AGENT BASED PROCESS SIMULATION FOR MANAGEMENT AND ECONOMICS

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, complex behaviour, process, metaphor

ABSTRACT

Simulation is an increasingly popular way of describing social models, alternative to other two symbol systems: the verbal argumentation and the mathematical one. The advantage is the high portability on computers; programs can then be used to model either quantitative theories or qualitative ones.
There are mainly two approaches: Process Simulation, which is generally used to create models of well known parts of enterprises or mechanical/electronic systems and Agent Based Simulation, which allows to study the emergence of social behaviour with the creation of models, known as "artificial societies".
The main goal of this work is to present a hybrid formalism, which uses the best parts of the two, to build realistic economical, management and social models. Metaphors Based Modelling is also introduced as a technique for converting a real social system into a computer model.

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. Simulation has a great advantage over the other two, which is to be found in its high portability on a computer, through a program or a particular tool. Computer programs can then be used to model either quantitative theories or qualitative ones.

There are mainly two different approaches to computer simulation, both of which lead to the creation of a computational model of a social or complex system:

1. Process Simulation, which is used to create models of well known 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 deeply analyze a part of a whole, studying its behaviour with "what if" analysis. This is why process simulation is a great support to decisions. 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. The second 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 well described and there is no possibility of self-organization.

2. Agent Based Simulation; 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. On the other side, there are 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 often complex and difficult to predict, and which 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 social simulation.

Symbolic: highly structured agents, described through expressions of 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.

Since an agent based model must be converted into software, this kind of simulation has a perfect counterpart in Object Oriented languages. In this kind of high level languages, it's possible to create a parent class, from which many similar objects will derive, with the same features. In this way, the code will be written just once, while creating all the agents that are needed. Besides, the single objects interact among each other in a simple, yet very powerful way. An object can "ask" something to another one, which "answers"; this is the ideal way for letting the single agents communicate among them.
Though, the fundamental step from a simple OO program and an agent based model is that, while in the former the objects can only communicate and execute actions, in the latter we want the single agents to have self-organization capabilities, and autonomous behaviour, which are based on Artificial Intelligence systems, like neural networks, genetic algorithms or classifier systems.
We can thus build an open system, with information interchange and coordination among agents.

ENTERPRISE SIMULATION: THE SCENARIO

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 here. I'll concentrate the discussion on enterprise simulation, since this is the most interesting field for the proposed approach. While deeply describing both the approaches is beyond the purpose of the present work, I'll just write few words on 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. The simulator can answer to many questions and 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's possible to check what would happen on the final output if something is changed. This is a very powerful tool, but requires a very deep knowledge of the real environment to be studied.

On the other end, it's advisable to choose Agent Based Simulation when the system to be simulated is very complex and not easy to describe; in a word, when the sum of the parts is not enough to describe the whole. So, if we want to model an entire supply chain, 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 approach. 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; some of them can simply act randomly, while others will "study" the trend before acting. Some of them could 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, which can increase the coherence of a simulated system. 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.

AGENT BASED PROCESS SIMULATION

There are many intermediate situations, though, in which neither Process Simulation nor Agent Based approach can be applied with good results: we may think of a generic enterprise, in which many sub-systems 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; that's where we can use agent based connections between the sub-systems.

These agents could simulate the behaviour of people that must take decisions, with certain rules or through artificial intelligence patterns; the 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.

As an example for this approach, we can think of two assembly lines, both belonging to the same enterprise. The second one needs the output of the first one, in order to produce its own output; it wouldn't be realistic just to create a process based simulator, in which the two lines are directly linked. Between them, we can have many different agents, that are, for example, people who manage the warehouses, or simply robots that must collect the output of the first assembly line, and bring it to the second (Figure 1). That's where the agent based approach shows its strength, allowing the observer to create self organizing entities, which can react to different situations in different ways. In a simulation built in this way, we can see what happens if we change the way we use the warehouses, or, for example, if the workers are on a strike.

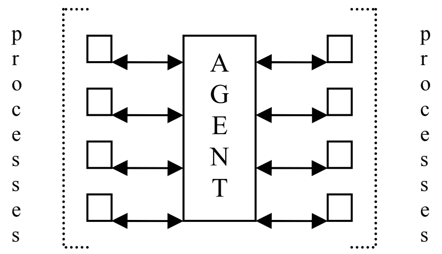

Figure 1: An Agent as the Connection among Processes

The one described above is the situation that might take the biggest advantage on the Agent Based Process approach I'm describing here, but it's not the only one. 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 single situation, that is the optimal environment, in which nothing can change its way of working.

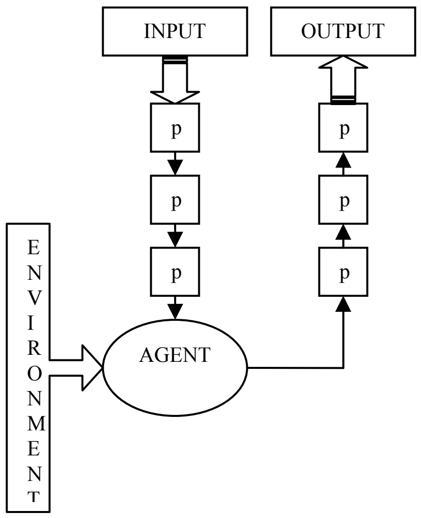

Figure 2: A Simple Agent as a Part of a Machinery

By considering certain parts of the machinery as very simple agents (Figure 2), 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.

APPLICATION SCHEMES

Usually, in process based models, the connections between the parts are managed with random numbers generators, probability functions or static sets of rules. That can be realistic when we must deal with simple Yes/No problems, but becomes insufficient when the situations are more complex. In Figure 3, a typical scheme for a process based model is shown: when a process is not linear, meaning that two or more different ways can be followed by the token, a binary function is used.

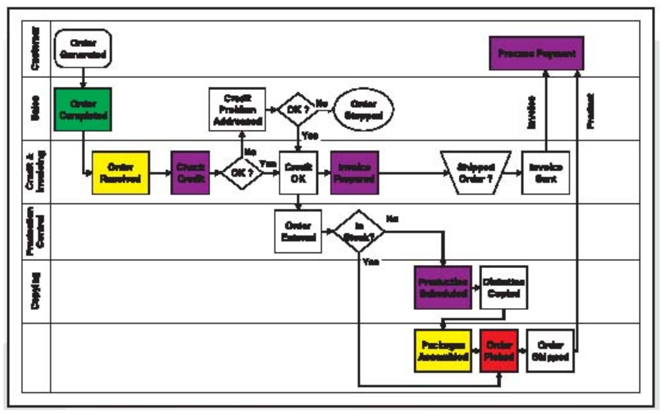

Figure 3: A Traditional Process Based Model

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 a change in the next step.
For example, we can now suppose that, in the model description, we have some subjects (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. The Planner decides whether "answering" Yes or No to a unit, according to its previsions
7. Unit [n], in order to decide whether producing or not, must "ask" to the Planner
8. If the Planner says Yes, then the Unit[n] must watch into the warehouse:
9. If the component is in the warehouse, then use it, else produce it

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 environment (orders, market, goods in the warehouses, and so on).

That's why, while the manufacture units of the enterprise 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 the agents, which, in this way, could be proactive towards the environment.

METAPHORS BASED MODELING

Even if Object Oriented approach is optimal to create agent based models and simulations, it's often very hard to transpose the observed features of a social system into a computer language. A formal method, called Metaphors Based Modelling is thus introduced here. It consists in using powerful metaphors, which allow the porting of a real situation into a software model. Of course, a computer can't understand concepts like "product", "seller", "buyer", "cost", "value chain" and so on; it's necessary to translate these concepts into something more similar to the computational structure of a machine.

If this action is performed without studying the consequences, it could bring to wrong results; in fact, an error at this level could compromise the whole model. That's why I propose a simple, yet powerful formalism, to validate the metaphors used to translate the real system into the model.

A metaphor must be found, which corresponds to the real situation, and which makes it easy to convert the observed situation into a programming language. High level languages, such as Java or C++, have advanced functions built in, such as multi-dimension arrays, typed variables, polymorphic objects, and so on. Above all, qualitative measures are very difficult to achieve, from a computer simulation: if we want to simulate customer satisfaction, or the quality of products, we must find a way to convert these in quantitative values. In Figure 4, the basic scheme that I propose to create metaphors is shown.

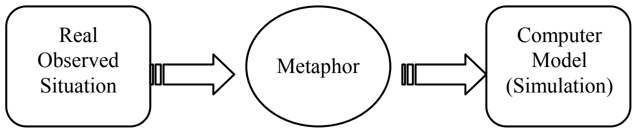

Figure 4: From the Real Situation to the Model

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, don't 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, which I'll call counter-metaphor, has to be directly derived from the metaphor used to port the observed system into the simulated model. This counter-metaphor will allow going back from the results obtained from the model to others that can be compared to the real data. So the previous scheme can be completed in the way shown in Figure 5.

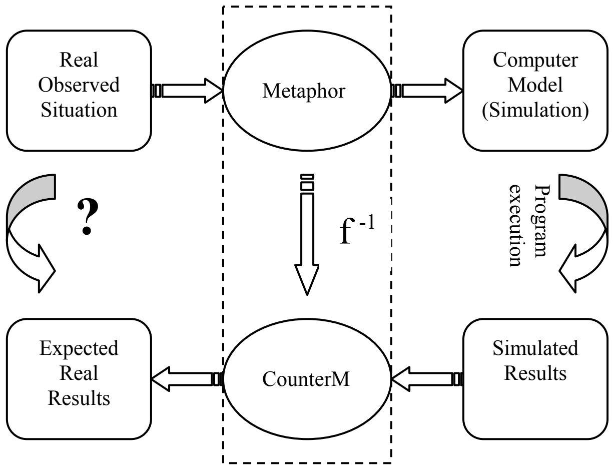

Figure 5: Metaphors Based Modelling Formalism

If, and only if, the counter-metaphor is exactly the inverse of the metaphor used, a model can be validated as representative for the reality we observed and want to simulate. For example, if we try to simulate a database-security environment for an enterprise information system, we can think of a metaphor which allows us to translate into a computer program the risk deriving from allowing everyone to access our files. Before doing this, we must study the statistics of the files accidentally destroyed in the real situation, in a previous period, and then build a function of probability, to be applied on an array, which represents the whole database.

At this point, we can change the security features, denying permissions to certain agents, and see what happens. Of course, this will reduce the risk of destroying files, but will require a longer time to operate on data. In fact, a subject who doesn't have the permission of modifying a file, must ask another one, which in turn has the permission, to do the work for him. Also the hierarchy must be derived from the real situation.

The results, which are "time elapsed" and "data corruption", are effective for the simulated model, but can be brought back and applied to the real situation, since we used a mathematical function of probability to simulate the corruption of the files, and a hierarchy which is directly derived from the real one. A counter-metaphor is thus easy to find and apply to the simulated results.

This is, of course, a very simple example, but this method can be used both for easy and complex systems, and assures good results and a correspondence of the simulation to the reality.

Another example is found in (Terna 2002): here, the author uses a powerful metaphor to model complex products, made of many different components. As shown in Figure 6, each part is represented by an integer number, that can be easily manipulated by a computer, and the whole product is a bi-dimensional matrix, in which the first row is an ordered array, containing these numbers, and the second row is another array, which has a zero (0) when the corresponding component is not yet assembled, and a one (1) when it is.

component	10	22	27	24	18	8	16	14	25	25
status	1	0	0	0	1	1	1	0	1	0

Figure 4: Example of a Metaphor

This is a very general metaphor, which can be applied to different enterprises, just by studying the frequency of the components to be used, and the composition of the final products.

A counter metaphor is also very easy to find, since we can just step back, with a conversion matrix, from the numbers to the real components, and a model based on this metaphor can be easily validated, according to the formalism I proposed here.

CONCLUSIONS AND FUTURE DIRECTIONS

In this paper, I have examined the feasibility of a hybrid approach for enterprise and social simulations, which takes the best part of both Process and Agent Based Modelling. Additionally, I presented a formalism to convert a real situation into a computer program, based on metaphors, which can be used to validate the models.

In the future work, I'd like to apply both methods to real enterprise models and create a simple tool which allows creating Agent Based Process Simulation. I also plan to create a general meta-model for this new kind of approach, which could be a framework for all the simulations built in this way.

REFERENCES

Huhns, M. and Singh, M. 1997. "Readings in Agents", Morgan Kaufmann

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

Gilbert, N. and K.G. Troiztsch 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

Pistoiesi, G and Paolucci, M. 2000. " Simulazione Sociale e Sistemi Multi-Agente", ThinkinGolem pscrl

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