Agent based artificial experiments in social science with jESOF

Pietro Terna
Dipartimento di Scienze economiche e finanziarie G.Prato, Università di Torino
terna@econ.unito.it

Abstract

We introduce here a small set of artificial experiments in social science and a simplified tool useful to develop them in the perspective of the analysis of complex systems, with different degrees of cognitive definition of agents' behavior.

The straightforward way to build this kind of structures requires the use of agent based simulation techniques, i.e. models in which small software routines behave representing artificial agents within artificial environments containing (in some case) ecosystems that allow the emergence of institutions such as market, organizations, hierarchal structures.

jESOF (java Enterprise Simulation Open Foundations, web.econ.unito.it/terna/jes) is a tool created to simplify the access to agent based simulation for social scientists, whose competences are not necessarily are the same of high level specialized computer scientists. jESof has been developed upon the Swarm (www.swarm.org) simulation library.

The two examples reported in this analysis show that it is relatively easy to develop models with jESOF and that we are virtually unconstrained in the definition of the contents of our simulation models.

The first model is a test implementation of the well known Preys Predators Model (PPM) and allows the introduction of non only two, but three or more interacting levels (hyper-predators etc.). PPM uses fixed rules in a deterministic way and complexity comes from the characteristics of the problem and of its environment.

The second model is based upon two coevolving population: Workers, with their Skills, and Firms (WSF): the two populations have quasi-independent behaviors, based on a bounded rationality paradigm. Despite the relative independence of the two populations, structures emerge, with the shape of industrial districts.

Keywords: agent based simulation, industrial districts, Swarm, labor market

Simulating organizations: from jES to jESOF

We can hardly use traditional equation based models to investigate organization behavior and, mainly, enterprise behavior; this is the direct consequence of non-linearity and complexity, on one side, and of the not quantitative and not rational basis of a large part of decision making in organizations and enterprises, on the other side.

As stated in Burton (2001), simulation requires us to specify the world we want to investigate. It can be rich or complex, or it can be simple. It can begin from simple and evolve into complex. We must specify the "black box"; we cannot just assume it exists. In simulation, we make behavioral specifications, not behavioral assumptions.

The central issue is that we know more about the simulated rich world than is usually the case when we use "real" world as our laboratory. With this necessary specification, the simulated world is a laboratory where we know important parameters because we specified them; we did not assume them.

With the simulator we can reproduce in a detailed way the behavior of a firm, of an organization, of a system of firms and organizations into a computer, specially if we build the simulation model employing agent based techniques (Axtell, 2000; Terna and Gilbert, 2000; Tesfatsion, 2001; Gilbert and Troitzsch, 2005).

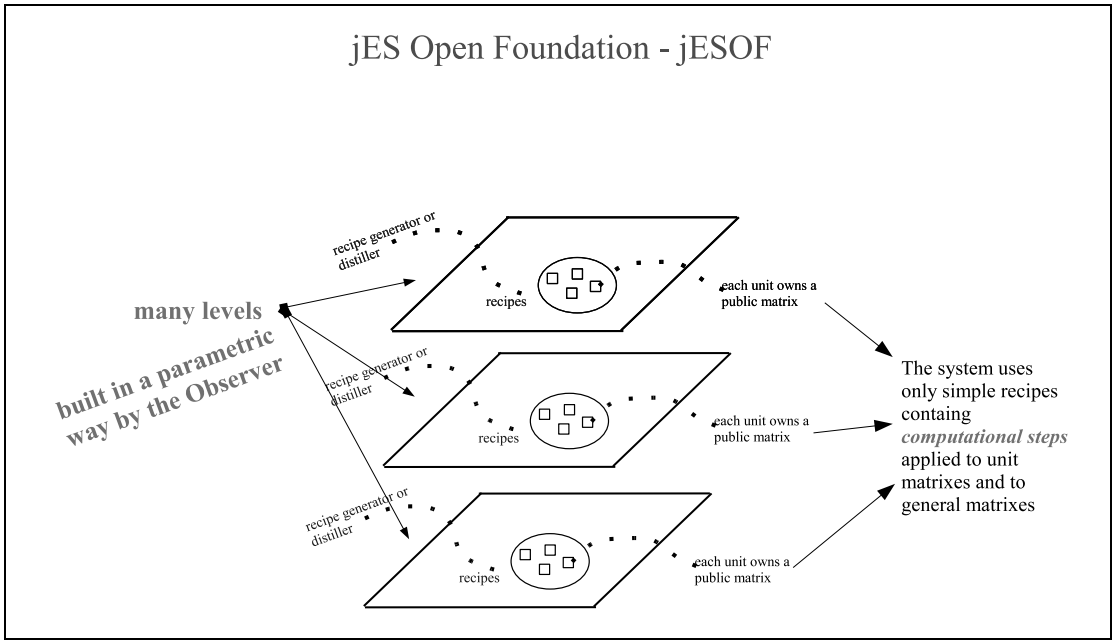

Fig. 1. The conceptual scheme of *jESOF*.

To run enterprise and organization simulations we introduced a few years ago a Swarm based framework, the Java Enterprise Simulator (*jES*, http://web.econ.unito.it/terna/jes). With our tool we describe in a detailed way a two side world, considering both the actions to be done, in terms of orders to be accomplished (the "What to Do" side, WD), and the structures able to do them, in terms of production units (the "which is Doing What" side, DW). Our simulation model is, first of all, a description of the enterprise or of the organization, as it is. Just like the various *flight simulator* programs put at our fingers the control of the simulated airplane and then execute our choices, *jES* executes exactly what we suggest has to take place into the simulated enterprise, on the two sides described

above. The plane flights or lands; the enterprise produces or stays clogged if our WD and DW choices are inconsistent.

The basis of the method has to be found into agent based simulation techniques, i.e., the reconstruction of a phenomenon via the action and interaction of minded or no minded agents within a specific environment, with its rules and characteristics. In our *jES* case, the world is populated with the things to be done (orders) and the units able to work with them; we have also minded agent s, i.e. the agents expressing decisions within the model.

The step ahead is *jESOF* (jES Open Foundation), a *jES* generalization built to simulate multi-model frameworks of system of units or agents. Both packages are based on Swarm (www.swarm.org), the first multipurpose library of functions for building agent based simulation models showing: (i) a clear distinction between the model layer and the observer layer; (ii) a direct tool to deal with time and events, via an object oriented construction of the groups of actions and of the related schedules; (iii) a system of probes to be used to look inside agents while the model is running and the agent are behaving into the artificial environment.

jESOF simulates systems of enterprises or production units (or systems of units without direct economic meanings, as in the case of the preys-predators model) in static or evolutionary contexts. In this second case new unit-agents arise continuously and some of the old are dropped out.

In Fig. 1 we present the conceptual scheme of *jESOF*, with several interacting models; in particular the scheme shows:

i. units: a unit is able to perform one of the steps required to accomplish an order, as a productive structure or, more generally, as an agent;

ii. orders, representing goods or services to be produced or actions to be done; an order contains technical information and their recipes, i.e. the sequence of steps to be accomplished to perform them;

iii. sequences of orders, coming from archives and managed via an “order distillers” or randomly generated via an “order generators”; may also concern the construction or elimination of units;

iv. matrixes to manage data, both at a unit level and at a general environment level; recipes, as sequences of steps, refer to matrixes running computational steps;

v. time schedule, managing “order distillers” and “order generators” actions, one for each layer or model.

All this features are driven through to a specialized time oriented script language, incorporated in text files and spreadsheet files; you can find the formalism in a document contained in *jESOF* distributions (look at http://web.econ.unito.it/terna/jes_files/ for the latest *How to use jES_O_F* file). The key of the script language is constituted by the recipes, simple as sequences of single acts or complex ones, calling computational functions written in Java, which are useful: to make complicated steps such as prey-predators or firms-workers interactions; to account for the actions of the different units etc.

Why jESOF?

Why another simulator? Simulation of organization simulation can be managed via system dynamics models, discrete event models or agent based models (Borshchev and Filippov, 2004); jESOF belongs to the field of agent based simulation, but with a lot of attention to the accurate description of the sequence of the events; most of all, it offers to the users not only a new paradigm for the analysis and the description of organizations, enterprises or environment as a whole, but also an easy to use tool to directly implement the outcome of those analyses (as you can see in the Appendix, where the script coding conventions of jESOF are shown).

jESOF at present is built as a layer upon Swarm, but light experimental releases exist also for NetLogo (http://ccl.northwestern.edu/netlogo/), StarLogo (OpenStarLogo, the new Open Source release of StarLogo, is at http://education.mit.edu/starlogo/, together with StarLogo TNG, which has graphical programming capabilities in a 3D world) and JAS (http://jaslibrary.sourceforge.net/); it would be possible to develop jESOF also upon other tools, such as RePast (http://repast.sourceforge.net/), AnyLogic (http://www.xjtek.com/), Ascape (http://www.brook.edu/es/dynamics/models/ascape/), and many others.

What is very important is that the simulation environment that we use to implement jESOF could be linked to new portions of code, wrote in Java or Python or any other language to develop completely the jESOF infrastructure and its capability of being extended, if necessary, by the user, through high level coding.

Comparing two models

About the models introduced in this paper, we underline first of all the presence of *soft* or *hard* links or connections in agent interaction. A PPM and WSF comparison is displayed in Fig. 2.

As we can see below, workers and firms are obviously connected, but with a form of loose relationships, where firms are hiring daily the workers specified via a simulation supplied sort of throughput and workers can work daily also for more than one firm; firms are able to store the excess of work, if any, and to employ the stored quantities subsequently. In the future a new version of the model will consider a strictly accounted form of relationships, implying that only necessary workers are hired in each production period and that the maximum work capabilities are controlled (future version WSF-2).

	Co-evolution with direct mutual benefits	Co-evolution with direct and indirect[1] mutual benefits

[1] Indirect = Predators eat preys allowing their survival as a population.

Soft links	WSF	-
Hard links	WSF-2 (future version, with a strictly accounted connection for workers and firms)	PPM

Fig. 2. Links or connection among agents.

The PPM works in a opposite way, because each predator agent is supposed to eat exactly one prey in each cycle, if it is possible to find one of them.

It is interesting to underline that both models are producing results with interesting results, offering some emergence insight, and suggesting the possibility to explore, as a theme of research, the level of strength to be placed in stylized agent links.

The Preys Predators Model (PPM)

We introduce now a preys predators model, with three levels instead of the classical two. A first level contains the grass; the grass grows at a fixed rate applied to the existing units; the new units appear near the existing ones. A second level contains the preys, or rabbits, eating the grass; in case of shortage of grass, rabbits disappear. The third level contains foxes, eating rabbits; in case of shortage of rabbits, foxes disappears. New rabbits and foxes appear at a fixed rate also in these cases applied to the existing units; the new units appear near the existing ones. If a level becomes empty, it is impossible to fill it again, due the rate of growth mechanism. Rabbits eat the grass and foxes eat the rabbits only if the units are reciprocally visible, i.e. if their light areas in Fig. 3 have in common at least one point.

Look at the complex codetermination of grass, preys and predators in the figure: with this model it is easy to produce complex evolutionary scenarios, like that in which predators disappear and after few cycle grass goes to a minimum level, due the excess of prey quantity.

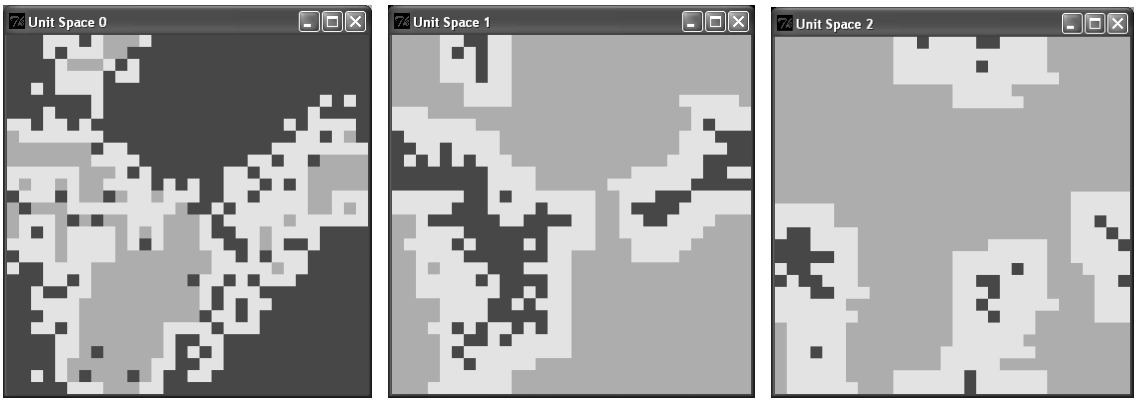

Fig. 3. The grass, on the left; the rabbits, in the middle; the foxes, on the right. Dark area are agents of the three types, light area report the visibility space of the agents.

We introduce here this well known model to compare the hard link used among agents in this case to the soft one described in the WSF model. But we introduce it also to demonstrate the easiness of jESOF programming, with the characteristic reported in the appendix.

Workers, Skills, Firms (WSF) model

The model has five strata: stratum 0 for the workers (all together), which are separately represented also in strata 1, 2 and 3 according to their different (three) skills; stratum 4 contains the enterprises.

The starting point is the global worker stratum (stratum 0), with an initial presence of all the types of workers (types 1, 2 and 3; types and skills are coincident).

The environment can be interpreted as a (social) space with both (i) metaphorical distances, representing trustiness and cooperation among production units (the social capital), or (ii) physical distances among the several units.

The activity, in an abstract sense, is represented by sequences of orders; each order contains a recipe, i.e. the description of the sequence of steps to be done by several units to complete a specific production or action. Two units can cooperate in the production process only if they are mutually visible in our (social) space, i.e. when their visibility areas (growing around them) are overlapping.

Units that do not receive a sufficient quantity of orders, as well as the ones that cannot send the accomplished orders to successive units, disappear. New enterprises continuously arise, in the attempt of filling the holes of our social network. A complex structure emerges from our environment, with a difficult and instable equilibrium whenever the social capital is not sufficient.

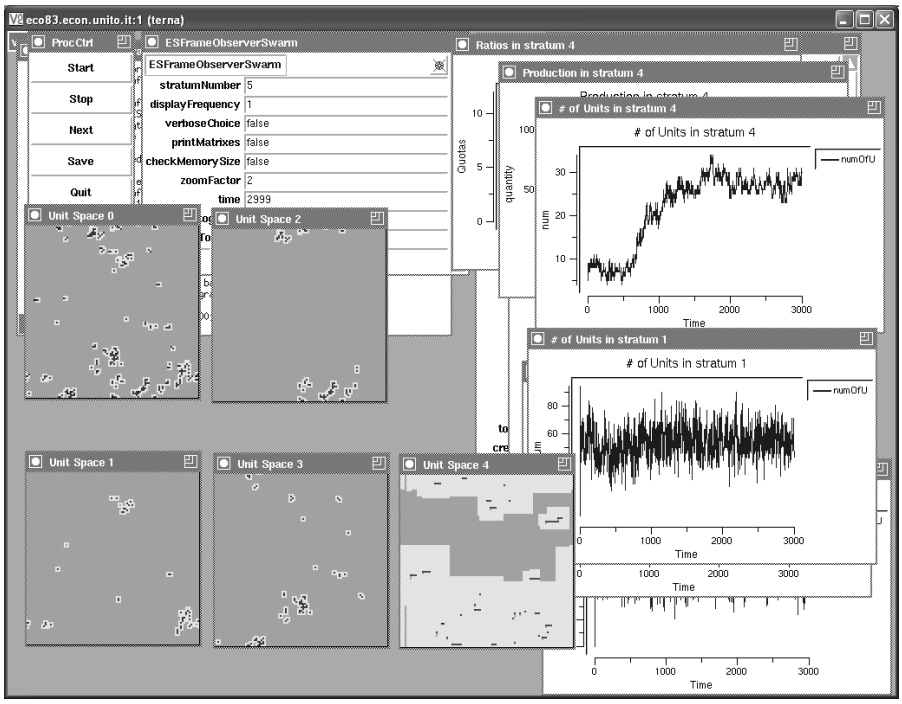

Fig. 4. Workers concentration by type of skill, with a relatively slow takeover of the economic system.

The scrip language uses two different sets of recipes included in orders. In the firm stratum we have recipes related to production, with sequences of steps describing the good to be produced. In the workers stratum, which is also the interaction place, recipes produce five kinds of effects: (i) new workers appear in the simulation context, either near to similar ones, or randomly distributed; (ii) firms hire workers and recipes modify workers and firms private matrixes; this is done accounting for both the availability of the labor production factor (firm side) and household income (workers side); (iii) firms make use of available labor production factors; (iv) firms either short of orders to be processed, or lacking adequate workers on the market, or being unable to deliver produced goods disappear from the economic scenario; (v) workers also disappears if unable to find a firm for prolonged time. Recipes are able to perform complex tasks, such as those described above, and are developed via computational steps. These steps can be interpreted as calls to code functions (methods of a Java class) invoked by the scrip language.

We can observe also the consequences of workers localizations in terms of system development (number of firms). When workers are clustered (by skill type), we observe a slow takeover of the economic system; by contrast when worker are randomly located we can observe a fast initial development of the economic system with an immediate stability of larger districts. This effect can be easily observed in the figs. 4 and 5, where the graphs of workers (stratum 1) and firms (stratum 4) are compared under the two situations.

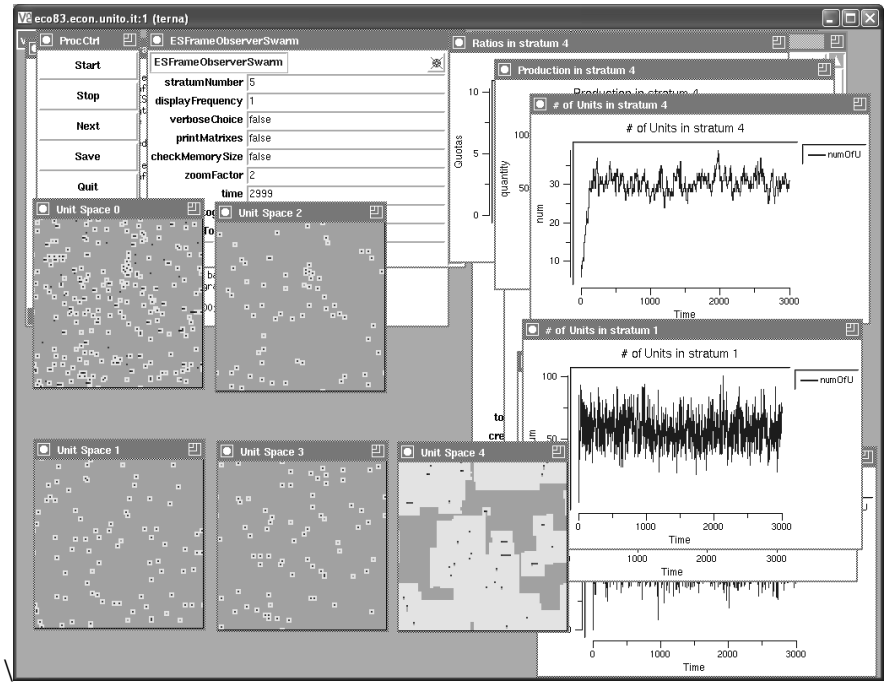

Fig. 5. Random diffusion of different skill workers, with a rather fast initial development of the economic system.

The orders coming from the market and their attribution to different firms determine cycles both in the workforce and the production. While in the relevant literature the frequency of cycles in the labor market is considered a challenging puzzle (see Hall, 2005; Shimer, 2005) in our model it is possible both to give a temporal dimension to cycles length and to give a theoretical interpretation of their occurrence. If three months of inactivity for real firms are assumed to be sufficient for default, the inactivity parameter we use in the model allows us to find the temporal dimension of our simulations. For instance, when setting the inactivity default parameter to 15 turns, a cycle of 1200 periods corresponds to about 20 years of actual time; in this span of we could observe (Fig. 5) about 8 turns and this is not too far from the empirical evidence.

We underline that all the scripting mechanism necessary to obtain this complex result is very similar to those of the appendix, plus a few small Java codes, specialized to solve the computational step cited above. In the distribution package of jESOF, folder apps/workers_skills_firms, you can find all the necessary file to reproduce the experiment.

Future development

In perspective, the goal of the WSF model is that to discover what happens if we introduce a more consistent double behavior of workers and firms in a concrete cognitive perspective, also by evolving agents' rules through soft computing techniques such as

classifier systems or neural networks, using in this case the Cross Targets technique (Terna, 2000), summarized at web.econ.unito.it/terna/ct-era/ct-era.html.

The WSF model can be improved introducing a population of banks, again with both a realization based on quasi-independent behavior and one on a structured cognitive solution.

Appendix: jESOF programming

Always at http://web.econ.unito.it/terna/jes/ look for the last version of jESOF, named in the form jesopenfoundation-x.y.zz.tar.gz; inside the distribution file you will find a folder apps/tutorial/ step3b_the_predators with the file running the example reported before. The programming code of the application is here reported in the spreadsheet of figs. 6, 7 and 8, respectively for grass, rabbits and foxes.

Look at http://web.econ.unito.it/terna/jes_files/ for the latest *How to use jES_O_F* file related to the coding system used here; the captions of Figg. 6 and 7 introduce some explanation for the examples of the script language used by jESOF.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	#	Recipes	;										
2		grassGeneration	1	c	1201	2	0	0	1	s	0	;	
3		utility	100	c	1100	3	0	1	2	1	s	0	;

Fig. 6. Coding system for grass: in col. C there is the number of the order containing the recipe to be executed; as an example, order 1 generates grass using a computational step invoked by 'c 1201'; the subsequent columns contain: the parameters of the step (which uses two matrixes with id 0 and 0, so twice the same matrix) and then the code of the unit (1, grass) able to launch the computation with a simulation time of seconds 0 (no delay).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	#	Recipes	;												
2		preysEating	1	1001	s	0	c	1202	2	1	1	1	s	0	;
3		timeStep	23	c	1299	2	1	1	1001	s	0	;			
4		preysReproducing	4	c	1206	2	1	1	1001	s	0	;			

Fig. 7. Coding system for rabbits: again in col. C we have the number of the order containing the recipe to be executed; order 1 tells to the preys (code 1001 in '1001 s 0', acting without delay) to look for grass (code 1 in '1 s 0') and to the grass to cancel itself when eaten and to inform the preys that they have eaten, increasing their food stocks; all these actions are done via the computational step 1202 (which uses two matrixes with id 1 and 1, so twice the same matrix).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	#	Recipes	;												
2		predatorsEating	1	2001	s	0	c	1202	2	2	2	1001	s	0	;
3		timeStep	23	c	1299	2	2	2	2001	s	0	;			
4		predatorsReproducing	4	c	1206	2	2	2	2001	s	0	;			

Fig 8. Coding system for foxes.

To study the rules of this script language please refer to the "How to use jES O F" to document cited above.

Langton ants (Steward, 1994) application is another significant example of the flexibility of jESOF; you can find the code in the folder apps/LangtonAnts.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	#	Recipes	;										
2		antMoving	1	c	1301	2	0	0	1	s	1	;	
3		utility	100	c	1100	3	0	1	2	1	s	0	;
4		addingAnts	2	c	1104	2	1	1	1	s	0	;	
5		addingMarks	3	c	1104	2	1	1	1001	s	0	;	

Fig. 9. Coding system for Langton's ants.

With a few lines of scripting code in jESOF (Fig. 9) and the use of the related Java computational steps, we can easily obtain the classical result of Fig. 10, here with more than one ant simultaneously acting.

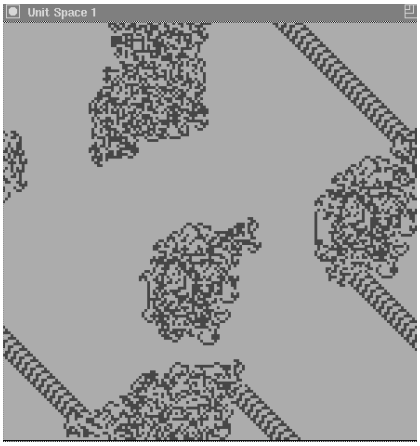

Fig. 10. The typical Langton's ants behavior reproduced with jESOF.

References

AXTELL, R. (2000), *Why agents? On the varied motivations for agent computing in the social sciences*. Forthcoming. Center on Social and Economic Dynamics, Working Paper No. 17, www.brook.edu/es/dynamics.

BORSHCHEV A. AND FILIPPOV A. (2004), *From System Dynamics and Discrete Event to Practical Agent Based Modeling: Reasons, Techniques, Tools*, on line at http://www.xjtek.com/files/papers/fromsystemdynamics2004.pdf

BURTON R. (2001), Afterword, in A. Lomi e E.R. Larsen (a cura di), *Dynamics of Organizations – Computational Modeling and Organization Theories*. Menlo Park, CA, AAAI Press / The MIT Press.

GILBERT N. AND TERNA P. (2000), How To Build and Use Agent-Based Models in Social Science. *Mind & Society*, vol.1, n.1, pp. 57-72.

GILBERT N., TROITZSCH K.G. (2005), *Simulation for the Social Scientist*. Buckingham, Open University Press.

HALL R.E. (2005), Employment Fluctuations with Equilibrium Wage Stickiness, *American Economic Review*, 1, XCV, pp.50-65.

SHIMER R. (2005), The Cyclical Behavior of Equilibrium Unemployment and Vacancies. *American Economic Review*, 1, XCV, pp.25-49.

STEWART I. (1994), The Ultimate in Anty-Particles. *Scientific American*, July 1994, independently reported at www.imsc.res.in/~sitabhra/teaching/cmp03/ian_stewart.html.

TERNA P. (2000), Economic Experiments with Swarm: a Neural Network Approach to the Self-Development of Consistency in Agents' Behavior, in F. Luna and B. Stefansson (eds.*), Economic Simulations in Swarm: Agent-Based Modelling and Object Oriented Programming*. Dordrecht and London, Kluwer Academic, pp.73-103.

TESFATSION L. (2001), Agent-Based Computational Economics: Growing Economies from the Bottom Up. *Artificial Life*, vol.8, n.1, pp.55-82.