Workers, Skills and Firms: a Simulation Model with *jES Open Foundation* [ϒ]

Pietro Terna

(September 2004)

Dipartimento di Scienze economiche e finanziarie G.Prato, Università di Torino, Italia
pietro.terna@unito.it

How would he look, to see his work so noble
Vilely bound up? (…)
Shakespeare, Winter's Tale

?. ABSTRACT

The simulation model is based on jES (java Enterprise Simulator) and jESevol, or "Evolutionary java Enterprise Simulator". Both the packages are based on Swarm.

jES is a large Swarm-based package aimed at building simulation models both of actual enterprises and of virtual ones. jESevol simulates systems of enterprises or production units in an evolutionary context, where new ones arise continuously and some of the old are dropped out.

The environment is a social space with metaphorical distances representing trustiness and cooperation among production units (the social capital). The production is represented by a sequence of orders; each order contains a recipe, i.e. the description of the sequence of activities to be done by several units to complete a specific production.

Two units can cooperate in the production process only if they are mutually visible in our social network. 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.

In a parallel way, other layers of the economic structure can evolve, always in an agent base perspective: banking system, employment structures, …

In this paper the focus is related to employment: when an enterprise produces a good, the sequence of the activities must match the presence of working units with the required skills. In this context a fine grain description of the steps in the recipes is needed.

Adequate labor units can be lacking, or might simply be already hired, thus fostering the emergence of competition among production units in the hiring process

Products change over time; as a consequence, productions units and labor skills have to adapt continuously, with co-evolutionary effects.

[ϒ] related to jesopenfoundation-0.1.40

?. THE MODEL

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

We launch in each cycle the order to the units representing the workers of reproducing near the copied unit, via a computational step[2]; unit in the workers' stratum (stratum = 0) are also created via the ordinary process of unit creation set in jesopenfoundation.scm.

We launch also three recipes[3] ordering to all the units with the same production phase of the unit receiving the order (which is sent to three units, having phase 1, 2 and 3) to display themselves on a specialized stratum: unit able in doing phase 1 on stratum 1, etc. (remembering that the first stratum is 0 numbered).

In stratum 4 we place a structure of firms, of type 1001, 1002, 1003 (production phases) and the we launch recipes of medium complexity using an OrderGenerator (in each other strata we have an OrderDistiller, using recipes and sequences, but in strata 2 and 3 with empty sequences). In the forth stratum units cooperates on the basis of their (increasing) intervisibility. New units are created with a low probability.

?. THE MODEL V.0

Initially we set as parameters for the stratum 4 the same values used for the jESevol simulation, case 5.3, adopting also that dimension for the rasters, but with only 3 types of units e 3 potential unit per type (it was 5 types with 2 potential units per type).

This ingenuous version, with the firm interacting and the worker evolving in a total independent way, is created only to test the model e to evaluate the behavior of the enterprises, with 1000 simulation ticks, in some way corresponding to 25 years of actual time (as showed in the presentation of jESevol to the 2004 SwarmFest).

It is frozen in apps/workers_skills_firms/workers_skills_firms_v_0/ folder.

---

[2] 1103 – this computational code creates a copy of the unit the order is in the model stratum of the original unit, near to it with the standard placement rule, with the probability set in position 0,0 of the first received matrix; the count of the created units is reported increasing by one the position 0,1 of the second received matrix; finally it changes the status to done.

[3] Using 1251, 1252 and 1253, referring to rows 1, 2 and 3 of matrix 0 (remember that the first row is row 0); these computational steps call internally the step 1250, which apply the step 1110 to a list of units.
Computational operations with code -1110 (a code for the jES Open Foundation extension): this computational code creates a copy of the unit the order is in, in the same x,y place of the copied unit, but in another model stratum, with the probability set in position 0,0 of the unique received matrix; the created unit is set in the stratum indicated in position 0,1 of the unique received matrix; the unit is created, with the above probability, if the content of the unit memory matrix in position 0,0 is positive (the actual position can be modified via the calling step, i.e. 1210 in tutorial step3c); if the destination position in the new stratum is occupied the new unit is not created, without any advice; the new unit is not created if the original one is in the destination stratum; the new unit has the same visibility of the old one; finally, the computational step changes the status to done.
c1110 is not normally utilized directly but via a special computational step (i.e. 1250 here) setting the position and content of the cell (<=0 or >0) indicated above; the memory matrix of the copied unit is shared among the original unit itself and its copies, avoiding the creation of new matrixes for the copies.
c1250: to have units displaying on another stratum we use the computational step 1250 (a code for the workers_skills_firms application), which acts via the 1110 code; this code, when received by one unit in a stratum via a recipe, acts vs. all the units of the same type (i.e., with the same production phase) in all strata (excluding those of the destination stratum), copying them on the destination stratum.
With c1251, c1252 and c1253 we apply c1250 to the cases of the units type 1, type 2 and type3.

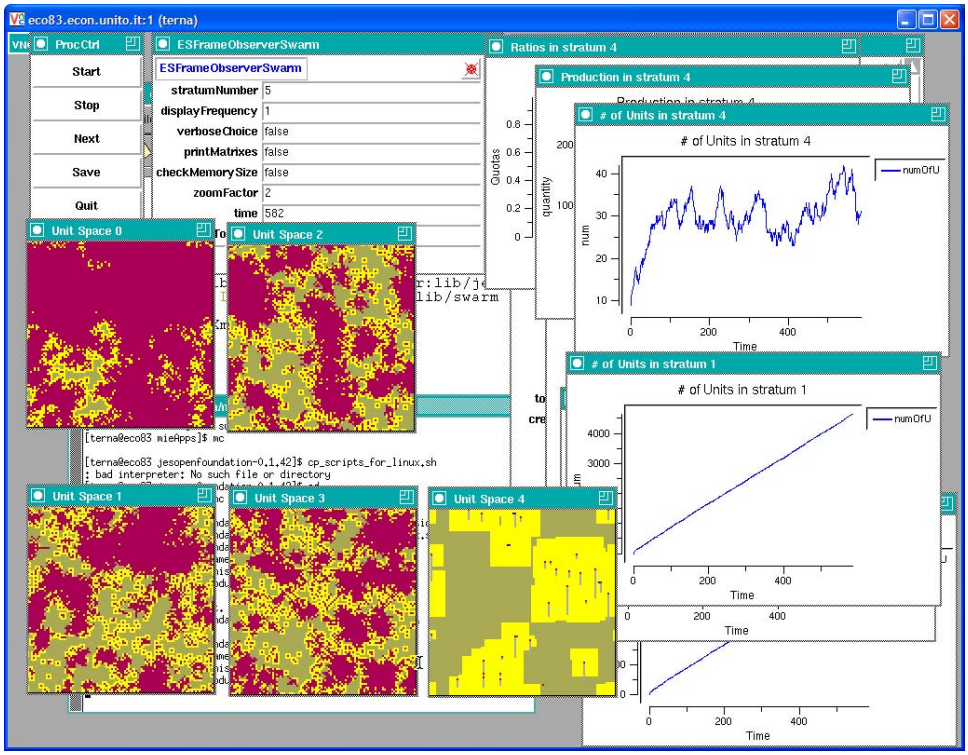

Figure ?. Version 0, no links between firms and workers

Matrix 0 is:

Probability for recipe 1, 2 or 3 in stratum 0, adding workers	count of the added workers of the 3 types
Probability for recipe 4 in stratum 0, displaying workers of type 1 in another stratum	the stratum where to display units of type 1
Probability for recipe 4 in stratum 0, displaying workers of type 2 in another stratum	the stratum where to display units of type 2
Probability for recipe 4 in stratum 0, displaying workers of type 3 in another stratum	the stratum where to display units of type 3

?. THE MODEL V.1 (WORKERS ARISING NEAR OLD ONES, IN A BALANCED WAY)

With the same set of parameters for the stratum 4 and the same rules for workers creation, now we introduce recipes – in the OrderDistiller of stratum 0 – going from units-firms of the three types (production phases 1001, 1002, 1003) to …

The copied units are created with the computational step 1111 (called via 1251, 1252, 1253 and 1250), acting as the step 1110, but with the use of an addendum (100 in our case) to modify the production phase of the unit created as copies.

The Matrix 0 now is:

Probability, for recipe 1, 2 or 3 in stratum 0, of adding workers	count of the added workers of the 3 types	
probability for recipe 4 in stratum 0, displaying workers of type 1 in another stratum	the stratum where to display units of type 1	amount (integer value of the addendum) to be added the production phase of the units created as a copy of an original unit
probability for recipe 4 in stratum 0, displaying workers of type 2 in another stratum	the stratum where to display units of type 2	amount (integer value of the addendum) to be added the production phase of the units created as a copy of an original unit
probability for recipe 4 in stratum 0, displaying workers of type 3 in another stratum	the stratum where to display units of type 3	amount (integer value of the addendum) to be added the production phase of the units created as a copy of an original unit

Using 1111 with the 100 addendum, we assign production phases 101, 102 and 103 to the copies of the units (workers) placed in strata 1, 2, and 3; the original (copied) units-workers have production codes 1, 2 and 3. In this way, recipes moving from units-firms (production phases 1001, 1002 and 1003) to units-workers (production phases 1, 2 and 3) can only find the original units of stratum 0 and are not interested to the copies created in strata 1, 2 and 3, where we want to represent the space diffusion of the professional skills.

Via the 1220[4] computational step we increase both the counter of the days-ticks a worker has been active (memory matrixes of the units-workers in position 0,0) and the counter of the days-ticks time units-workers the unit-firm has used (memory matrixes of the units-firms in position 0,0).

The new fifth row line of the Matrix 0, considering 7 columns (cols form 4th to 7th are empty in the preceding rows) is now:

addendum for position 0,0 (plus a0, b0 displacements) of the memory matrix of the unit the order is in (the worker)	addendum for position 0,0 (plus a1, b1 displacements) of the memory matrix of the unit the order come from (the firm)	A0	b0	A1	b1

[4] The 1220 computational step uses the 1120 one, modifying the coordinates of the memory matrixes where to apply the addenda, both in the unit the order is in and in the unit the order comes from. In this case the displacement in the memory matrixes is zero, but as a trace for future modifications, this feature is fully developed. The four coordinates (two for the memory matrix of the unit the order is in and two or the unit the order comes from, are in positions 0,2; 0,3; 0,4, 0,5 of the second received matrix

Now we clear the units-workers not employed or firms without employees via the C1297, C1298 and C1299 computational steps (codes for the workers-firms model), to decrease position 0,0 of the memory matrix of each unit by the content of position 0,5 of the second matrix and dropping the unit if the content o f 0,0 is < of position 0,2 of the second matrix; the usual trick of rd=k is used here; C1297, C1298 and C1299 act via C1199[5].

The last rows of Matrix 0 are similar: the first one for stratum 0 (the whole set of the workers); the second one for strata 1, 2, 3 (specialized for workers' skills); the last one for stratum 4 (the firms)

acting probability for time step (units in stratum 0, using time and, if any, moving away),	count of moved units (dropped)	Moving if the value in row, col of unit memory matrix is < the level set here	row of the memory matrix we refer to	col of the memory matrix we refer to	value to be added to the row,col position of unit memory matrix (zero if we refer to a copy of the original unit)

To introduce two options in hiring strategies of the firms, in recipe0Data/ we have both orderSequence.xls and orderSequence_with_high_hiring.xls (to be renamed orderSequence.xls to be used); in the first file we launch 10 orders of each one of the types 11, 12, 13 (hiring orders); in the second file we launch 20 of them.

With new unit probability 0.2, increase visibility 0.5, 3000 cycles and hiring with a moderated launch of recipes 11, 12, 13 (10 of each type per cycle), new workers with skills equal to that of their neighbors, we obtain:

[5] 1199 – this highly usable computational code, when received by one unit in a stratum, acts vs. all the units of the same type (i.e., with the same production phase) in all strata; actions: with the probability set in position 0,0 of the first received matrix, c1999 adds the value contained in 0,5 of the second received matrix to each unit memory matrix a row and column reported in pos 0,3 and 0,4 of the second matrix; if the resulting content is < the level contained in position 0,2 of the second matrix, the considered unit is dropped (if it is the unit the order is in, also the order is dropped), counting the dropped units in position 0,1 of the second matrix (obviously, first and second matrixes can be coincident).

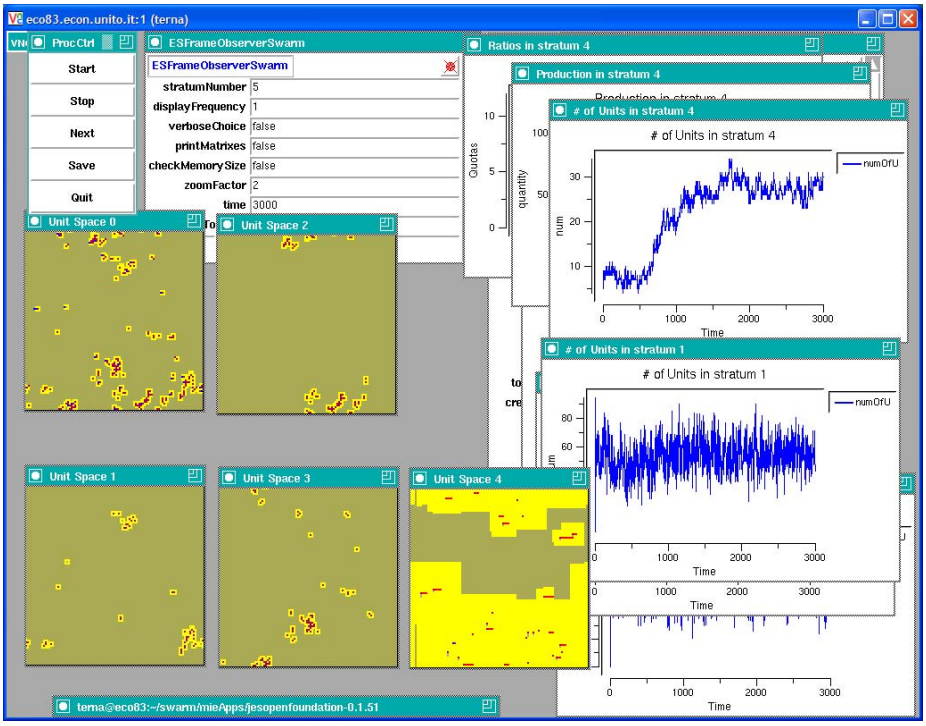

Figure ?. Version 1, new workers with skills equal to that of their neighbors

?. THE MODEL V.2 (WORKERS ARISING IN RANDOM POSITIONS, IN A BALANCED WAY)

Now we launch in each cycle the order to the units representing the workers asking for a random generation of workers, via a different computational step[6] if compared with v.0 and v.1 of the model; unit in the workers' stratum (stratum = 0) are also created via the ordinary process of unit creation set in jesopenfoundation.scm.

The first row of Matrix 0 now is:

Probability, for recipe 1, of adding workers with random skills	stratum in which to search and create the various types of unit-workers	count of the added workers of the 3 types

[6] 1201 – this computational step (a code for the workers_skills_firm model, but coming from the tutorial) acts via c1101 and creates a new random unit in the model stratum reported in position 0,1 of the first received matrix, with the probability set in position 0,0 of the same matrix and increases the position 0,2 of the second received matrix by 1, to count the created units; created units are chosen randomly (here of types 1, 2 or 3).

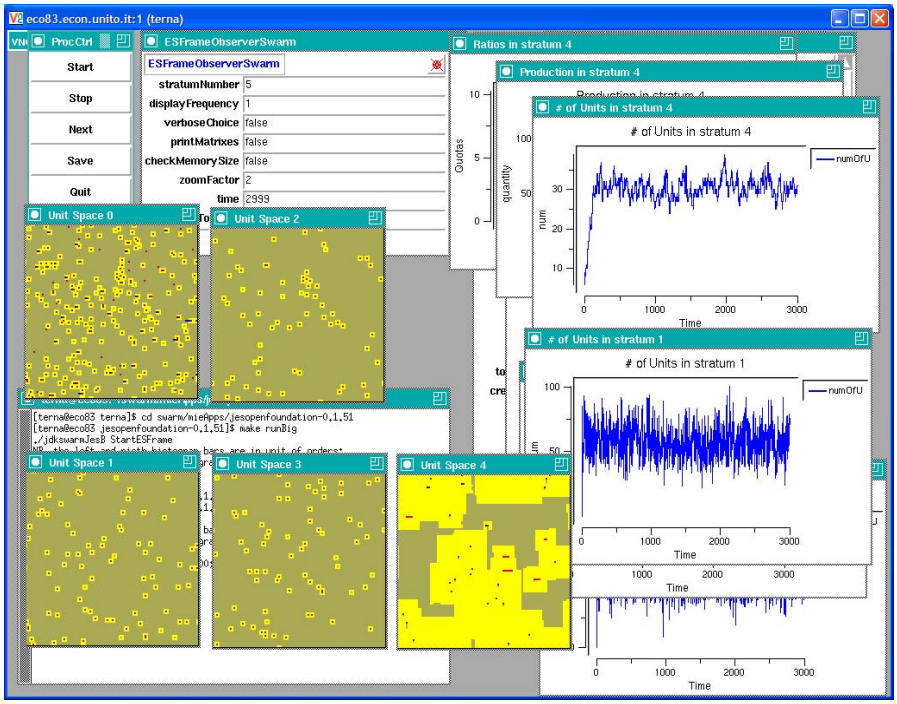

Figure ?. Version 2, new workers with skills randomly distributed in the stratum space

?. THE MODEL V.3 (WORKERS ARISING NEAR OLD ONES, IN AN UNBALANCED WAY)

The V.3 of the model is based on V.1, but in each cycle instead of launching 10 order of new workers creation for each type of skill (skills 1, 2 and 3), we launch 16 orders for skill 1 and 7 for each of the skills 2 and 3.

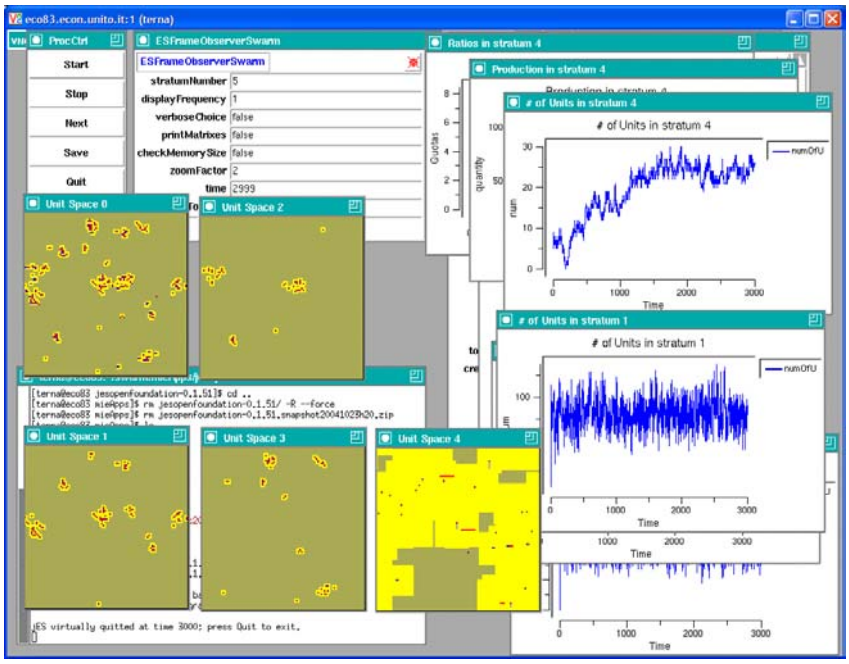

Figure ?. Version 3, new workers with skills equal to that of their neighbors, but arising in unequal quantities

?. THE MODEL V.4 (WORKERS ARISING IN RANDOM POSITIONS, IN AN UNBALANCED WAY)

The V.4 of the model is based on V.2, but in each cycle instead of creating 30 units-workers in random position extracting the agents from the three possible types at chance[7], we create in random position[8] 16 agents of type 1 and 7 of each of the other types (2 and 3).

The first row of Matrix 0 now is:

Probability for recipe 1, 2 or 3 of adding workers with 1, 2, 3 skills, in random position	count of the added workers of the 3 types	

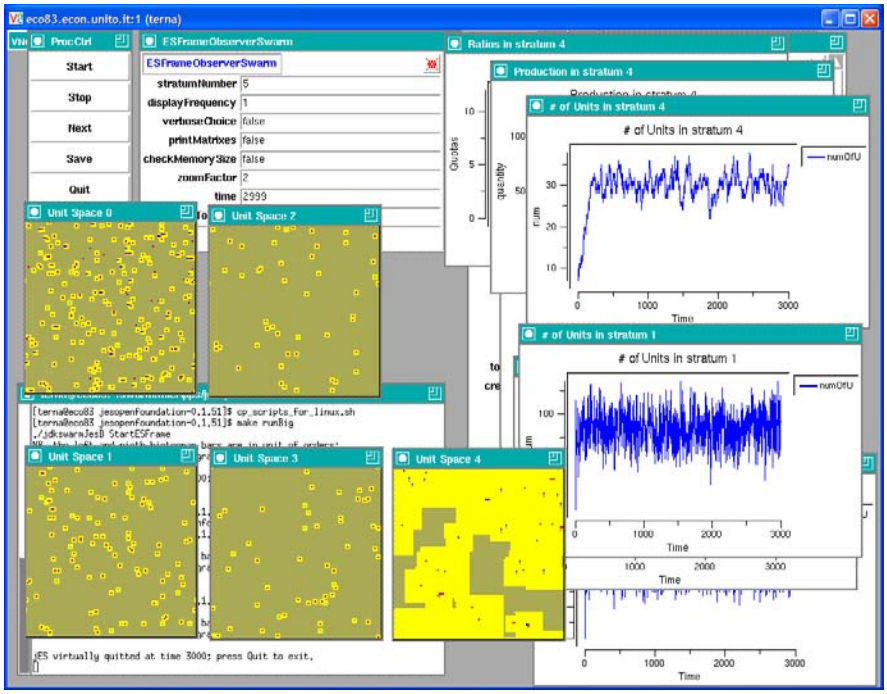

Figure ?. Version 4, new workers with skills randomly distributed in the stratum space, but arising in unequal quantities

[7] Using 1201 and 1101 steps.
[8] Using c1104 that duplicate in a random position the unit the order is in, in the same stratum.