Working Paper

A New Method for Structural Simulation

Carlos Domingo, Marta Sananes, Giorgio Tonella, Fernanda Sarmiento

WP-95-54
June 1995

A New Method for Structural Simulation

Carlos Domingo, Marta Sananes,
Giorgio Tonella, Fernanda Sarmiento

WP-95-54
June 1995

IIASA International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 807 □ Fax: +43 2236 71313 □ E-Mail: info@iiasa.ac.at

Abstract

In this paper structural change is defined and a tool to simulate structural changes is introduced which consists of a new simulation language which allows to deal separately with quantitative changes and structural qualitative changes. Two strategies of structural simulation are described. In the first one, the user defines the possible structures and conditions of change. In this case, the simulation process finds the structural paths through successive structures. In the second strategy, the structures are generated by the simulation process based on the model of creative thinking proposed by Poincaré and Hadamard. AI and genetic programming techniques are used to implement the model. A simple example is given to illustrate the method of the second strategy.

Keywords: *simulation, structural simulation, artificial intelligence, simulation language, variable structure.*

A New Method for Structural Simulation[1]

Carlos Domingo[2], *Marta Sananes*[2], *Giorgio Tonella*[3], *Fernanda Sarmiento*[4]

Creativity occurs not just upward from the bottom,
with new forms arising from less complex systems by spontaneous jumps;
it also proceeds downward from top,
through the creative activity of higher level fields.
Rupert Shelgrake. *The Rebirth of Nature.*

1 Structural Simulation through Predefined Structures

An important class of problems may be solved or explored by simulation. However, when the system to be simulated undergoes structural changes, the simulation is still possible but, unless the changes are trivial ones, the difficulties of implementation cause confusion in the design and the simulation becomes a complex problem. In this case, it is necessary to develop new methods, such as the ones presented in this paper. The basic idea of this new approach is to adopt a clear definition of structural changes and to develop programming techniques to deal separately with the quantitative changes of the variables, usually found in common simulation, and the qualitative changes which are managed by structural simulation.

The *structure* of a system is determined by:

- The components of the system (subsystems) and their connections (information interchanges).
- The parameters that determine the behavior of the components and their connections.

[1]A preliminary version of this working paper under the title *Problem Solving by Structural Simulation* was sent to the IASTED International Conference Modelling and Simulation held in Pittsburgh, USA, 27-29 April 1995.

[2]Instituto de Estadística Aplicada y Computación, FACES, ULA, Mérida, Venezuela; email: carlosd@faces.ula.ve

[3]Centro de Simulación y Modelos, Facultad de Ingeniería, ULA, Mérida, Venezuela: email: tonella@ing.ula.ve; at IIASA on sabbatical leave.

[4]CDCHT Training Program Universidad de los Andes, Mérida, Venezuela

A *structural change* is any change involving adding or removing of components, changing their connections or changing the values of the parameters that alter the behavior [6] [7].

There are many ways to manage the simulation of structural changes [12] [16]. However, none of them keep usual simulation separate from structural simulation. GLIDER is a new simulation language developed at the Universidad de los Andes for common and structural simulation, and is used in this research as the basic tool for simulating structural change. In this language [3] [5], the system components are represented by objects called *nodes* which have data and methods to describe the information processing capabilities of the subsystems represented. The algorithms and data description are coded into a basic general purpose language enriched by a set of simulation facilitating instructions, procedures and functions. The nodes may relate not only through common (global) data structures and files (as is typical in objects and procedures) but also by *messages* that the nodes can send to one another. Messages are objects that carry information through all types of variables and references to the procedures. They may, among other things, act as transactions or traveling entities such as those used in classical simulation languages. The nodes and their relations constitute a network or oriented graph which represents the system structure. A node may be activated by itself or by other nodes.

Although the programmer has the freedom to program the nodes, she or he can use some predefined nodes that have built-in facilities to handle messages and particular ways of being activated. They are suited to simulate different processes and subsystem types: generation and destruction of messages, gates, resources, selection and routing of messages, continuous processes, and discrete event processes.

The language is able to handle messages, and it has the graphical and statistical facilities included in most simulation languages. It has been tested in many practical applications and teaching courses in simulation [4].

The language is well suited to represent the structures defined above: subsystems are represented by nodes; global variables and messages describe the relationships. The network of nodes and some parameters define a structure. Different structures may coexist in the same program as disjointed or overlapping parts of the total network. These parts may be activated one after the other when the conditions of structural change are fulfilled. Nodes of similar types may be introduced during the simulation by means of object creation or by use of subscripted nodes. Some nodes can take charge of structural changes by periodically monitoring the conditions for structural change and, if the change is required, they can activate new nodes, deactivate others and change parameter values. With this design the structural changes are separated from ordinary quantitative changes in the program because the latter changes the task of different nodes. When used by enterprises or institutions the nodes for the two types of changes are usually designed by people from two different levels of management.

The problem of structural simulation was solved using two different strategies. The first uses predefined structures. In this strategy the situation may be depicted as a tree of alternatives (see Figure 1) that the user defines in the simulation program:

- The structures that could appear in the course of simulation: sets of nodes, interconnections and values of the parameters defining structures.
- The set of conditions to change from one structure to another.

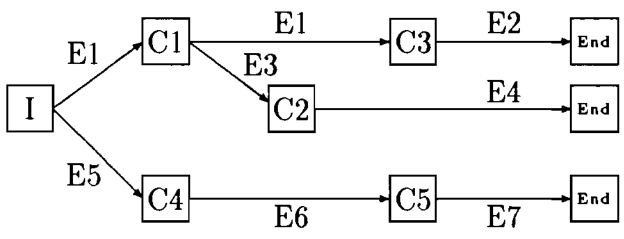

Figure 1: *Simulation Tree Using Predefined Structures*

The arrows E_i represent the structures. Each node of the tree C_j represents a set of conditions to change a structure to one or more successive structures. The node I is an initial node that represents the initial structure. When more than one structure follow, it is necessary to indicate how to select the one that will be processed next. This selection can be made by the conditions of structural change or using predefined heuristics (which may include goal seeking as in heuristic programming) or by indicating a given order of selection so that the system can explore the whole tree, and all possible structural paths are simulated. The programming language has facilities to indicate structures, to specify sets of conditions of structural change, and to store the actual state of the system to simulate new alternatives in the case of a complete searching of the tree. A criteria supplied by the user may be used to compare the performance of different structural paths.

This strategy of predefined structures was applied to the scenario analysis of an expanding firm that grows by adding storehouses and factories in a series of cities. Details of this strategy are described elsewhere [15] .

2 Structural Simulation through Generation of Structures

In the second strategy the computer system creates possible structures and tries different structural paths generated by a controlled random process. This solution was suggested in several methods and theories of creative thinking [1] [2], but it was specially inspired in the model by the mathematicians H. Poincaré [14] and J. Hadamard [11]. This model was based on the experience of Poincaré. His observations were later confirmed by inquiries made by Hadamard. They observed that, in many cases, after a problem is posed and an unsuccessful attempt at the solution is made, the work is discontinued. Days or weeks afterwards a solution may come to mind where the mathematician is not thinking about the problem. The model suggested divides the creation process into three stages:

1. The problem is defined, the elements of the problem are distinguished and the conditions that the solution has to fill are set.
2. A conscious effort is made to find a solution based on previous knowledge and logical reasoning. This is mainly a top-down thinking process. If the solution is found the process is finished, but in some cases no solution is obtained. During this effort

some conditions and restrictions are imposed on the pursuit of the solution or on the combinations of elements in the problem.

3. An unconscious process is triggered by the previous stages. In this stage a random combinatorial handling of the elements of the problem takes place until a suitable solution is found and by a process that is not well understood, this structure is brought to the conscious level. It is essentially a bottom-up constructive process. Poincaré emphasized the aesthetic value of the found structure and he thought that the mind of a good mathematician is particularly sensitive to this aesthetic appeal.

Perhaps a fourth stage may be added in which by a conscious process the emerging solution is tested, details are refined and an intelligeable form is given to it.

This schema was adopted in this research method of solving problems that requires generation of structures without any claim about psychological soundness, which has been questioned by some scholars [13].

The *first stage* is strongly problem dependent on the problem and the method assumed by the user. Expert systems in the problem area may play an important role. But usually the user defines the elements of the problem, possible relationships and general standards to evaluate the behavior of the structural paths.

The *second stage* may also be problem dependent but it is possible to use techniques that apply to a large class of problems. The analysis of AI may be well suited during this deductive stage. The possible structures are found and perhaps some restrictions to structural changes may be set.

The *third stage* is a mechanical one, using random generation of structural paths and testing them with the conditions set in the first stage. It is also possible to generate whole trees of possible structural paths at random. When a satisfactory solution is found (as measured by the standards and conditions fixed at stage 1), it is adopted and the process is stopped. At this stage a small set of the best solutions are worth considering.

To implement the third stage various alternative random combinatorial processes may be used. The one used here is based on genetic algorithms [9]. These are convenient for several reasons:

- A structural path can be easily represented by a *chromosome*, its *genes* being the elements of a vector of the successive structures and vectors of parameters values for each structure.
- A fairly thorough expanded search of the possible structural paths is spanned.
- An organized generation of structural paths is accomplished, and the best ones are selected. Less obvious ones are also generated and allowed to catch unexpected optima.
- It is easy to make parallel processing (however, this has not been tried yet) because in the simple version of genetic algorithms there are no interactions between individuals beyond mating. Parallel processing (which some attribute to subconscious processes) may partially balance the long period, that is spent in the execution of random combinations.

Some further experiments may be made with the solution obtained to refine details and to test its adequacy in other contexts. This activity covers the *fourth stage* mentioned above.

3 A Simple Example

The problem is to find an adequate succession of structures in a processing system in which random parts are submitted to successive processes A and B. Units perform process A, process B, or the succession of processes A and B. In the simulation model there are three types of processing nodes for the A, B, and AB processes. Queues may be formed at these nodes. The queues are lists introduced by the system for these nodes and are called entry lists. Also the enter node (INP) generates parts (messages), and the exit node (E) destroys them. The connection rules of these elements (or subsystems) of the problem are given in Figure 2; in this case, multiple connections from A or to B are forbidden.

INP $\rightarrow$ A	INP $\rightarrow$AB	A $\leftarrow$ INP $\rightarrow$A
AB $\leftarrow$ $INP \rightarrow A$	AB $\leftarrow$ $INP \rightarrow AB$	A $\rightarrow$ B
B $\rightarrow$ E	AB $\rightarrow$ E	

Figure 2: *Possible Connections, Example System of Processing Parts*

It is possible to program all the different structures (in this case five) as disjoint networks. A more compact program may be obtained by programming only one network using subscripted nodes and by changing some parameter values to produce the different structures during the process. These values are determined by a logical process based on some rules for finding the structures.

Using the GLIDER simulation language, the implementation of this process is the following:

- An INP node may be represented by:

```
INP (I):: IT:=EXPO(TBA);
          IF CES1 THEN SENDTO(PROC[J])
                  ELSE SENDTO(PROC[K]);
```

  where INP is an I (input) node that, when activated, generates a message and schedules the next activation of itself. The IT (interval time) indicates the next arrival time. It is chosen from an exponential distribution with mean TBA. CES1 is a logical parameter, and J and K are integer parameters whose values may be set by the structural change according to the required successors of INP in the structure procedure. Thus, the message generated (the part) may be sent to a different processing node.

- Processing nodes are programmed, for example, with the following:

```
PROC (R) [1..6] :: RELEASE IF CES2 THEN SENDTO(PROC[M])
                                   ELSE SENDTO(E);
                   STAY:=PROCTIME[INO],
```

where PROC is an R (resource) node that has a subscript. Actually it represents six independent nodes distinguished by the subscript INO. They are processed with INO=1,2,3,4,5 and 6:

- PROC[1] PROC[2] perform the process A
- PROC[3] PROC[4] perform the process AB
- PROC[5] PROC[6] perform the process B.

They are distinguished from other nodes by having different processing times and successors. The STAY instruction defines the processing time, i.e. the time that the arriving message remains in the node in the internal list. During this time, other arriving messages are queued in the entry list for the node. This list is introduced automatically and is called

EL_PROC[subscript] (entry list of the node PROC[subscript])

When the message is released the RELEASE part is executed and the message is routed to another processing node or to the exit node E.

- The two above codes and the exit node E are enough to represent (with adequate values for CES1, CES2, J, K, and M), all the possible structures for this simple case of nodes with no more than two successors. So in this case the code is very compact.

To simulate the structure given in Figure 3, in which parts may be processed by machines AB or machines A and later machines B, the variables must have the following values:

$$\begin{tikzcd}
 & AB \arrow[rr] & & \\
INP \arrow[ur] \arrow[dr] & & & E \\
 & A \arrow[r] & B \arrow[ur] &
\end{tikzcd}$$

Figure 3: *A Possible Structure*

- $CES1 = LL(EL\_PROC[J]) \leq LL(EL\_PROC[K]);\ J = 1; K = 3;$
- if $INO = 1$ then $M = 5$, $CES2 = TRUE$;
  if $INO = 3$ then $CES2 = FALSE$

where, LL is a function that gives the length of the queue. This assignment of values to $CES1$, $CES2$, J, K and M makes the INP node send the message to the smaller queue and the processing PROC node send the message to process B or to the exit E node. The parameters are the processing times at each processing node. They are random values taken from a uniform distribution between two given limits.

The structural paths in this example is confined to three successive structures at most. The condition of structural change is a predefined degree of system crowding given by the joint length L of all queues. The performance is given by the function: $F = (aL + bC - 15)^{-2}$ where a, b are constants and C is the cost of the processing structures for each part processed. It is different for different configurations of the nodes and increases for shorter processing times. The form of the function warrants a good separation of different performances.

More sophisticated and realistic fitness functions could be designed, but this explanation is enough for the present demonstration purpose. All of these conditions and coding are given to the computer system by the user and they should be strongly dependent on the problem. They correspond to stage 1 of the process.

The objective of the second stage is to find the possible structures. They result from the connection rules. The elements may be coded as follows:

INP		0
PROC[1]	(A)	1
PROC[2]	(A)	2
PROC[3]	(AB)	3
PROC[4]	(AB)	4
PROC[5]	(B)	5
PROC[6]	(B)	6
E		7

The structure may be represented by a 5x7x7 array G, in which G_{nij} equals 1 if there is a connection from element i to j in the structure n, and 0 otherwise. For instance, if the structure given in the above is numbered 4, then the elements of the array G that are different from 0 are:

- $G_{401} = G_{403} = G_{415} = G_{437} = G_{457} = 1.$

A straightforward algorithm is required to obtain from the given connection rules the matrix structure for each feasible structure. Five structures (models of the processing system) are generated with these mapping rules.

The following procedure is used to map a structure into the parameters of the above nodes.

- If there is only one $G_{n0j} = 1$, set: $CES1 = TRUE, J = j$.
- If there are two $G_{n0j} = G_{n0k} = 1$, set:
  $CES1 = LL(EL\_PROC[J]) \leq LL(EL\_PROC[K])$, $J = j, K = k$.
- For node PROC[INO] $INO = 1,2,3,4,5,6$:
  If rows 1, and 2 do not contain a 1 then set $CES2 = FALSE$ (only one pass processing)
  otherwise:
  If INO=1, and 2 and $G_{n1k} = 1$ or $G_{n2k} = 1$ set: $CES2 = TRUE, M = k$;
  If INO=3,4,5, and 6 set: $CES2 = FALSE$.

These rules are also valid for other sets of structures. This completes the second stage which is governed by rules that may be applied to different sets of elements giving different structures.

The third stage of the Poincaré and Hadamard creativity model was programmed in the following way. The structural path in this example is made of a series of three of the five E_i structures given in stage 2. They are generated randomly by taking a sample with replacement of the set 1,2,3,4,5 of structures. For each of the three processing elements the parameters P_j (that is J, K, M) are generated by taking a value at random from a uniform distribution within the limits given in stage 1.

One structure is coded in a chromosome with the genes for the structures and parameters as a record of fields (see Figure 4): A field F to store the computed fitness was

E1	E2	E3	P1	P2	P3	F

Figure 4: *The Chromosome of the Processing Parts Example*

added for convenience. It is not subject to mutations or cross-overs. At the beginning a population of chromosomes is generated randomly from the values of the structures and the parameters. The F is estimated by running the model with the given structural path and parameters (gene "expression") for each chromosome and computing the given fitness function. During this process, the structural changes are made according to conditions given in stage 1. The cross-over is made separately in the structural succession part and in the parameter part of the chromosome. Two parents are chosen from the population, a point of division is taken at random (1, 2 or 3), and from this point onwards genes are interchanged.

This process generates two new chromosomes corresponding to new structural paths. Mutations are randomly introduced (with low probability) replacing one of the structures in the path by another one, chosen randomly from the possible paths. The same procedure occurs for the parameters. The pair of parents are chosen with a probability proportional to their F value, so that better-fit individuals have a greater chance of transmitting their gene schemata to descendants. Pairs of offsprings are generated by this mating process until a new population that is the same size as the old one is generated. After the evaluation of F for all individuals a search is made to see if some individual's satisfy the criteria given in stage 1. In the above case a minimum level for F is required. If a satisfactory individual is not found a new generation is produced. The process continues until a solution is found or a maximum prefixed number of generations are processed. This completes stage 3.

In addition to the described model, the GLIDER program has nodes to initialize the population, to control the structural change by periodical inspection of the state of the system and testing its conditions and to control the decoding and mating of the chromosome, as well as a graphical output showing the evolution of mean and maximum F. As the simulation time can be freely manipulated in GLIDER the time starts at 0 in each simulation run of the model. A program controlled time variable is maintained for the population evolution. Many experiments with changes in the limits of the parameters are run. A typical output for maximum and mean F for successive generations is shown in Figure 5. An interesting result was that some structural paths, such as the paths from

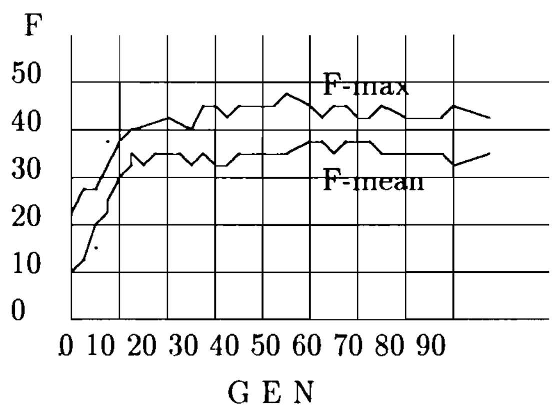

Figure 5: *Output of Performance Function (F) of Successive Generations (GEN)*

structure 5 to structure 2 and followed again by structure 2 (path 5 2 2) and path 2 1 2, achieved higher efficiency but only during a few generations as they were more sensitive to random changes in the parameters. On the other hand, other slightly less efficient paths such as 5 1 2 and 1 2 2, dominated the population most of the time. They were more robust and could deal with random fluctuations. Therefore, when analyzing the results it might be necessary to consider not only the best path but also the more abundant paths if they have a good value of the fitness F. This robust characteristic of genetic solutions has been stressed in the literature [9]. Structural simulation is a new area and for this reason analysis of structural simulation results is still an open field for research [8].

4 Problems and Possible Developments

This method can be improved in many ways. As it requires considerable computer time, parallel processing may be used, especially at the combinatorial stage. Feedback between the different stages may be considered. If a solution is not found, the conditions fixed in stage 1 may be relaxed. The chromosomes may represent trees of structural chains instead of simple successions. Competition among the individuals generated may be introduced (as was made by people working in a-life [10]) to improve the quality of the solution. Other random processes may also be tried.

References

[1] Arieti Silvano, *Creativity. The Magic Synthesis.* Basic Books. 1976.

[2] Bloomberg Morton, *Creativity: Theory and Research.* Albany: New College and University Press, 1973.

[3] Domingo Carlos, Hernández Marisela, Sananes Marta, Tonella Giorgio, *Lenguaje de Simulación GLIDER: Guía de Referencia, Version 3.* IEAC and CESIMO, Universidad de los Andes. Mérida, Venezuela, 1994.

[4] Domingo Carlos, Sananes Marta, Tonella Giorgio, *The GLIDER Simulation Language.* Report IEAC-CESIMO, Universidad de los Andes. Mérida, Venezuela, 1994.

[5] Domingo Carlos, Tonella Giorgio, Herbert Hoeger, Marisela Hernández, Marta Sananes, Silva Jose G., *Object Oriented Programming Ideas in a New Simulation Language.* J. Schoen (ed.), Proceedings of Summer Computer Simulation Conference, Boston. July, 1993. The Society of Computer Simulation Corp. San Diego. pp. 137-142. July 1993.

[6] Domingo Carlos, *El Cambio Estructural.* Edicion Departamento de Computación, Universidad Central. Caracas. 1973.

[7] Domingo Carlos, *Simulación del Cambio Estructural.* XVII Conferencia Latinoamericana de Informática. Caracas. 1991.

[8] Domingo Carlos, Quiróz Segundo, Terán Oswaldo, *Statistical System for The GLIDER Simulation Language.* III Network of the Biometric Society. Caracas, July 1994.

[9] Goldberg David, *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.

[10] Hillis Daniel, *Co-evolving Parasites Improve Simulated Evolution as an Optimization Procedure.* A-Life II.

[11] Hadamard Jacques, *The Psychology of Invention in the Mathematical Field.* Dover Publications Inc. Mineda. 1954.

[12] Oren T.I. *Simulation of Time Varying Systems.* Proceedings of the International Conference of Cybernetics and Systems. Gordon and Breach. Oxford, 1990. pp. 1229-1236.

[13] Perkins D.N., *The Mind's Best Work.* Harvard University Press. Cambridge. 1981.

[14] Poincaré Henry, *Science et Méthode.* Flammarion, Paris. 1908.

[15] Terán Oswaldo, *Simulation of Structural Change and Scenario Analysis.* In Spanish, M.Sc. Thesis, IEAC, Universidad de los Andes. Mérida, Venezuela, 1994.

[16] Zeigler Bernard, Tag Gon Kim and Chilgee Lee. *Variable Structure Modelling Methodology: An Adaptive Computer Methodology Example*, Transactions of the Society for Computer Simulation. Vol 8(4), December 1990, pp. 291-314.