

# Agent Based Process Simulation: Modelling Complex Systems Using an Hybrid Methodology

**Marco Remondino**  
**University of Turin, Italy**  
**Email: remond@di.unito.it**

## ABSTRACT

Process based enterprise simulations can be efficiently applied to *what-if* analysis and *business process re-engineering*, while agent based paradigms are effective when used to model complex social systems. Though, neither of the two can be successfully used to create enterprise simulations in which both the human factor and the structure are relevant and dynamically linked. A hybrid approach is thus presented in this paper: it uses propositional logic to model the deterministic structures acting as a background, while it employs evolutionary methods derived from the research on artificial intelligence to create agents capable of learning and self organizing. With this technique it becomes possible to find the optimal organization for a given enterprise and realistically model various situations in which deterministic structures, e.g. machineries, are operated by human beings and affected by exogenous factors. Some operational examples are discussed, and a building methodology is presented.

**Keywords:** business process reengineering, enterprise simulation, multi agent systems, evolutionary algorithms

## 1. INTRODUCTION

According to (Ostrom 1988), simulation can be considered a third way to represent social models; in particular, it can be a powerful alternative to other two symbol systems, the verbal argumentation and the mathematical one. The former employs natural language, thus being a non computable way of modelling, though a highly descriptive one. By using mathematical methods, everything can be done with equations, in principle, but the complexity of differential systems increases exponentially as the complexity of behaviour grows: describing complex individual behaviour with equations often becomes intractable. Simulation has a great advantage over the other two, consisting in its high portability on a computer, through a program or a particular tool, and in the possibility of describing a complex behaviour starting from simple entities. Computer programs can then be used to model either quantitative theories or qualitative ones. There are many different approaches to computer simulation; two of them are at the basis of this work and are used in different fields, with distinct goals: process based and agent based. Both of them can be used to model enterprises or firms, but with some fundamental differences, which will be discussed in detail. Agent Based Modelling (ABM) 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. For this reason, process simulation is not the ideal tool to model these complex environments, but is the most used approach to model structures, such as machineries or business units, in a deterministic and static way.

On the other side, there are formalisms which allow to observe 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 often complex and difficult to predict, and which can be used in open and distributed systems. In order to simulate situations in which the structure of the enterprise is fundamental, but also the human factor and the environment are crucial, a hybrid approach called Agent Based Process Simulation is proposed in this paper.

## 2. A HYBRID APPROACH FOR SIMULATION

Both process simulation and agent based simulation are powerful approaches for creating models of enterprises and complex systems, but they also have some flaws and limits. I'll concentrate the discussion on enterprise simulation, since this is the most suitable field for the proposed approach. While deeply describing both the existing methodologies is beyond the purpose of the present work, I'll analyze the main differences among them, which will lead to the hybrid formalism will be presented. Usually, process simulation is employed to model a very well structured and known situation, in order to perform what-if analysis and to support business process re-engineering: it's used to create models of parts of enterprises or mechanical/electronic systems. Its greatest advantage is that it starts with a basic scheme, often derived from existent documents, through which it becomes very easy to bring a real situation into a process simulator. This kind of approach is widely spread and allows to analyze a part of a whole, by studying its expected behaviour when changing something. This is why process simulation is a great support to strategic decisions; the simulator can answer to many questions and situations that would require big efforts and long time to be tested in the real environment; for example, a part of a manufacturing plant can be simulated by dividing it into its main processes. After that, it becomes possible to check what would happen on the final output if something is changed in its structure or in the input. 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 it. Each event has associated a 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 one 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, 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 as 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 to port the existing models from one tool to another. The second disadvantage is that, in order to subdivide it into its main processes, an enterprise must be very well known in all its parts; if something is uncertain or non deterministic, then it's impossible to validate a process based simulation as a model of the analyzed real situation. Besides, this method is quite static, meaning that the relations between the various parts involved in the model must be deeply described and there is no possibility of self-organization.

On the other end, it's advisable to choose agent based approach when the system to be simulated has a complex aggregate behaviour, not easy to describe just by studying and modelling the single entities; in a word, when the sum of the parts is not enough to describe the whole. So, if we want to model a society, a supply chain, an ant colony or a stock market, 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, in agent based simulation the most important aspect is the interaction among entities, i.e. the aggregate behaviour. By creating many, yet simple intelligent agents and letting them interact, complex behaviour emerges. For example, an artificial stock market can be simulated by creating some different types of intelligent agents, which follow inner rules (Terna 2000); some of them can simply act randomly, while others will study the trend before taking their own decision, or even 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. Besides, agents can be modelled with inner reasoning capabilities, a factor that can increase the coherence of a simulated system. Each agent can be endowed with the capacity to reason on the global effects of local actions, or even to create its own forecasts on the actions performed by other agents.

The agents built using this approach can choose the action to perform, according to the stimuli coming from the environment, and not simply following a given set of rules or a static pattern. In (Bonabeau, 2002), we read that ABM has three main benefits over other approaches: it captures emergent phenomena; it provides a natural description of a system; and it is flexible. In fact 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, when they can interact with other artificial agents, robots, and humans. An intelligent agent with all these features can be referred to as a Belief-Desire-Intention (BDI) one. There are mainly three agent based paradigms that can be applied to simulation:

*Symbolic*: highly structured and deterministic agents usually described through expressions of propositional or modal logic. This is perfect when a single agent, programmed with a wide set of rules, must interact with the environment, but it's not versatile when used to simulate big communities

*Sub-symbolic*: simple agents, with few rules of behaviour, but with dynamic patterns for the interaction with other

entities. 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 actions. It can be really hard to describe the real world under every aspect: some fundamental macro-actions can thus be defined on individual agents, which allow cooperation with the environment and among themselves. 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. A sub-symbolic agent can also be endowed with an artificial form of intelligence and learning; this allows a sort of reasoning on the actions to perform, that will not be based deterministically on the rules given by the programmer, but derived from the experience, the environment and other agents' actions

*Hybrid Architectures*: at the lower levels, we find reactive, sub-symbolic agents, while at the upper levels there are more complex and structured agents acting as background. In this way we can combine reactive capabilities with planning; symbolic agents involved will have a deterministic behaviour, described with rules, while the reactive agents will produce self organization patterns for the given situation in which they operate

If the aim is to simulate enterprises in which the environment and especially human factor are relevant, neither process simulation, nor agent based paradigms can fully accomplish the task. In fact, there are many tools used to create deterministic models of enterprises through processes, but none of them takes into account the existence of non-deterministic subjects. On the other side, with a pure agent based approach, it's possible to create self-organizing and learning entities, using techniques derived from the AI field, but it's extremely difficult and time consuming to build a realistic structure for an enterprise, defining machineries and so on.

### 3. TWO CATEGORIES OF AGENTS

The main idea behind the approach presented in this paper is the use of two different categories of agents, interacting in the same simulation; the first will be used to model the processes, in a deterministic and logical way, while the other one will represent the human beings and other non-deterministic parts of the environment in which an enterprise operates. Since processes can be usually modelled as deterministic flows of actions, I will discuss how propositional or modal logic can be successfully used 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 (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, propositional logic could be enough, since it allows the creation of truth tables for

the sub-processes, here defined as the lowest level of decomposition that could be represented by a binary Boolean function. Modal logic could create 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. possible). In the framework presented here propositional logic only will be used to model simple processes: it allows an analytical description of a process and the creation of its model. Besides, it simplifies the transition to programming code, required to port it into a working simulation. A sub-process is an element that can't be simplified anymore; it consists of a Boolean formula, with two inputs and an operator. It produces output\_1 if the logic formula is True, or output\_2 if it's False; one of the two outputs can be simply a dummy variable, bringing the Void value, in case only one result is needed. The sub-processes are linked among themselves by their inputs and outputs. In this way, a part of a deterministic process can be represented by a schema like the one in Figure 1.

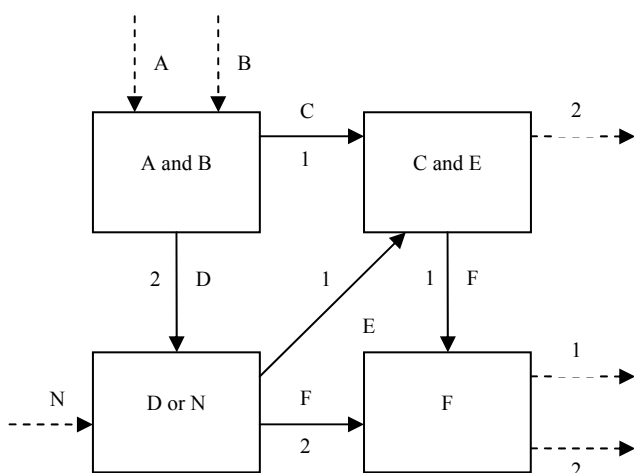


Figure 1: Propositional Logic Used to Represent Processes

Porting this kind of structure to a programming language is a very easy step, since all the elementary blocks can be converted using if-then functions and logic operators. In this way even a very complex deterministic process can be modelled starting from very simple building blocks. Modal logic could even add concepts of probability and necessity to a given output. In this case a probability function can be given, specifying how often, averagely, an output can be produced given the same formula (an average probability of 100% means necessity).

This approach allows the creation of software models for machineries and production units; the most difficult part to simulate, but probably also the most interesting for a social context, is the human factor, e.g. the workers involved in the structure of the enterprise. Finding the optimal enterprise organizational structure is a very difficult task to accomplish, though a crucial topic. Some techniques, derived from AI studies, allow embedding a sort of reasoning and learning capabilities into software agents; in particular, genetic algorithms and classifier systems will be discussed. In some situations effective results can be obtained just by building simple agents, whose behaviour is randomly determined or is built by applying basic and predefined reaction rules; this is the best 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 (e.g. studies on ant colonies); of course, the way in which 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 endowed with the ability to compute rules and strategies

and able to evolve according to the environment in which they act; in order to model them, we can use some techniques derived from the research on artificial intelligence, such as neural networks and evolutionary algorithms. While the formers are collections of mathematical functions, trying to emulate nervous systems in the human brain in order to create learning through experience, the latter derive from observations of biological evolution. Genetic algorithms are inspired to Darwin's theory of evolution, often referred to as "survival of the fittest": individuals are modelled as strings of binary digits and are an encoding for the solution to a given problem. The first generation of individuals is often randomly generated, and then some fitness rules are given (i.e. benchmarking parameters), in order to select the fittest entities that will constitute the best solutions to the problem. The selected ones survive, while the others are killed; during the following step, a cross among some of the fittest entities occurs, thus creating new individuals directly derived from the best ones of the previous generation. This is usually done by randomly selecting some couples of strings, breaking them at the same point and swapping the parts. Again, the fitness check is operated, followed by the cross and so on for many steps. In order to add a random variable to the genetic paradigm, that is something existing and crucial for evolution in the real world, a probability of mutation is given; this means that from one generation to the next, one or more bits of some strings can change randomly. This creates totally new individuals, thus not leaving as candidates only the direct derivatives of the very first generation. Genetic algorithms have proven to be effective problem solvers, especially for multi-parameter functions optimization, when a near optimum result is enough and the exact 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 processed 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 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 crossing 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 selected ones. It's crucial to carefully choose the basic parameters, such as cross 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 is composed by a collection of agents called classifiers, which through training evolve to work together and solve difficult, open-ended problems. They were introduced in (Holland 1976) and successfully applied, sometimes with 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 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, that's the basic technique extended to multi-agent environments by introducing the concepts of organizational learning. According to (Takadama 1999), in such environments agents should cooperatively know each other and learn to solve a given problem. The system often solves a problem with multi-agents'

organizational learning, when the same problem cannot be solved simply by the sum of individual learning of each agent.

Agent Based Process Simulation is a way to model deterministic structures, made up of single processes divided into building blocks based on propositional logic; these are then put to interact with agents belonging to the sub-symbolic paradigms, modelled with AI techniques. This allows to simulate situations in which not only the deterministic structure, but also unpredictable situations caused by the environment and the human factor are important. The intelligent agents can cooperate and organize, having a deterministic structure as their background; genetic algorithms and classifier systems will evolve, side by side with structures built using propositional logic, representing real machineries, business units and so on, and thus they will be able to give precise solutions to real, open handed problems.

#### 4. APPLICATION EXAMPLES

Usually, in process based models, the connections between the parts are managed with random numbers generators, probability functions or static sets of rules. This can be realistic when we must deal with simple Yes/No problems, but becomes insufficient when the situations are more complex: when a process is not linear, meaning that two or more different ways can be followed by the token, a binary function is used. This construction, very simple and effective for static situations, becomes unusable for complex environments, where a decision can't be simply a Yes or No, but also involves a structural change in the next step, according to some parameter coming from the environment. For example, we can now suppose that, in the model description, we have some subjects (production units, components, products, planner, and warehouses) and these rules:

1. One or more final products, made of components, can be assembled at each step
2. Each unit can produce only one kind of component
3. Not all the units require the same time to complete their own component
4. At each step, a unit can, according to the Planner's previsions:
  - Start producing a component for the warehouses
  - Start producing a component for the market
  - Continue the production started in a previous step
  - Do nothing
5. Warehouses can be managed in different ways (LIFO, Last In First Out, FIFO, First In First Out...) according to the market
6. Unit [n], in order to decide whether producing or not, must "ask" to the Planner
7. The Planner decides whether "answering" Yes or No to a unit, according to its previsions (derived from the market, and other endogenous parameters like costs and time)
8. If the Planner says Yes, then the Unit[n] must watch into the warehouse:
  - If the component is in the warehouse, then it's used, else it's produced

This simple example is very difficult to represent in a standard process simulator, since the way to manage warehouses can't be dynamically changed according to the market and to the orders. Besides, if we want the Planner to be realistic, then it can't be just a random generator or a probability function, but must be able to decide, according to the different situations (orders, market, costs, goods in the warehouses, times and so on). Besides it's very important, for a real planner, to be able to improve his previsions, according to previous experiences. In

this framework, while the manufacturing units can be described with a process based approach, it'd be much more realistic to use intelligent agents for warehouse management and decision making. Neural networks, genetic algorithms or classifier systems can be encapsulated into these agents which, in this way, could be proactive towards the environment.

The Planner could start making random previsions and obtaining terrible results; for example, he could decide to produce even if there are no orders and many components are already stored in the warehouses. If the Planner's "mind" is built using genetic algorithms or classifier systems, then the actions that produced a result lower than a certain threshold will be discarded, while the others will survive and cross among themselves, producing a new set of rules, and so on; after a number of learning steps, the planner will be able to produce very accurate forecasts, whatever happens around him.

As another example, we may think of a generic enterprise in which many sub-systems, i.e. business units, can be described with a pure process based approach. The interaction between these basic subsystems, though, is usually really complex in the real world and generally involves a human and non-deterministic participation. This would again be 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 just modelled with structured and logic based processes. In this case, intelligent agents can be used as connections between the process based sub-systems. These agents could be quite simple, but structured ones, able to start from stimuli coming from the environment (i.e. the output of a unit), and consequently affecting the way other sub-systems will work. The model built in this way can give representations of what happens when changing the management structure, when using more or less experienced employees or even when the workers organize a spontaneous strike. The scheme for this situation is shown in Figure 2.

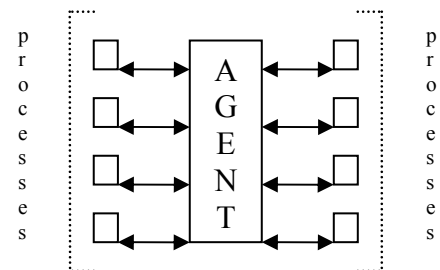


Figure 2: Intelligent Agents as Connection Among Process Based Business Units

With the same approach, it's possible to go down to a micro level, by inserting agents into models of single machineries and business units. In a single, but very complex machinery, not all the parts are strictly deterministic, in the sense that they may be affected by some unforeseen influence coming from the environment and react in an unexpected way. By using a process based approach, it is possible to model the machinery quite deeply, but just in a deterministic and static situation. Only the optimal environment, in which nothing can change its way of working or, at best, a finite set of scenarios can be implemented. With such a tool we can simulate the variation of the output by varying the input, or by changing part of the system; or we can prove the resistance and endurance of the machinery, but always in optimal or defined conditions.

Though, such a simulation rarely reflects the real world; a pure process based approach, for its nature, wouldn't be able to include any chaotic or unforeseeable exogenous action, that

could compromise the machine operations (e.g. damages caused by moist, fire, and so on). The model would act as a function which receives an input, that is the independent variable ( $x$ ), processes it and produces a deterministic output, which is the dependent variable ( $y$ ). Though very powerful and easy to validate, this is not always realistic. A representation of a typical process based model of a machinery is given in Figure 3.

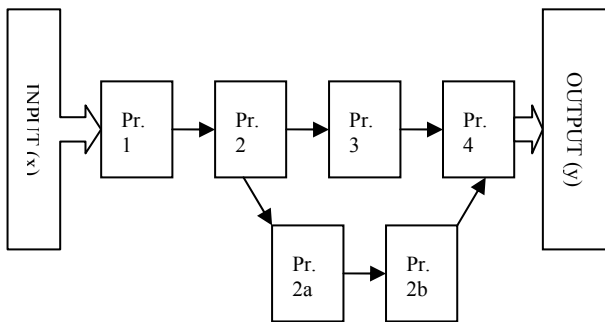


Figure 3: Typical Process Based Representation of a Machinery

By considering certain parts of the machinery as very simple agents, it would be possible to create a more realistic model of the object, that could react to the stimuli coming from the environment according to certain self-evolving rules. This can give the whole machinery a complex and less deterministic behaviour, similar to that it would show in the real world. The previous schema is thus changed as shown in Figure 4.

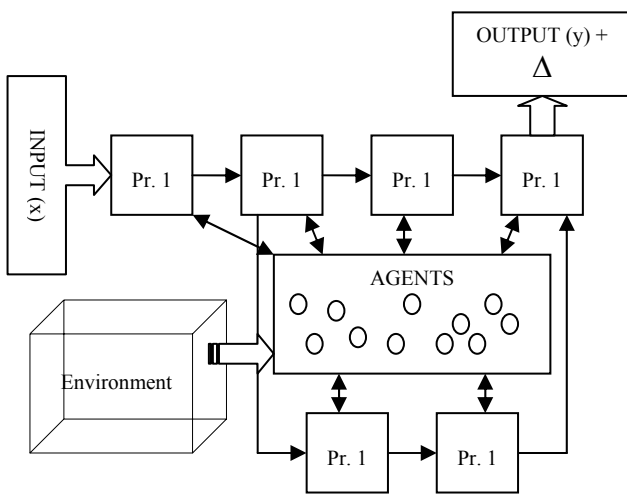


Figure 4: Same Schema, but with Agents Reacting to Environmental Stimuli

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, the employees or the person running the simulation. In this way, the model is not strictly deterministic and static, in the sense that given an input ( $x$ ), same as before, the output is no more a linear function of just the independent variable, but also of those factors processed by the agents, that can't be determined a priori. The output is not the deterministic ( $y$ ) as before, but ( $y$ ) plus a delta, which is caused by external, unforeseeable 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 the creation of models

which are more realistic and dynamic, as opposed to the static ones, where only the deterministic flows are simulated. The greatest difficulty, when using this approach, is model validation, because the unexpected circumstances are difficult to reproduce more than once. The validation could then go top-down, in the sense that first we observe certain situations in the reality and then artificially try to reproduce them in the model, by using the agents in a piloted way. If a comparable result occurs, it is possible to calculate an average error and validate the model for those particular situations. After that, it's possible to extrapolate these results, and consider the model valid also for other situations that can't be controlled and created at will 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, two goals can be achieved: 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 or business unit, which would be modelled through deterministic processes. In order to do that, we start from some basic parameters and performance indicators, which will serve as the benchmark to determine the fitness of the agents. The agents that 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 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 formalisms derived from propositional logic. The human beings involved in each production unit are modelled as agents based on 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: we could say that they start operating the units in a random way. The agents that produce the best local output, given a certain input, are saved and crossed among themselves. 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 wouldn't exist with the simple crossover among the subjects involved in the simulation. After several steps all the business units will have maximized the local output (or at least the agents will have reached the best possible result in the solution space observed), and thus also the final output, that is the result of the interaction between the various units, will 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 genetic algorithms, classifier systems could be used to map the rules of the individual 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 for the process based structures. Again, this could be based on very simple performance indicators, such as the local output given a local input and the costs, 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 couldn't be applied, in the real world. This could be caused by laws, syndicates or other issues; in this case, those solutions should be discarded even if their local performance is higher than others.

## 5. CONCLUSION

In this work, I have examined two existing approaches to enterprise and social simulation: process based and agent based. Both of them are very powerful and effective for certain situations, but none of them allows to simulate situations in which it is important to model the deterministic behaviour of a machine or the structure of an enterprise, but also the human factor and the environment are crucial. A hybrid approach is thus proposed, which uses symbolic and structured agents to represent processes, modelled using propositional and modal logic, while all those aspects that can't be strictly modelled with a deterministic structure are built as sub-symbolic agents by employing paradigms derived from the AI field.

In particular, these agents will be able to evolve and self-organize, by operating in the environment described with process based techniques. This is particularly suitable for enterprise simulations in social context: the machineries and production units can be modelled using process based approach, while the workers at every level and many aspects of the market (such as competitors, orders, customers, suppliers and so on), can't be represented by deterministic structures. By using genetic algorithms or classifier systems and putting them to operate in a deterministic environment, they will evolve according to some core parameters till they reach an optimal organization for a given situation. This can be particularly effective when the aim is to study the optimal human structure for an enterprise, or also when a machinery must be simulated keeping into account unforeseeable exogenous factors that could affect it. Some operational situations have been examined, that couldn't be represented with pure process based or agent based approaches, but can be successfully modelled using Agent Based Process Simulation.

## 6. REFERENCES

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

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.

Ostrom T. 1988., Computer simulation: the third symbol system. Journal of Experimental Social Psychology, vol. 24, 1998, pp.381-392.

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. 2000., SUM: a Surprising (Un)realistic Market: Building a Simple Stock Market Structure with Swarm, working paper.

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