flaws. In order to overcome the limits of both the simulation approaches, the possibility of a hybrid methodology is studied in (Remondino, 2003). In the present work I'll concentrate the discussion on enterprise simulation and I'll discuss how intelligent agents, based on AI paradigms, in particular genetic algorithms and classifier systems, can show an emergent aggregate behaviour when put side by side with formal and deterministic processes. While deeply describing both the approaches is beyond the purpose of this paper, I'll analyze the main differences among them, which will lead to the hybrid formalism that I'm studying. Usually, process simulation is used to model a very well structured and known situation, in order to perform a what-if analysis: it's used to create models of parts of enterprises or mechanical/electronic systems. Its greatest advantage is that it starts from a basic scheme, often derived from existent documents, through which it becomes very easy to bring a real situation into a process simulator: usually, a model to be used for process simulation looks like a flow chart, in which a token passes from one box to another one, in a deterministic way, on the basis of the given rules. This kind of approach is widely spread and allows to deeply analyze a part of a whole, studying the expected behaviour of a system, when some change is operated. This is why process simulation is a great support to decisions; the simulator can answer to many questions and what-if problems, that would require big efforts in the real environment; for example, a part of a manufacturing plant can be simulated, by dividing it into its main processes, and then it will be possible to check what would happen on the final output if some change occurs. According to (Helsgaun, 2000), the process based approach, when building deterministic simulations, is alternative to the *event based* and the *activity based* ones. In the former the model consists of a collection of events and each event models a state change and is responsible for scheduling other events that depend on that event. Each event has associated an event time and some actions to be executed when the event occurs. In the activity based approach the model consists of a collection of activities: each activity models some time-consuming action performed by an entity. Each activity has associated a starting condition, some actions to be executed when the activity starts, the duration of the activity, and some actions to be executed when the activity finishes. In the process based approach the model consists of a collection of processes. Each process models the life cycle of an entity and is a sequence of logically related activities ordered in time. Since processes resemble objects in the real world, process based simulation is often easy to understand; implementation, however, is not always easy and execution efficiency may be poor if the implementation is not done properly.

Unfortunately there isn't a universal modelling language for process simulation and this often requires deep translations for the models to be ported from one tool to another. Another disadvantage is that, in order to use this approach to simulate a process, this must be very well known; if a part of the process is uncertain, then it's impossible to validate a simulation as a model of the real world to be represented. Besides, this method is quite static, meaning that the relations between the various parts involved in the model must be described in deep and there is no possibility of emergent behaviour and self-organization.

When the system to be simulated has a complex aggregate behaviour, not easy to describe just studying and modelling the single entities, agent based simulation is the only usable approach. In complex systems the sum of the parts is often not enough to describe the whole, and usually from the interaction of many simple entities a complex behaviour emerges. So, if we want to model an enterprise in which also the human factor is present, or we want to consider also the influence of the environment, it will be impossible to do that with a process based approach, thus leaving agent based simulation as the only feasible method. While in process simulation the stress is on the function of the single parts, that are deeply modelled as resembling the reality, in agent based simulation the most important side is interaction among entities, which creates the aggregate behaviour. The single agents can even be very simple, with few rules and directives. For example, an artificial stock market can be simulated by creating some different types of intelligent agents, which follow inner rules; some of them can simply act randomly, while others will "study" the trend before acting. Some of them, on the contrary, could use advanced techniques, such as *stop loss*. By observing the general trend of an artificial stock market created with these rules, one can be amazed, by seeing that it resembles in many ways a real one. On the other side, agents can be modelled with inner reasoning and learning capabilities, for example using *neural networks*, *genetic algorithms* or *classifier systems*, which create an evolutionary environment. Each agent has the capacity to reason on the global effects of local actions, or even to create its own forecasts on the actions that will be performed by other agents. The agents built using this approach can decide on which action to perform, according to the stimuli coming from the environment, and not only according to their internal rules. According to (Bonabeau, 2002), Agent Based Modeling has three main benefits, over other approaches: it captures emergent phenomena; it provides a natural description of a system; and it is flexible.

AGENT BASED PROCESS SIMULATION

In (Bonabeau, 2002), we read that agent based paradigm can be used successfully to model different situations, like flows, markets, organizations, social diffusion of phenomena; on the other hand, process based approach has proved to be very useful for detailed, but static and deterministic, machinery and firm simulations. There are many intermediate situations, though, in which neither process simulation nor agent based approach can be applied with good results. Besides, the works of evolutionary economics, which use agents to represent industrial processes, also have a vision which is opposite to the static equilibrium, but are not meant to describe an enterprise or a machinery in detail. In (Remondino 2003), some examples can be found, about situations that can be described neither with pure process based nor pure agent based approaches, but could be modelled using a hybrid derived approach. Here I will only present a general framework for Agent Based Process simulations, shown in Figure 1. The market is the environment for the enterprise; there are buyers, i.e. the customers, and sellers, i.e. the suppliers. In a traditional process based simulation, these actors would be left out of the model, and the stress would be put on the way the single enterprise works. On the contrary, by using a pure agent based approach, we could model all these entities, but we couldn't model the real structure of enterprises in detail.

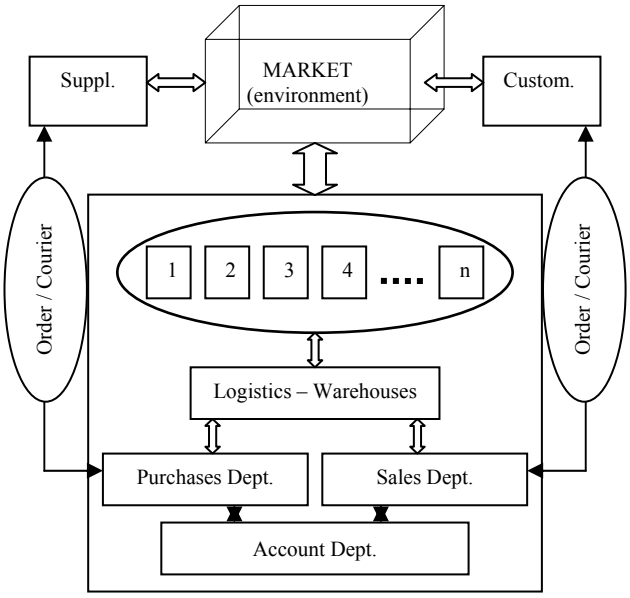

Figure 1: A General Framework for ABPS

Combining the two approaches, we can have a detailed model of the whole enterprise, with its production units, sales, purchases and account departments, logistics, warehouses and so on, modelled with a process based approach, and the environment, customers and sellers behaviour simulated using agent based technology. Besides, also the workers of the enterprise, i.e. persons in charge of machineries, department directors, disposers and so on. For example, sales and purchased departments could be modelled as shown in Fig. 2 and 3, while both customers and suppliers could be simple agents, acting on the basis of a probability function, based on real

data coming from market studies or simply randomly, if we want to see how the modelled enterprise reacts to whatever situation, even not realistic, coming from outside.

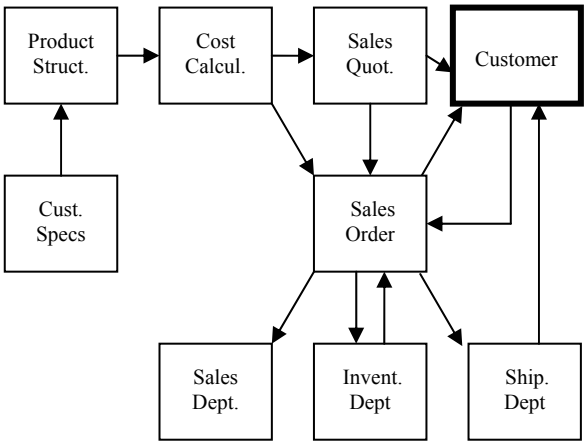

Figure 2: Typical Process Based Model of Sales Department

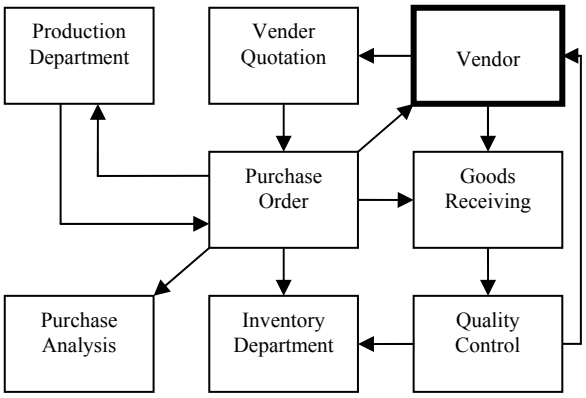

Figure 3:Typical Process Based Model of Purch Department

The blocks constituting the departments are processes, modelled in a deterministic way; their structure is made of elementary building blocks that use formalisms derived from Propositional or Modal Logic.

SYMBOLIC AGENTS REPRESENTING PROCESSES

Usually, since processes can be modelled as deterministic flows, my proposal is to use both Propositional and Modal Logic to describe their structure. In (McCartney 2001) we read that:

"The basis for most current systems of formal logic is Propositional Logic, also known as Propositional Calculus or PC. PC describes truth-based rules using the fundamental ideas of not and or, and derivations of the concepts of and, implication, and strong implication. A common extension to PC is predicate logic. Predicate logic includes variables as well as non-truth-based validity; or mapping variables into values other than the Boolean true or false. Another non-truth based logic is modal logic, which is based on PC and introduces the concepts of necessity and possibility. Modal logic is closely related to PC and predicate logic, but is able to describe states that would be indescribable in either of these languages"

In order to model a deterministic process, the Propositional Logic could be enough, since it allows to create truth tables of the single sub-processes. Modal Logic allows having a more versatile environment, allowing to determine if a proposition is true for sure, false for sure or sometimes true and sometimes false (i.e. it's possible). In my framework I will only suppose the use Propositional Logic, to model simple processes: this allows to describe a process, create a model of it and simplify the transition to programming code required to port it into a working simulation. A sub-block of a process produces output_1 if the logic formula is True, or output_2 if it's False; one of the two outputs can be simply Void. In this way, a part of a whole process can be like exemplified in Figure 4.

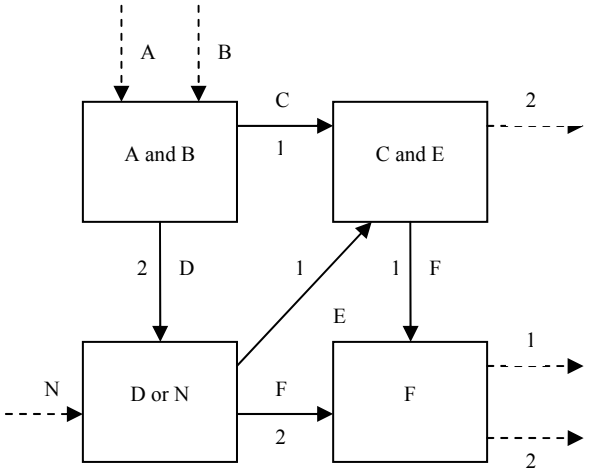

Figure 4: Propositional Logic Based Sub-block

Passing from this kind of representation to a programming language is a very easy step, since all the single boxes can be represented with if-then functions. In this way a very complex deterministic process can be modelled starting from very simple building blocks. Modal Logic can even add concepts of probability and necessity, so that a particular output going out from a basic building block can always occur or it's possible that it occurs. In this case a probability function can be given, representing the views on the possible modal worlds, to specify how often an output can be produced, given the initial rule.

This approach allows to model machineries and the production units of an enterprise; the most difficult part to simulate, but probably also the most interesting for which regards the emergence of aggregate behaviour and self organization, is the human factor, e.g. the workers involved in the structure of the enterprise. Finding the optimal organizational structure of an enterprise is a very difficult task to accomplish, though a critical subject. There exist some tools, derived from AI studies, that allow embedding some sort of reasoning and learning capabilities into software agents. In particular, I will discuss about Genetic Algorithms and Classifier Systems.

GENETIC ALGORITHMS AND CLASSIFIER SYSTEMS

In some situations, effective results can be obtained just by building simple agents, whose behaviour is randomly determined or is built by applying fixed pre defined reaction rules; this could be the case, for instance, of Heatbugs, one of our canonical Swarm demonstrations (www.swarm.org):

"It's an example of how simple agents acting only on local information can produce complex global behaviour. As we read on Swarm main site, each agent in this model is a heatbug. The world has a spatial property, heat, which diffuses and evaporates over time. In this picture, green dots represent heatbugs, brighter red represents warmer spots of the world. Each heatbug puts out a small amount of heat, and also has a certain ideal temperature it wants to be. The system itself is a simple time stepped model: each time step, the heatbug looks moves to a nearby spot that will make it happier and then puts out a bit of heat. One heatbug by itself can't be warm enough, so over time they tend to cluster together for warmth"

This is a useful approach when we wish to simulate situations in which we give the rules of the environment and we want to observe some emerging aggregate behaviour arising from simple entities; of course, the way the agents will act tends to be deeply dependent on the choices made by the programmer. As an alternative we can choose to create agents with the ability to compute rules and strategies, and evolve according to the environment in which they act; in order to model them, we can use some methods derived from the studies on artificial intelligence, such as artificial neural networks and evolutionary algorithms. While the former is a collection of mathematical functions, trying to emulate nervous systems in the human brain in order to create learning through experience, the latter derives from observations of biological evolution. Genetic Algorithms derive directly from Darwin's theory of evolution, often explained as "survival of the fittest": individuals are modelled as strings of binary digits and are the encode for the solution to some problem. The first generation of individuals is often created randomly, and then some fitness rules are given (i.e. better solutions for a particular problem), in order to select the fittest entities. The selected ones will survive, while the others will be killed; during the next step, a crossover between some of the fittest entities occurs, thus creating new individuals, directly derived from the best ones of the previous generation. Again, the fitness check is operated, thus selecting the ones that give better solutions to the given problem, and so on. In order to insert a random variable in the genetic paradigm, that's something crucial in the real world, a probability of mutation is given; this means that from one generation to the next one, one or more bits of some strings can change randomly. This creates totally new individuals, thus not leaving us only with the direct derivatives of the very first generation. Genetic Algorithms have proven to be effective problem solvers, especially for multi-parameter function optimization, when a near optimum result is enough and the real optimum is not needed. This suggests that this kind of methodology is particularly suitable for problems which are too complex, dynamic or noisy to be treated with the analytical approach; on the contrary, it's not advisable to use Genetic Algorithms when the result to be found is the exact optimum of a function. The risk would be a convergence to some results due to the similarity of most the individuals, that would produce new ones that are identical to the older ones; this can be avoided with a proper mutation, that introduces in the entities something new, not directly derived from the crossover and fitness process. In this way, the convergence should mean that in the part of the solution space we are exploring there are no better strategies than the found one. It's crucial to choose the basic parameters, such as crossover rate and mutation probability, in order to achieve and keep track of optimal results and, at the same time, explore a wide range of possible solutions.
Classifier Systems derive directly from Genetic Algorithms, in the sense that they use strings of characters to encode rules for conditions and consequent actions to be performed. The system has a collection of agents, called classifiers, that through training evolve to work together and solve difficult, open-ended problems. They were introduced in (Holland 1976) and successfully applied, with some variations from the initial specifics, to many different situations. The goal is to map if-then rules to binary strings, and then use techniques derived from the studies about Genetic Algorithms to evolve them. Depending on the results obtained by performing the action corresponding to a given rule, this receives a reward that can increase its fitness. In this way, the rules which are not applicable to the context or not useful (i.e. produce bad results) tend to loose fitness and are eventually discarded, while the good ones live and merge, producing new sets of rules. In (Kim, 1993) we find the concept of Organizational-learning oriented Classifier System, extended to multi-agent environments with introducing the concepts of organizational learning. According to (Takadama 1999), in such environments agents should

cooperatively learn each other and solve a given problem. The system solves a given problem with multi-agents' organizational learning, where the problem cannot be solved simply by the sum of individual learning of each agent.

EVOLUTIONARY METHODS APPLIED TO ABPS

Agent Based Process Simulation is a way to model deterministic structures, made up of single processes, divided into Propositional Logic based building blocks, and having them interact with agents belonging to the sub-symbolic paradigms. This allows to simulate situations in which not only the deterministic structure, but also unpredictable situations could arise, caused by the environment or the human factor are important; we can think about many different situations, that couldn't be represented by a pure process based approach, and would result too difficult and inaccurate to be modelled just using self organizing agents. For example, agents could be part of the structure of an enterprise modelled with process based approach; they could be regarded as parts acting more like human beings than like machines. We may think of a generic enterprise, in which many sub-systems, i.e. units, can be described with a process based approach. The interaction between these basic subsystems, though, is usually really complex, and generally involves a human or non deterministic participation. This would be very difficult, or even impossible to represent with a process based model; but it would also be useless to use a pure agent based approach, since many parts could be only modelled with structured and Logic based processes. That's where we can use agent based connections between the process based sub-systems. These agents should be quite simple, but structured ones, able to act starting from stimuli coming from the environment (i.e. the output of a sub-system modelled with process based approach), and to produce an output, that will effect the way other sub-systems will work. In a simulation built in this way, we can see what happens if we change the way we manage the warehouses, if we use more experienced employers or, for example, if the workers are on a strike.

With the same approach, we can go down to a micro level, for example by inserting agents into models of the single machineries and business units. If we think of a single, but very complex machinery, not all the parts are strictly deterministic, in the sense that they can be affected by some unforeseen influence coming from the environment. By using a process based approach, it is possible to model the machinery quite deeply, but just in a deterministic, static situation, which is the optimal environment, in which nothing can change its way of working: we can simulate the variation of the output by varying the input, or by improving some part of the system. Or we can prove the resistance and endurance of the machinery in optimal conditions. Though, such a simulation, for its nature, wouldn't be able to consider any chaotic or unforeseeable action, coming from outside, that could compromise the machine operations (e.g. damages caused by moist, fire, and so on). A representation of a typical process based model of a machinery is given if Figure 5.

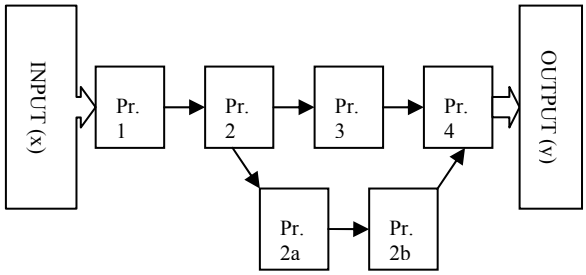

Figure 5: Example of Structure for Process Based Machinery

The model acts as a function which receives an input, that is the independent variable (x), processes it and produces a stochastic output, which is the dependent variable (y). Though very powerful and easy to validate, this is not always realistic. By considering certain parts of the machinery as very simple agents (Figure 5), it would be possible to create a more realistic model of the object, that will be able to react to the stimuli coming from the environment according to certain rules, written in the single agents, that would give the whole machinery a complex, and less deterministic behaviour, just as the one it would have in the real world. An example for this is given in Figure 6, derived from the previous one, with the insertion of some agents into the process based structure.

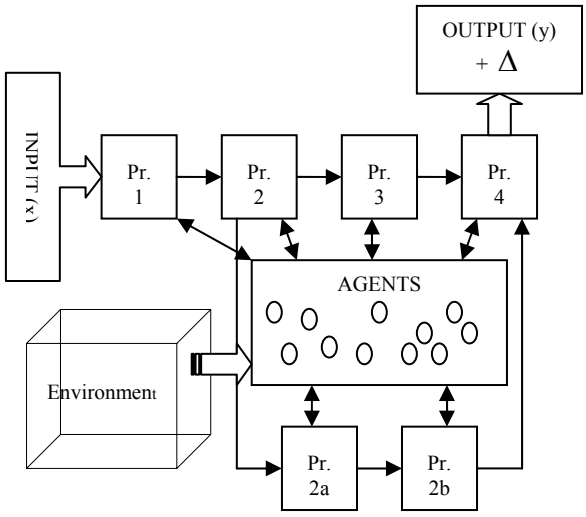

Figure 6: The Same Structure, with the Addition of Agents

Some agents are now put side by side with processes: while the main flow remains unchanged, now there is an influence coming from a hypothetical environment, just like in the real world. The agents can react to the stimuli coming from outside, which can be the rest of the enterprise, another machine, or even the person running the simulation. In this way, the model is not strictly static anymore, in the sense that given an input (x), same as before, the output is not a linear function of just the independent variable, but also of all the other ones that can be processed by the agents. The output is not the same (y) as before, but (y) plus a delta, which is caused by external, non deterministic influences on the agents. Of course these agents must act logically, on the basis of what could possibly happen in the real world; a totally random acting agent would be, obviously, useless. This kind of approach allows to build models which are more realistic and dynamic, as opposed to the static ones, where only the deterministic flows are simulated. The greatest difficulty, with this approach, is model validation, using data from real experiments, because the unexpected circumstances are difficult to reproduce more than once. The validation could then go top-down, in the sense that we observe certain data in the reality and then try to reproduce the same situation in the model, by using the agents in a piloted way. If the same results occur, it is possible to calculate a standard statistical error and validate the model for those particular situations. We can then extrapolate the results, and consider the model valid also for those situations that can't be controlled and created in the real world. If the agents involved in the simulation, at each level, are modelled using evolutionary methods such as Genetic Algorithms and Classifier Systems, we could achieve two main goals: the agents simulating the environment could evolve and self organize, thus creating a realistic situation.

Besides, if we use Classifier Systems to model the agents, we could find the optimal rules for the organization of a given enterprise, which is modelled through deterministic processes. In order to do that, we start from some basic parameters and performance indicator, that will serve as the rules to determine the fitness of the agents involved. The agents that will produce the higher local results will survive and merge, in order to create new generations derived from them; after many simulated steps, we should be able to find an optimal global organization of the simulated enterprise (or business unit, or even machinery), modelled using processes. For example, we can choose to maximize the local output of a production unit, given an input; the production unit will be modelled with a process based approach, using Propositional Logic formalisms for the building blocks. The human beings involved in each production unit are modelled as agents based on Classifier Systems, as represented in Figure 7. In this framework the single agents, which act as self evolving connections between the processes within the various business units, are modelled using Genetic Algorithms. By using some simple reference parameters, in particular the local output, that's the output produced by the single business units, it's possible to assign a fitness value to the agents. When the simulation starts, a population of random agents is created (random binary strings) for each unit. They produce certain effects operating on the process based parts of the production unit through their actions: we could say that they operate the units in a random way.

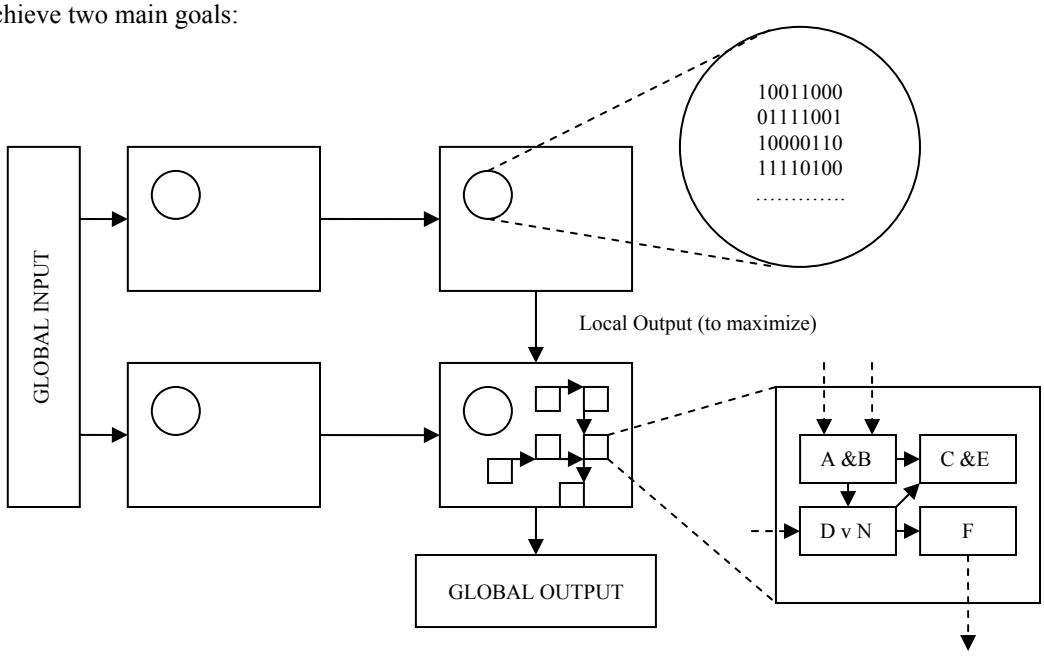

Figure 7: AI Evolutionary Methods Applied to ABPS

The agents that produce the best local output, given a certain input, are saved and used for the crossover among them. The resulting agents now compete against the previous local optimum, and again only the ones with better results are kept and crossed. The mutation factor is also important, since it introduces a variability that won't be present with the simple crossover between the subjects involved in the simulation. After several simulation steps, all the business units will have maximized the local output (or at least the agents will have reached the best possible result among the solution space observed), and thus also the final output, that is the result of the interaction between the various units, would be maximized. At this point, by looking which kind of agents survived to the selection and produced the optimal results, we can understand what the best way to operate each unit is.

Instead of using Genetic Algorithms inside the production units, we could use Classifier Systems to map the rules of the single processes; since the rules, modelled with Propositional Logic, can be encoded as if-then conditions, the Classifier Systems can map them successfully and evolve them in order to find the optimal organization of the process based structures. Again, this could be based on very simple performance indicators, like the local output, given a local input or the time required to complete an operation. Other constraints could be found for both the approaches; for example, it's possible that some of the found structures are not applicable to the real world. In this case, those should be discarded even if their local performance is higher than others.

CONCLUSIONS AND FUTURE DIRECTIONS

Evolutionary Methods derived from the studies in AI fields have proven to be effective problem solvers, particularly for situations where a near optimum result is enough and the real optimum is not needed. This suggests that they can be successfully applied to those situations in which the traditional approaches (e.g. mathematical ones) are inapplicable, since the system is too complex, dynamic or fuzzy. Enterprise simulation, done through the proposed Agent Based Process approach, can be an application field for Genetic Algorithms and Classifier Systems; here an enterprise or a part of it is split into a set of processes, made of basic building blocks, each of one modelled as a very simple rule of Propositional Logic. These are put side by side with reactive and evolving agents, in order to find an optimal structure for the modelled enterprise through self organization. The agents are evolved on the basis of a fitness function, representing some core performance indicator (e.g. the local output of the single production unit, the time used for the single process and so on). There are at least two main configurations for the proposed approach: the first one uses Genetic Algorithms for the agents operating every business unit in the simulation; the second one employs Classifier Systems to optimize the single processes, by mapping the Propositional Logic rules and making them evolve. In the future a working example of both the approaches will be created and some practical results will follow, showing that evolutionary methods can be a great help for decision making and business process reengineering. A meta-model for Agent Based Process Simulation is also in the works, to be considered as the prototype for all the models built in this way, showing which parts of a generic enterprise can be modelled using Logic based deterministic processes, and which require the use of intelligent learning agents.

REFERENCES

Bahrami, A., Sadowski D. and Bahrami S. 1998. "Enterprise architecture for business process simulation", *Proceedings of the 1998 Winter Simulation Conference*

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

Gilbert, N. and Troiztsch, K.G. 1999. "Simulation for the Social Scientist", Open University Press

Gilbert, N. and Terna, P. 2000. "How to build and use agent-based models in social science", Mind & Society 1, 57-72

Helsgaun, K.2000. "Discrete Event Simulation in Java", Writings on Computer Science, Roskilde University

Holland, J.H.1976. "Adaptation", In R. Rosen and F. M. Snell, editors "Progress in theoretical biology", New York: Plenum

Kim, D. 1993. "The Link between individual and organizational learning", Sloan Management Review, pp. 37 50.

McCartney, R. 2001. "A Short Introduction to Modal Logic", UNCG CSC 656, Spring

Remondino, M. 2003. "Agent Based Process Simulation and Metaphors Based Approach for Enterprise and Social Modeling", *ABS 4 Proceedings*, SCS Europ. Publish. House

Takadama, K. et al. 1999. "Making Organizational Learning Operational: Implication from Learning Classifier System" in J.Comp. and Mathematical Organization Theory, Vol. 5, No. 3, pp. 229-252.

Terna, P. 2002. "jVEFrame: a Virtual Enterprise Frame in Swarm", *SwarmFest 2002 Conference*, working paper

AUTHOR BIOGRAPHY

MARCO REMONDINO was born in Asti, Italy, and studied Economics at the University of Turin, where he obtained his Master Degree in March, 2001 with 110/110 cum Laude et Menzione and a Thesis in Economical Dynamics. In the same year, he started attending a PhD at the Computer Science Department at the University of Turin, which will last till the end of 2004. His main research interests are Computer Simulation applied to Social Sciences, Enterprise Modeling, Agent Based Simulation and Multi Agent Systems. He has been part of the European team which defined a Unified Language for Enterprise Modeling (UEML). He is also participating to a University project for creating a cluster of computers, to be used for Social Simulation.