DRAFT, please do not quote - v. 0.2.5.7

Workers, Skills and Firms: a Simulation Model with *jESOF*[ϒ]

Pietro Terna

(March 2005, rev. September 2006)

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

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

1. ABSTRACT

The simulation model is based on jES (java Enterprise Simulator) and on jESOF (jES Open Foundation), a jES generalization built to simulate multi-model frameworks of system of units or agents. 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 or actual) distances representing the inverse of trustiness and cooperation among production units. The social capital comes from the capability of the different units in working effectively in chains of exchanges, on the basis of reciprocal knowledge and confidence.

The production is represented by sequences of orders; each order contains a recipe, i.e. the description of the sequence of activities to be done by one or more units to complete a specific production. Orders at present are exogenously generate; *an important improvement of the model is that of relating the generation of order to the quantity of workers, seen as consumers.*

Two production units can cooperate in the production process only if they are mutually visible in their spaces. Production units that do not receive a sufficient quantity of orders, as well as the ones that cannot send the accomplished orders to other production units, disappear. New

---

[ϒ] The graphs and the maps reported in the figs. are generated with jesopenfoundation-0.1.71; successive modifications related to the copy of the visibility value, which was different from the original one due to an (int) vs. (double) problem and to an unnecessary initial application of the increasing step, do not modify the visibility of the copied workers, which has no increasing steps, being fixed at the value eight. So v.0 model results are not affected by these changes.
NB, figs. with a number followed by a letter are produced with version 0.1.72 or subsequent ones (at present 0.1.81).
Working to jesopenfoundation-0.1.72 I have noticed that the service recipes (look for a worker; duplicate yourself and so on) are, as obvious but I was not previously thinking at that aspect, modifying unsent queues (if the firm does not find workers) and inactivity value (a firm receiving and order of looking for a worker is considered as activated); now, in `ESFrameObserverSwarm` pane, we have the `wholeZeroTimeOrdersAreInvisible` options to avoid this problem. As above, v.0 model results are not affected by these changes, having no interactions between workers and firms; the service recipes sent to workers do not modify their behavior, being them not accounting for unsent products or inactivity days.

enterprises continuously arise, in the attempt of filling the holes of the production system. A complex structure emerges from this environment, with a poor number of surviving production units 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, workers presence, …

In this first step. the focus is related to employment: when an enterprise produces its outputs, it also needs the possibility of hiring working units with the required skills. In this context, a detailed description of the steps in the recipes is needed (see below, par.??).

Adequate labor units (workers) can be lacking: in this situation, if protracted in time, production units can disappear.

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

2. THE WSF (WORKERS, SKILLS, FIRMS) MODEL

The model has five strata: stratum 0 for the workers (all together); the workers are represented also in strata 1, 2 and 3 following their different (with three possibilities) 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 skill are coincident).

We have two sources of production orders:

- in stratum 0 we use the orderDistiller (i.e. an instance of the Java class OrderDistiller, able in reading files in folders named RecipeData? (where ? is the stratum number) to distill from recipe.xls file the recipes[2] contained in the orders to be launched following, fist of all, the sequence reported in the orderStartingSequence.xls file, and subsequently into the orderSequence.xls file, repeating indefinitely the sequence[3] contained in the second one.
- in stratum 4 we use the orderGenerator (i.e. an instance of the Java class OrderGenerator, able in generating orders containing random recipes) to generate production order following the scheme described the model panel of the stratum. In our case the production orders, one per cycle of time, contain max 5 steps each one with a

---

[2] Look at the content of the quoted file to discover the simple format used to report each recipe, here one per each row, but we can start anywhere; a row with a '#' sign in column 1 is a comment. Comment rows and recipes are concluded by a cell containing a ';' sign. Empty cells are allowed everywhere.

Recipes have a number, then a name, and finally one or more groups of three elements reporting the step to be done, the time unit ('s' meaning seconds in real world, not in the simulation) and the time necessary to do that step; each three element group can be preceded by a computational step, starting with a 'c' followed by the step id number, by the number of general matrixes used by the step and, finally, by the numerical codes identifying them; note that the id number can be repeated if we associate the same matrix to internal id of potentially different matrixes.

[3] Look at the content of the related files to discover the format of the sequences; a sequence starts everywhere in rows ('#' is used as in the previous note), with an id number and contain one or more three element groups, with the id of a recipe, the '*' sign and the number of recipes of this kind to be launched into the simulation. Sequences and comment lines are concluded by a cell containing a ';' sign.

Sequences are executed in the order they are written down and are execute one per each time cycle, only once if contained into the orderStartingSequence.xls file, and repeatedly if they are contained into the orderSequence.xls one.

max length of 2[4]. In the future we have to close the model, relating the launch of the production orders to the quantity of workers-consumers, via the use of an orderDistiller, without limitation about the number of production orders that can be launched in each time cycle.

Strata 1 to 3 formally use their own orderDistiller, but the related files in their RecipeData? are reporting no actions.

The orders launched from stratum 0 are related to the units of stratum 0 and 4; those launched from stratum 4 are related to the units of the same stratum. The id codes of the units and the related production phase are reported into the files unitBasicData.txt within the unitData? folders. In the strata 1 to 3 the unitBasicData.txt files are unused, but have to exist with some content. Those file are used also to create units as a background process, if activated; in strato 1 to 3 that process will be never activated, being the unit in those strata simply copies of those in stratum 0; they will never be used by our recipes.

The orderDistiller uses the phase codes of its recipes repertoire; if no unit accepts a specific step of an order, the order is lost, but if it has a computational step (CS) in it. In this case it is the CS to deal with the problem, holding the order or throwing away it. The orderGenerator of the stratum 4 uses the phase information of the related unitBasicData.txt file to create the recipes; if no unit accepts a specific step of an order and the step is the first one, the order is lost and has no contribution to the whole production; if the step is a successive one, the order stays in the list of the products that a specific unit is not able to send to another unit in the supply chain (unsent product list).

We can right click on our mouse, with the head of the mouse arrow sign on the little square of a unit in each stratum screen to discover its data with the unitProductionPhase datum.

In jESOF we have two kinds of matrixes: *general matrixes*, pertaining to the model as a whole; *unit matrixes*, which are specific to each unit CSs uses both the general matrixes described below, containing the parameters necessaries to computations, and unit matrixes.

3. THE MODEL V.0

3.1 WORKER GENERATION

Stratum 0 uses an OrderDistiller, with the recipes described below.

At the beginning of the simulation we have 50 potential workers per type, with initial generation probability 1.0 (so always a starting number of 50 workers per type) and a probability of unit generation per each time cycle (the background process of unit generation) of 0.9; the visibility is fixed at 8.

The initial order launching section, of the recipeData0/orderStartingSequence.xls file, contain only the launch of an order with the recipe 100; that recipe contains the computations step 1100, devoted to load in memory one general matrix (that with number 0), from the file

---

[4] Considering that we have three possible kinds of units, related to the correspondent production step, with id 1001, 1002, 1003 (and equals code for the production phases, valid recipes are:
1001 1001 1003 1002 1001 1001 (four steps, with length 2, 1, 1, 2);
1003 1002 1001 1001 1001 (four steps, with length 1, 1, 1, 2 or 1, 1, 2, 1);
1001 1001 1001 1001 1001 10011001 1001 1001 1001 (five steps, with length 2, 2, 2, 2, 2).

unitData0/memoryMatrixContents.txt using the unitData0/memoryMatrix.txt dimensions. That recipe is a "utility" one and it does not affect any unit; it is committed to a unit able in performing the phase "1", but only because each CS has to be committed to a unit. Note that if more than a unit can perform a task, the first one in the unit list is used; anyway, after each time cycle, the unit lists are randomly reordered.

Then we switch to the continuum section, following the recipeData0/orderSequence.xls file.

We launch in each cycle a given number of orders containing the recipe number 1, 2 and 3; those orders are addressed to the units representing the workers, commanding them to reproduce themselves (copying themselves) with a given probability (see fig. 1, row 0 and col. 0) near their position; the result is obtained via a CS[5] with code 1103 (internally -1103).

We launch 10 orders to workers of type 1, 10 to workers of type 2 and 10 to workers of type 3; workers are randomly reached by the orders (shuffleListsAtEachAssignment set to true) and the same worker can receive more than one order of reproducing itself (orders exceeding the first one will be executed in future steps). The code 1103 create the copied unit near to the original one with our standard placement rule, i.e. one of the eight cells near to the original position and in case the cell is not free we move following a straight line until a place is free.

probability for recipe 1, 2 or 3 in stratum 0, adding workers **(0.9)**	count of the added workers of the 3 types **(0.0)**
probability for recipe 4 in stratum 0, displaying workers of type 1 in another stratum **(1.0)**	the stratum where to display units of type 1 **(1)**
probability for recipe 5 in stratum 0, displaying workers of type 2 in another stratum **(1.0)**	the stratum where to display units of type 2 **(2)**
probability for recipe 6 in stratum 0, displaying workers of type 3 in another stratum **(1.0)**	the stratum where to display units of type 3 **(3)**

FIG. 1 – Matrix 0 in v.0 model.

Unit in the workers' stratum (stratum = 0) are also created via the ordinary process of unit creation set in jesopenfoundation.scm, with a new worker created in each cycle with probability=0.9, with random skill and random position (if the position randomly chosen is not free a new position is searched moving along a straight line in a random direction). This feature could also be switched off, simply choosing a 0.0 creation probability both in the model panel of the stratum 0 or correcting the value of #:newUnitGenerationP in the eSFrameModelSwarm0 block of the jesopenfoundation.scm file.

---

[5] 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 (i.e. one of the eight cells near to the original position and in case the cell is not free we move following a straight line until a place is free); the action is done 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.

After the double creation phase, we launch also three orders containing recipes 1251-3[6] ordering to all the units with the same production phase of the unit receiving the order (we send one order to a unit implementing the phase 1, one to a unit implementing the phase 2 and one to a unit implementing the phase 3) to display themselves on a specialized stratum: units able in doing phase 1 on stratum 1, etc. (remembering that the first stratum is numbered 0). It is absolutely not relevant which unit is receiving this order; when a unit receive it, the unit activate itself and all the similar units in all the strata, but that of the destination stratum expressed in the copying order.

3.2 ENTERPRISE STRATUM

In stratum 4 we have a system of firms, of the types 1001, 1002, 1003 (the code identifiy their production phases); we launch recipes of medium complexity using an OrderGenerator (in each other strata we have an OrderDistiller, using recipes and sequences; note: in strata 1, 2 and 3 with empty sequences).

In the fifth stratum (number 4) units cooperates on the basis of their (increasing) mutual visibility. Initial units are of the three types, potentially 3 per type, with creation probability 0.8. New units are created with probability 0.8 in each step, with random production phase, random position and initial visibility between 0 and 200. Visibility increases of 5 squares in each tick.

Initially we set as parameters for the stratum 4 the same values used for the jESevol simulation, case 5.3, adopting also the same dimensions for our toroidal world (the raster), but with only 3 types of units e 3 potential unit per type (in jESevol we had 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 and to evaluate the behavior of the enterprises, with 1,000 simulation ticks, in some way corresponding to 25 years of actual time (as showed in the presentation of jESevol at the 2004 SwarmFest). This consideration about the time scale is as follows: maxInactivity is 15 and maxUnsentProducts is 10; if we set both to 10, we have a situation of max ten ticks without work (maxInactivity) or of max ten ticks without delivery (maxUnsentProducts) which is related in the real world to a quarter

---

[6] 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 applies the step 1110 to a list of units (using the row displacement set by each one of the calling step, to inspect the memory matrix).

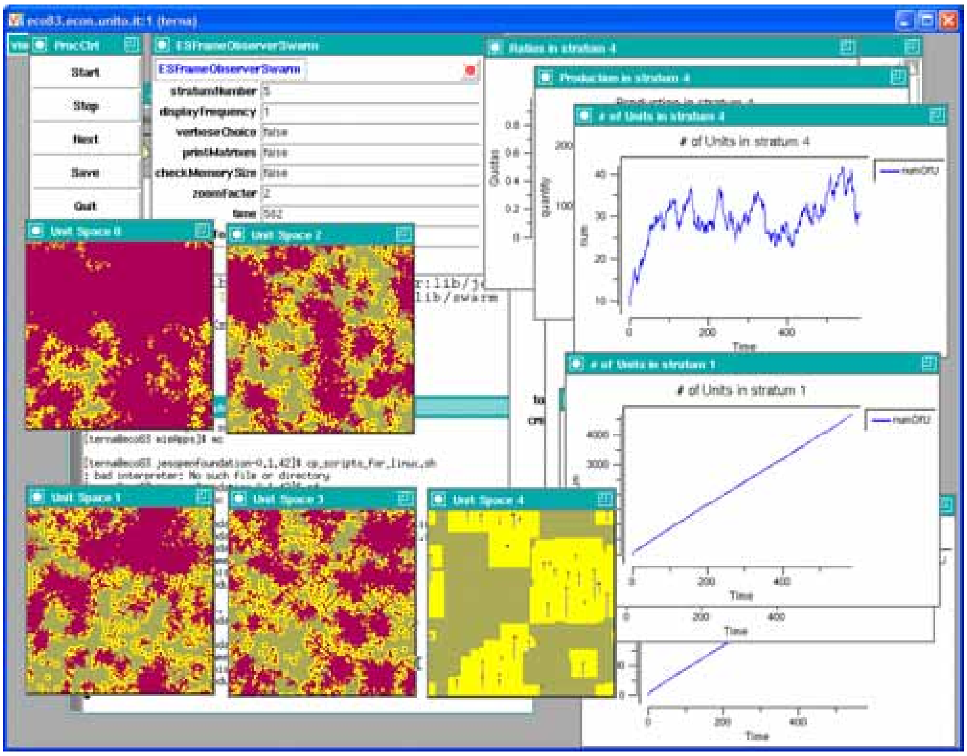
Computational operations with code -1110 (a code for the jES Open Foundation extension): this computational code creates a copy of the unit the order is in, in the same (x, y) place of the copied unit, but in another model stratum, with the probability set in position 0,0 of the unique received matrix; 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 of this information within the memory matrix of the unit can be modified via the calling step, e.g. 1210 in tutorial step3c; here it is effectively displaced of one column); 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.

The CS 1110 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.

1250: 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 the CSs 1251, 1252 and 1253 we apply the CS 1250 to the cases of the units type 1, type 2 and type 3.

of maximun reasonable survival of an enterprise without work or delivery. So 1,000 simulation ticks are considered equivalent to 25 years of real time. Then we relax the maxInactivity hypothesis to 15 ticks, considering as an example some intervention of the banking system and the simulation runs with a smoother behavior.



stratumNumber 5
displayFrequency 1
verboseChoice false
printMatrixes false
checkMemorySize false
zoomFactor 2
time 582

FIG. 2 – Version 0: no links between firms and workers. The results are the same also with wholeZeroTimeOrdersAreInvisibile = true.

The v.0 model produces the behavior reported in fig. 2 and it is frozen in apps/workers_skills_firms/workers_skills_firms_v_0/ folder.

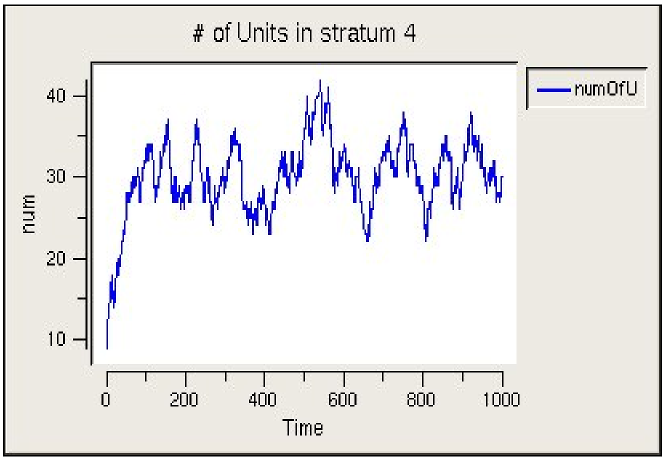


FIG. 3 – Version 0: number of workers in stratum 4 after 1,000 cycles, i.e. about 25 year in real time. The results are the same also with wholeZeroTimeOrdersAreInvisibile = true.

Running the model for 1,000 tick, the number of main economic cycles that we can recognize in fig. 3 is four; the first one is subdivided in three minor cycles. Globally, an highly realistic picture on a business cycle.

4. THE MODEL IN V.1 – THE INTERACTION BETWEEN WORKERS AND FIRMS (WORKERS ARISING NEAR OLD ONES, IN A BALANCED WAY, AS IN V.0)

With the same set of parameters for the stratum 4 and the same rules for workers creation, we introduce now a set of orders – launched by the OrderDistiller of stratum 0 – going from units-firms of the three types (with the production phases 1001, 1002, 1003) to the unit-workers, to hire them.

The copy of the workers in the strata 1, 2 and 3 (done merely for display reasons) are now created with the CS 1111 (called via 1251, 1252, 1253 and 1250), acting as the step 1110, but with the use of an addendum (100 in our case, look at fig. 4, row 1 and col. 2, always starting form row 0 and col. 0) to modify the production phase of the unit created as a copy. In this way 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. As a consequence, 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 represent the space diffusion of the professional skills.

In recipeData0/orderSequence.xls we can verify that we launch now, at each tick, 10 orders containing the recipe number 11 directed to firms with the first production phase (1001): the orders then move to workers with a consistent skill (production phase number 1); the recipes move from the firms to the workers immediately (these are steps with zero time execution; a firm sees a worker if the visibility areas both of the firm and of the worker are overlapping at least by one square. We do the same for phase 1002 with workers 2 and 1003 with 3, using the orders containing the recipes number 12 and 13, 10 at each tick.

probability for recipe 1, 2 or 3 in stratum 0, adding workers **(0.9)**	count of the added workers of the 3 types **(0.0)**	
probability for recipe 4 in stratum 0, displaying workers of type 1 in another stratum **(1.0)**	the stratum where to display units of type 1 **(1)**	amount (integer value of the addendum) to be added the production phase of the units created as a copy of an original unit **(100)**
probability for recipe 5 in stratum 0, displaying workers of type 2 in another stratum **(1.0)**	the stratum where to display units of type 2 **(2)**	amount (integer value of the addendum) to be added the production phase of the units created as a copy of an original unit **(100)**
probability for recipe 6 in stratum 0, displaying workers of type 3 in another stratum **(1.0)**	The stratum where to display units of type 3 **(3)**	amount (integer value of the addendum) to be added the production phase of the units created as a copy of an original unit **(100)**

FIG. 4 – Matrix 0 in v.1 model.

Via the 1220[7] CS contained in the orders above (those with recipes 11 or 12 or 13), we increase both the counter of the time cycles a worker has been active (memory matrixes of the

---

[7] 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

units-workers in position 0,0) and the counter of workers the unit-firm has hired, each per one time cycle (memory matrixes of the units-firms in position 0,0).

addendum for position 0,0 (plus the a0, b0 displacements) of the memory matrix of the unit the order is in (the worker) **(1.0)**	addendum for position 0,0 (plus the a1, b1 displacements) of the memory matrix of the unit the order come from (the firm) **(1.0)**	a0 **(0.0)**	b0 **(0.0)**	a1 **(0.0)**	b1 **(0.0)**

FIG. 5 – Matrix 0 in v.1 model, a fifth row.

Both workers and firms can accumulate more than one unit of work per time cycle or tick (the worker is overworking and the firm is over-hiring workers). Worker always accept work, also if just working (simply think about the worker as an household). We have anyway to introduce here, as a future improvement, a decreasing productivity, falling to zero when the worker is over-occupied.

The new added fifth row line of the Matrix 0, shown in fig. 5, has 6 columns (cols from 4th to 6th are empty in the preceding rows) and contains the addenda to be used to increase the contents of the above mentioned matrixes both of the unit-workers (if hired) and of the units-firm if hiring.

Now we manage consumption for the units-workers (in terms of unit of work accumulated when hired) and work capability utilization for the unit-firms (in terms of unit of work in some way “accumulated” hiring workers in each cycle).

The operation is done via the 1297, 1298 and 1299 CSs (codes for the workers-skills.-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 of 0,0 is less than of the threshold in position 0,2 of the second matrix; the usual trick of rd=k is used here.

probability of the correction (of the value whose id are in cols. 3 and 4 here) and, if necessary, of unit dropping **(1.0)**	count of dropped units **(0.0)**	dropping if the value in row, col. of unit memory matrix is less than the threshold set here **(-5.0)**	row of the memory matrix we refer to **(0.0)**	col. of the memory matrix we refer to **(0.0)**	value to be added to the row, col. position of unit memory matrix **(-1.0)**

FIG. 6 – Matrix 0 in v.1 model, a sixth row, for unit-workers in stratum 0.

The CS 1297 is applied to units doing phase 1 or 2 or 3 (workers) via the orders containing the recipes 21, 22 and 23. We launch only a copy of each order, due to the way the internally recalled CS 1199[8] acts, propagating the order to al the units able in doing the same production phase or step. We are referring here to the sixth row of the matrix 0, as reported in fig. 6, so workers are dropped (to reappear in another place, as moving away, due to the process of workers’ creation) when their accumulation of working cycles falls under -5.0, being the accumulation value diminished of 1.0 in each cycle and increased of the same amount for each hiring event (remember that more than one hiring event can occur in each cycle).

---

[8] 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, 1999 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 less than the threshold 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).

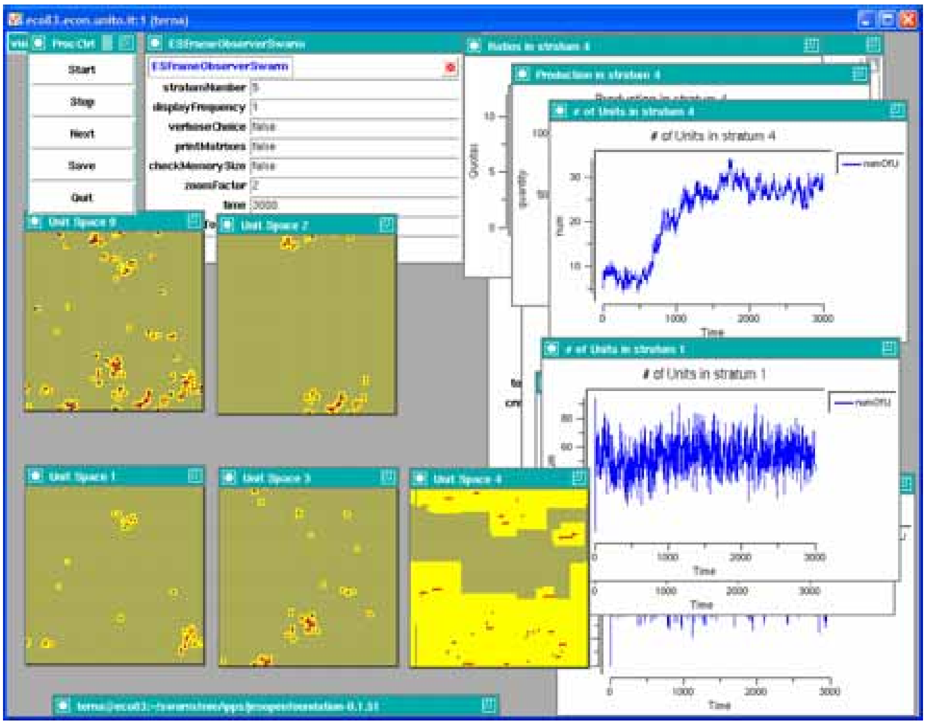
probability of the correction (of the value whose id are in cols. 3 and 4 here) and, if necessary, of unit dropping **(1.0)**	count of dropped units **(0.0)**	dropping if the value in row, col. of unit memory matrix is less than the threshold set here **(-5.0)**	row of the memory matrix we refer to **(0.0)**	col. of the memory matrix we refer to **(0.0)**	value to be added to the row, col. position of unit memory matrix **(0.0)**

FIG. 7 – Matrix 0 in v.1 model, a seventh row, for unit-workers in stratum 1 or 2 or 3.

The CS 1298 is applied to units doing phase 101 or 102 or 103 (worker copies) via the orders containing the recipes 31, 32 and 33. We launch only a copy of each order, as above. We are referring here to the seventh row of the matrix 0, as reported in fig. 7, so the copies of workers are dropped (to reappear in another place, as moving away, due to the process of workers' creation) when their accumulation of working cycles falls under -5.0. The correction used here, as shown in the last column, is 0.0, being the correction made upon the original unit and being unique the memory matrix of the original unit and of its copies. If the original unit is dropped, the memory matrix is anyway conserved until one of the copies exists.

probability of the correction (of the value whose id are in cols. 3 and 4 here) and, if necessary, of unit dropping **(1.0)**	count of dropped units **(0.0)**	dropping if the value in row, col. of unit memory matrix is less than the threshold set here **(-10.0)**	row of the memory matrix we refer to **(0.0)**	col. of the memory matrix we refer to **(0.0)**	value to be added to the row, col. position of unit memory matrix **(-1.0)**

FIG. 8 – Matrix 0 in v.1 model, a eighth row, for unit-firms in stratum 4.



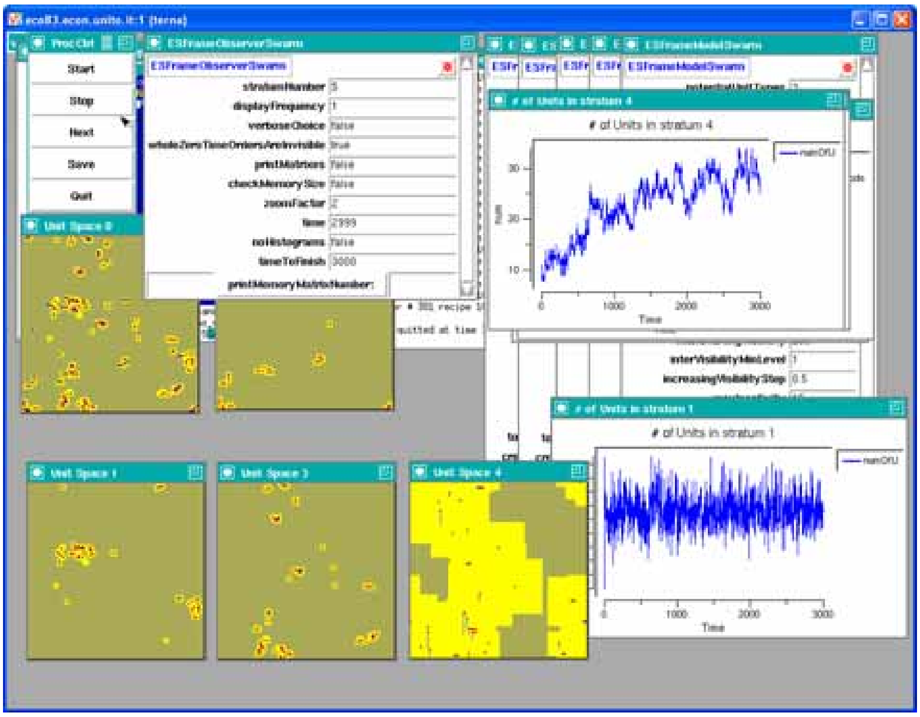
stratumNumber 5
displayFrequency 1
verboseChoice false
printMatrixes false
checkMemorySize false
zoomFactor 2
time 3000

FIG. 9 – Version 1, new workers with skills equal to that of their neighbors.

The CS 1299 is applied to units doing phase 1001 or 1002 or 1003 (firms) via the orders containing the recipes 41, 42 and 43. We launch only a copy of each order, as above. We are referring here to the eighth row of the matrix 0, as reported in fig. 8, so firms are dropped when their accumulation of working capabilities falls under -10.0, being the accumulation value diminished of 1.0 in each cycle and increased of the same amount for each hiring event (remember that more than one hiring event can occur in each cycle).

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, 3,000 cycles and hiring with a moderated launch of recipes 11, 12, 13 (10 of each type per cycle), new workers appearing near the previous ones, with skills equal to that of their neighbors, we obtain the results of fig. 9 or of fig. 9b with wholeZeroTimeOrdersAreInvisibile = true.



strataNumber 5
displayFrequency 1
verboseChoice false
wholeZeroTimeOrdersAreInvisible true
printMatrixes false
checkMemorySize false
zoomFactor 2
time 2999
noHistograms false
timeToFinish 3000
printMemoryMatrixNumber:
interVisibilityMinLevel 1
increasingVisibilityStep 0.5

FIG. 9b – Version 1, new workers with skills equal to that of their neighbors, with wholeZeroTimeOrdersAreInvisibile = true.

We observe a slow process of development, requiring 25 years (we use here the same time estimations of par. 3.2); in the successive 50 years we have about 4 major economic cycle, i.e. an absolutely reasonable situation, more stable than that of fig. 2[9].

---

[9] With other random number we can have similar results, also with greater evidence of the development period; as an example, see the fig. 9c here, generated with the WSFB v.0 model (workers_skills_firms_orders) with a p=0 probability of doubling firms in stratum 5; the difference with the model used in fig. 9b is that of the generation of more random numbers, so with a different series of random numbers.

5. THE MODEL IN V.2 – THE INTERACTION BETWEEN WORKERS AND FIRMS (WORKERS ARISING IN RANDOM POSITIONS, IN A BALANCED WAY)

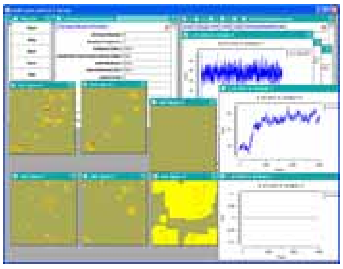


FIG. 9c – Version 1, but generated with WSFB v.0 model, with p=0.

In fig. 9d we run the WSF v.1 model with WSFB v.0 model having eliminated the launch of the orders related to the doubling process. Anyway random nyumer series changes because in stratum 5 (the sixth one) we have to chose to create or not new unit (p=0, but the randomizer is called).

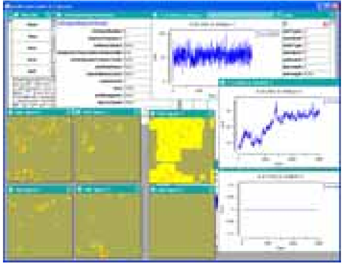


FIG. 9d – Version 1, but generated with WSFB v.0 model, eliminating doubling orders.

Finally, in fig. 9e we have exactly the results of fig. 9b (but from WSFB instead from WSF) having eliminated the creation of straum 5 (the sixth one) via the observer probe.

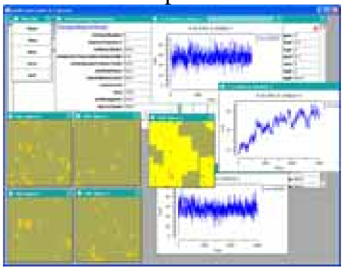
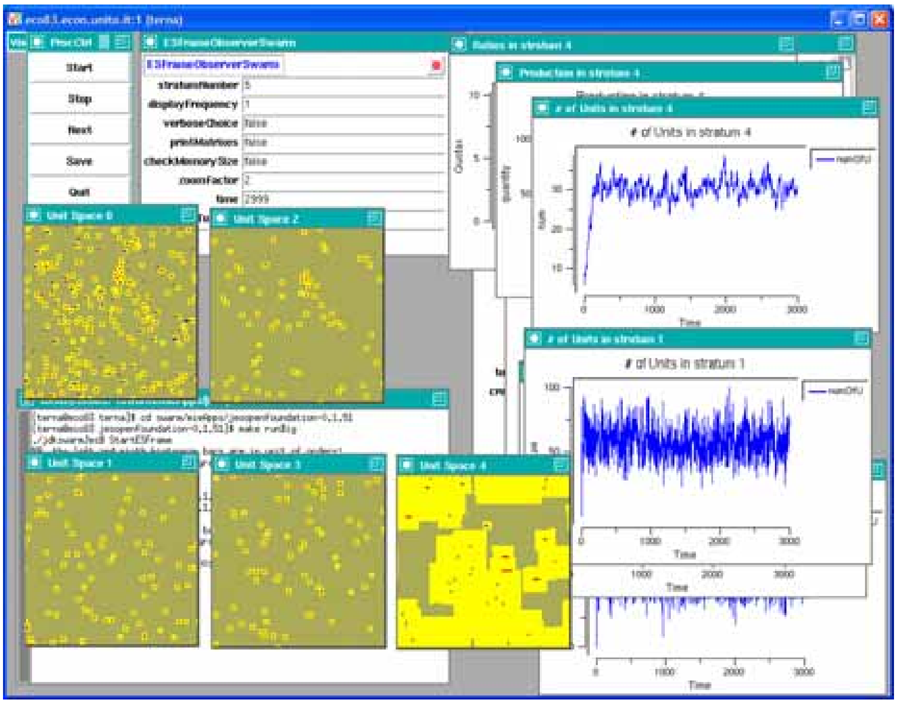
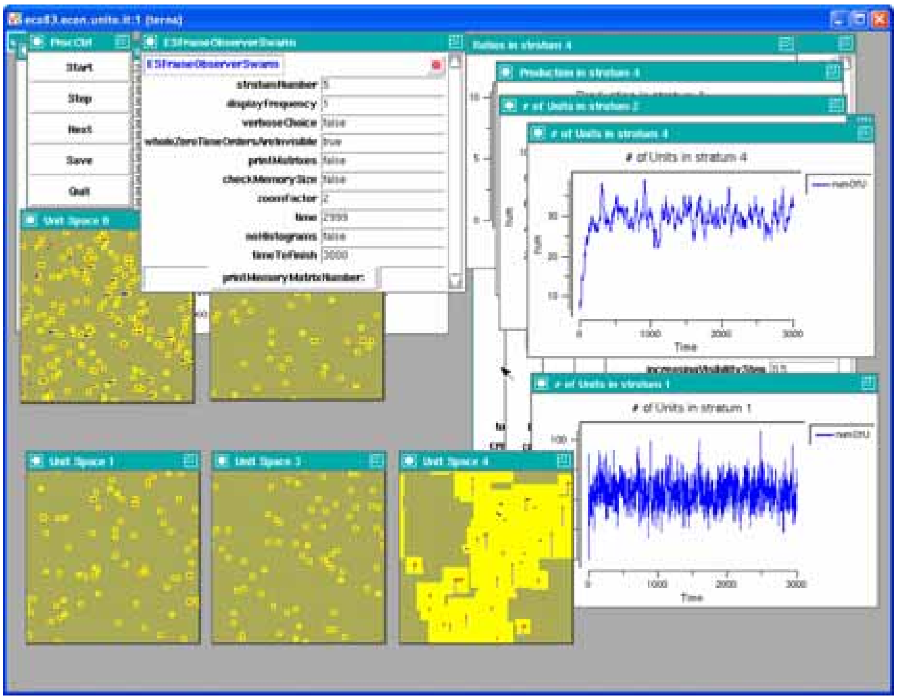


FIG. 9e – Version 1, but generated with WSFB v.0 model, eliminating the sixth stratum (# 5).



stratumNumber 5
displayFrequency 1
verboseChoice false
printMatrixes false
checkMemorySize false
zoomFactor 2
time 2999

FIG. 10 – Version 2, new workers with skills randomly distributed in the stratum space.



stratumNumber 5
displayFrequency 1
verboseChoice false
wholeZeroTimeOrdersAreInvisibile true
printMatrixes false
checkMemorySize false
zoomFactor 2
time 2999
noHistograms false
timeToFinish 3000
printMemoryMatrixNumber:

FIG. 10b – Version 2, new workers with skills randomly distributed in the stratum space, with wholeZeroTimeOrdersAreInvisibile = true.

As in the other version we launch in each cycle a given number (30 in our case, as in v. 1) of orders addressed to the units representing the workers, commanding them to reproduce themselves (copying themselves) with a given probability (see fig. 11, row 0 and col. 0) in a random position; the result is obtained with a unique recipe (number 1) calling the CS 1201[10]. It is not relevant that the orders containing the recipe 1 are all directed to a unit of type 1, being the new units created with random phases. Data are reported in fig. 10 and 10b.

prob., for recipe 1, of adding workers with random skills **(0.9)**	stratum in which to search and create the various types of unit-workers **(0)**	count of the added workers of the 3 types **(0.0)**

FIG. 11 – Matrix 0 in v.2 model, first row.

Unit in the workers' stratum (stratum = 0) are also created via the ordinary process of unit creation set in jesopenfoundation.scm, with a new worker created in each cycle with probability=0.9, automatically with random skill and random position (if the position randomly chosen is not free a new position is searched moving along a straight line in a random direction).

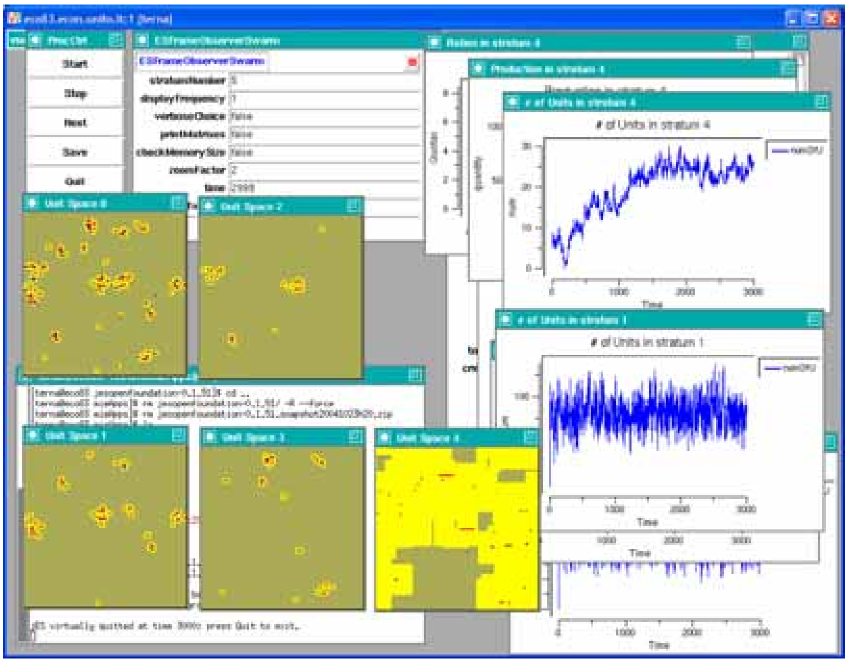
The results of fig. 10 and 10b show a fast development process (about five year in the scale above) and a limited number of economic cycle (about seven or eight) in 45 years, which is less than in the reality of the business cycle. Pay attention: the realistic model is the first one, with specialized workers arising near the existing one, via existing schools and local cultural heritage. In other words, via the structure of local districts, not useful in a growth perspective. The random spread of the skills, on the contrary, is also useful to stabilize the cycle.

These results are robust to an attempt of generating the presence of the workers in an unbalanced way, as reported in pars. 6 and 7.

---

[10] 1201 – this computational step (a code for the workers-skills-firms model, but coming from the tutorial) acts via the CS 1101 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 unit type or phase are chosen randomly (here of the types 1, 2 or 3).

6. THE MODEL IN V.3 – THE INTERACTION BETWEEN WORKERS AND FIRMS (WORKERS ARISING NEAR OLD ONES, IN AN UNBALANCED WAY)



stratumNumber 5
displayFrequency 1
verboseChoice false
printMatrixes false
checkMemorySize false
zoomFactor 2
time 2999

FIG. 11 – Version 3, new workers with skills equal to that of their neighbors, but arising in unequal quantities. The model had to be rerun with `wholeZeroTimeOrdersAreInvisibile = true`, but without expectations of significative differences.

The version 3 of the model is based on version 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.

As we can see in fig. 10, the growth process reported here is lightly slower than that shown in fig. 9.

7. THE MODEL IN V.4 – THE INTERACTION BETWEEN WORKERS AND FIRMS (WORKERS ARISING IN RANDOM POSITIONS, IN AN UNBALANCED WAY)

The version 4 of the model is based on version 2, but in each cycle, instead of creating 30 units-workers in random position extracting the agents from the three possible types in a random way[11], we create, always in random positions[12], 16 agents of type 1 and 7 of each of the other types (2 and 3).

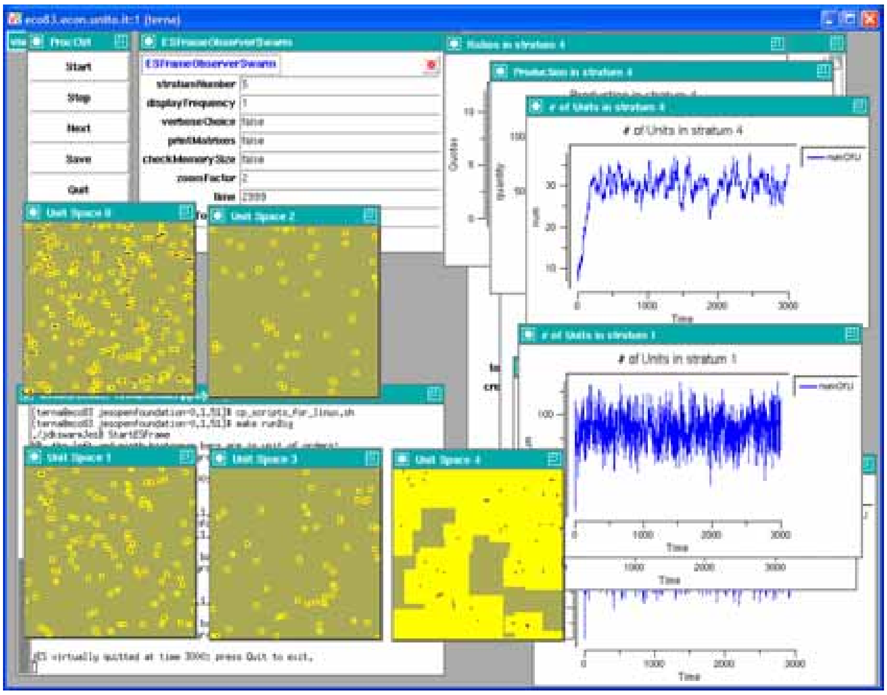
probability for recipe 1, 2 or 3 of adding workers with 1, 2, 3 skills, in random position **(0.9)**	count of the added workers of the 3 types **(0.0)**	

FIG. 12 – Matrix 0 in v.2 model, first row.

---

[11] Using `1201` and `1101` steps.

[12] Using the `1104` CS that duplicate in a random position the unit the order is in, in the same stratum.



stratumNumber 5
displayFrequency 1
verboseChoice false
printMatrixes false
checkMemorySize false
zoomFactor 2
time 2999

FIG. 13 – Version 4, new workers with skills randomly distributed in the stratum space, but arising in unequal quantities. The model had to be rerun with wholeZeroTimeOrdersAreInvisibile = true, but without expectations of significative differences.

As we can see in fig. 13, the development process is fast as in fig. 10.