Stanisław Gasik
Department of Psychology
University of Warsaw

# WSL – A Formal Language for Social Processes Description

1993 Western Simulation Multiconference on Simulation Applications in Business Management and MIS, the Society for Computer Simulation, La Jolla, California, USA.

## 1 Introduction

Following Davies and O'Keefe (1988) programming languages used for simulation may be categorized into three groups. The first contains *general purpose programming languages* like PASCAL, FORTRAN, C and hundreds of others. They may be applied for simulation as well as for any other domain of programming. The second group includes *simulation programming languages*: SIMULA, SIMSCRIPT and GPSS are among them. They provide essential facilities designed for simulation: "processes", "events", "time" or "objects". Their implementation is hidden from the user and this way these languages are much better suited for simulation. The third class is a class of *domain dependent languages (DDLs).* Most of concepts from given domain of application are built into such language by its author. For social sciences we may give examples of "subject", "characteristic", "interaction", "initial distribution", "population" etc. as examples of concepts which must be handled in description of every process. This way programs written with use of a DDL are most concise. DDLs may be used by people with little computer experience. As for professionals from given domain they are as transparent as possible, they may be used not only for simulation, but also for formulating theories. So, they may become tools for exchanging ideas among research workers.

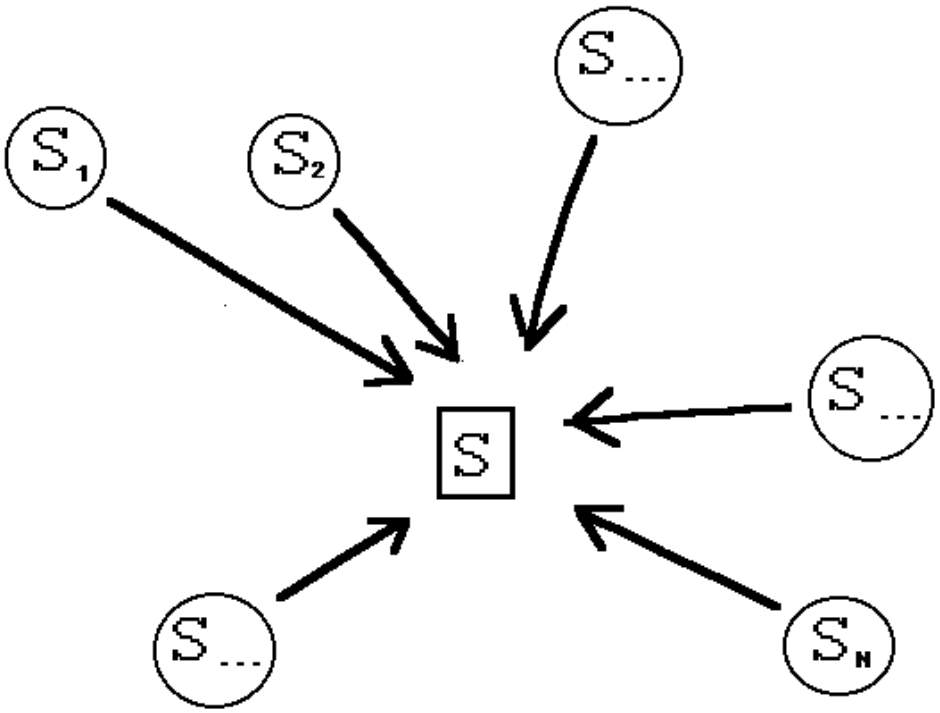
In this paper I introduce a DDL for description and simulation of social processes - the Warsaw Simulation Language (WSL). Originally, starting in 1989, it was designed for simulation of social impact following the theory developed by Latane (1980). The current version has been expanded to cover many other classes of social processes.

## 2 Basic concepts

The most basic concept in every social process is perhaps the concept of a *subject*. In the WSL every subject is equivalent to and described by a set of *characteristics*. This set of characteristics is the same for every subject in the population being described. The *population* is placed on a flat, rectangular area, divided into rows and columns - the *process area*. It does not have to be fully filled with subjects.

In the reductive approach to description of social processes, behavior of a population is equivalent to a sum of behaviors of its members. So, description of population's behavior is reduced to description of behavior of a single member of this population. Hence, we get the concept of the *current* subject. A user of the WSL must provide just the description of the current subject. Speaking other words, a process is described from the point of view of one, current subject. Other

subjects are present in this description as sources of interaction for the current one. Behavior of subjects may be differentiated on the basis of differences of values of the current subject's characteristics.




A process, described with use of WSL, may be simulated within the environment of the Warsaw Simulation System (WSS). The process of simulation consists of many *passes*. One pass consists of two main actions:

- selection of a current subject,

- execution of interaction rules for the current subject.

A *step* is a unit of simulation greater than a pass. Step is a number of passes equal to number of subjects in given population. It may be treated as the basic unit of time.

# 3 Process description

A description of a process consists of four parts:

- *general condition* part,

- *initialization* part,

- *interaction* part,

- *results* part.

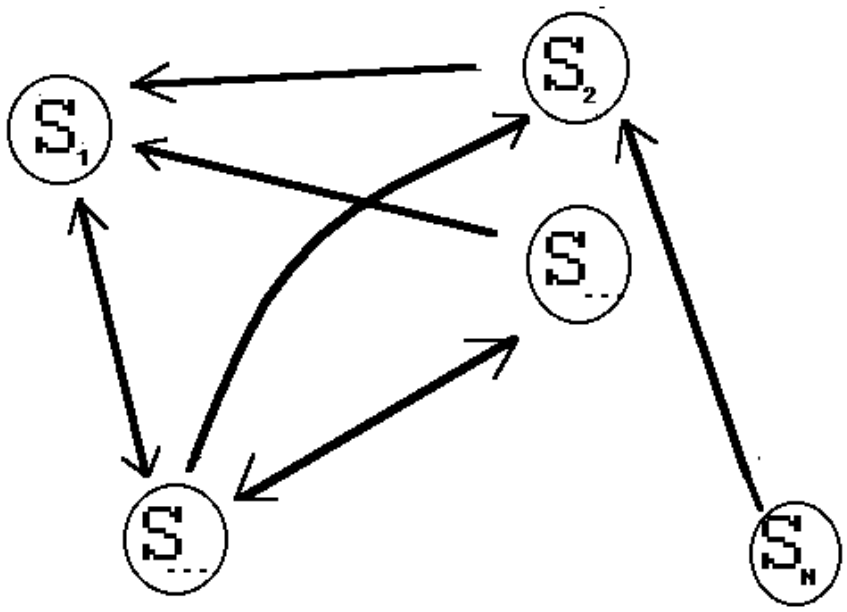
In the first part the process area size, names of characteristics, information about area's topology, group size and some other information of general meaning should be passed to the system.

In the initialization part the starting situation must be defined. The initial distribution may be described from scratch or it may be loaded from an output of former simulation.

The process part describes interaction between subjects. This part reminds traditional, procedural languages. The process part should end with *stop rule*, stating a condition for process termination.

The result part is optional. There are several kinds of output, describing dynamics of a process as well as the final state. It may be produced in ASCII format (for analysis by external packages) or in the internal WSS format (for further processing).

The WSL contains a mechanism for description of automatic repetition of simulations. Values of so-called *parameters* may be changed to test their influence on given process.




# 4 General conditions

The general conditions of a process may be divided into three basic groups:

- declaration of variables,
- topological parameters,
- other conditions.

## 4.1 Declaration of variables

There are two kinds of *variables*: characteristics and working variables. The main difference between them is such that there is only one copy of every existing working variable

while there are as many values of every characteristic as subjects in the population. Characteristics are of two kinds: the main one and others. The main characteristic may assume no more than ten values from the set of numbers from the 0 - 255 interval. A graphical representation must be given for every value of this characteristic, as it is permanently displayed on the screen during simulation.

The other characteristics, as well as working variables, may be of the REAL (contiguous) or INTEGER (discrete) type.

There are two predefined characteristics: COORDX and COORDY, giving the horizontal and vertical coordination of every subject. Their values may not be changed in an explicit way.

## 4.2 Topological conditions

The following general conditions belong to the group of topological conditions:

- group size,

- window size,

- homogeneity,

- filling ratio,

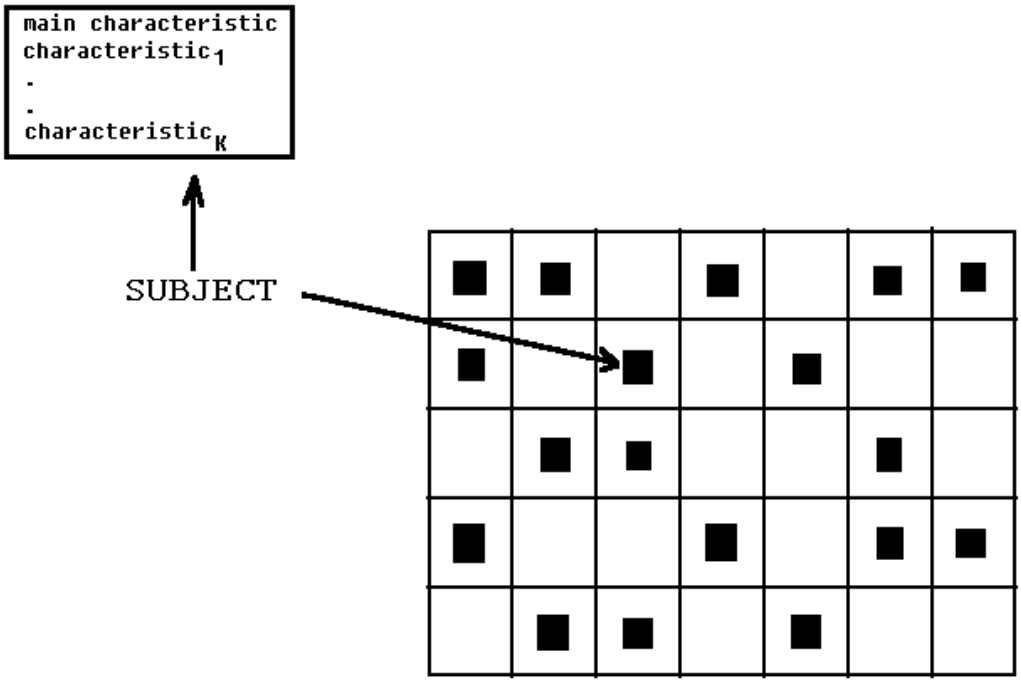
- holes.

The group size ranges from 1 by 1 up to 40 rows by 40 columns, giving population of 1600 subjects.

The concept of a *window* is very important for description of interaction. A window is a square territory around a subject. The window size is the distance between its central point (the subject for which the window is constructed) and any of its edges. Every edge belongs to the window. E.g., a window of size 1 covers 9 subjects; its edge's size is 3. Generally, a subject may interact with other subjects only within its window. If the window size is 1, then subjects interact with its closest neighbors, only.

If the process area is *homogeneous*, then every subject on the upper (left) edge of the simulation is treated as direct neighbor of proper subjects from lower (right) edge. If the area is not homogeneous, subjects on edges (and in corners) of process area have respectively smaller number of direct neighbors.

The process area may be fully populated with subjects or some territories, pointed as particular subjects, rectangles or sums of them, may be left free. Moreover, a user of the WSL may state the ratio of filling of the non-empty process area. In the second case the empty and populated placed will be distributed randomly according to the ratio of filling.




## 4.3 Other conditions

There are two other general conditions.

The *method definition* will be described in chapter 6.

The *definition of sources* states, which parts of process area will be reinitialized, according to their general initialization rules, before every step.

**Example 1**

**PROCESS DESCRIPTION;**

**SIZE** 10 **BY** 10**;**

**MAIN CHARACTERISTICS** Attitude **VALUES** 0 = '-', 1 = '+', 2 = '?'**;**

**CHARACTERISTICS** Strength **INTEGER;**

**WORKING** GlobalResource, Work2 **REAL;**

**HOMOGENEITY ON;**

**METHOD PARALLEL;**

# 5 Initialization

Every process must start from some situation. There are two possible ways of initialization:

- assigning new values on the basis of fixed values or random distribution,

- loading values of variables produced by previous simulations.

These two ways may be mixed, IE. some variable(s) may have assigned new values, while other(s) may be loaded.

## 5.1 Assigning new values

A single subject, a rectangle, a sum of these units, or all the population may be pointed as a group of subjects to be initialized in a particular way. These groups may be constructed in different way for every characteristic.

One may assign a fixed value or a random distribution to every initialized variable. The normal and uniform distributions are currently available in the WSL.

Every characteristic must be initialized, while initializations of working variables are optional.

## 5.2 Loading situation

Files of the .STN type, containing situations (refer to chapter 7) may be fully or partially loaded for further processing. A working variable(s), a characteristic or all the variables may be restored from such file. As it was mentioned before, parts of process area may be reinitialized, giving the effect of utilizing only a part of situation produced by previous simulation.

**Example 2**

**INITIALIZATION;**

Message **FOR ALL IS** 0, **FOR** (10, 20) **IS** 1;

Money **FOR ALL IS NORMAL** (100, 30, 50, 150);

WorkVar **IS** 0;

**LOAD** OldChar **FROM** 'OLD';

# 6 Interaction

## 6.1 Methods of interaction

There are two methods of interaction:

- parallel,

- Monte Carlo.

### 6.1.1 Parallel

In the parallel method all subjects from a population within one step execute their rules of interaction in the same conditions. This is achieved by maintaining two arrays of every characteristic: one with values at the beginning of given step and another with values produced during execution of interaction rules for every subject. The latter is copied onto the former upon completion of every step.

Subjects are selected to become the current subject in an ordered way, row after row. So,

every subject becomes the current subject once every step.

Every working variable has only one copy. So, if the same effect of parallelism for working variables should be achieved, another copy of this variable must be declared and maintained by the user.

### 6.1.2 Monte Carlo

In the Monte Carlo method, there is only one copy of every characteristic and working variable. So, every modification is recorded immediately after execution of given interaction rule. Every next current subject inherits situation modified by his predecessor.

In this method some subjects may be fired once, some may be not fired and some may be fired more than once in a step.

## 6.2 Modes

There are two modes of interaction:

- the *block mode*,

- the *single mode*.

Both modes of interaction may be utilized in one process description.

### 6.2.1 Block mode

This is the many-to-one type of interaction. Given current subject is exposed to influence of other subject from his window. The three so-called *aggregating* functions: SUM, MEAN and COUNT has been designed to handle this mode of interaction. The SUM (MEAN) computes sum (mean) of arithmetic expressions, built from characteristics of subjects within given window. Both of these functions may operate for all subjects in the window or only for those which have given value of the main characteristic. This way sum or mean may be computed for some group of subjects. The COUNT counts number of subjects with particular value of the main characteristic within a window.

In this mode values of characteristics of the subject for which the window is constructed may be updated, only. There is no way to change values of other subjects' characteristics. Their characteristics may be changed in another pass, for example, when they become the current one.

Window size may be a parameter of every aggregating function.

**Example 3**

**SUM** (0, Strength/**DISTANCE**, 3)

Computes sum of Strength/**DISTANCE** within window of size 3 for subjects with the **MAIN CHARACTERISTICS** equal to 0.

**COUNT** (1);

Computes the number of subjects with the value of the **MAIN CHARACTERISTIC** equal to 1 within the default (or declared in the general condition part) window.

### 6.2.2 Single mode

This is the one-to-one type of interaction. There may be full (for retrieval and update) access to every characteristic of every subject involved in an interaction.

By default, any accessible operations may be performed only on the current subject. To describe single mode, there must be an apparatus for selection of other *active* subjects. An active subject is a subject involved fully in interaction, with full access to its characteristics. The *select* statement gives a possibility of firing other active subjects. As there may be more than one select statement there has to be a possibility of identify active subjects. The *reference number* were designed for this purpose.

In the select statement a subject from some window around the current or around previously selected active subject, may be chosen. The value of the main characteristic for searched subject may be stated. If the selection operation is successful, a reference number is assigned to the new active subject. This reference number may be used further on to qualify characteristics' names.

With use of the single mode there is a possibility of direct modification of characteristics of more than one subject at a time.

The effect of a select statement is valid until the end of given pass.

**Example 4**

```
* Lines with * in the first column are not analyzed by the WSS
SELECT 0 REFNO 1 WINDOW 3;
* Selects an active subject with the MAIN CHARACTERISTIC equal
* to 0 within window of size 3. This subject, if found, will be
* referred to as the first active subject.
IF SELECTED (1) = THEN
*                test, if it was found
 MeanMoney = (OWN_Money + Money.1) / 2;
*                OWN_ prefix refers to the current subject
 OWN_Money = MeanMoney;
 Money.1 = MeanMoney;
END IF;
```

### 6.2.3 Mixed mode

Both modes may be mixed in one process description. The aggregating functions may operate on the basis of any active subject. Values of aggregating functions and values of characteristics of particular subjects may be used in one expression to produce combined types of interaction.

## 6.3 Statements

Statements from the interaction part of the WSL have been designed in a traditional, PASCAL-like syntax.

### 6.3.1 Assignments

In the basic statement of *assignment,* a value of a characteristic or a working variable may be changed. This operation may be performed on any characteristic of any active subject because reference numbers may be used. Assigned values are described by expressions constructed over characteristics, constants and functions with use of traditional arithmetic operations.

#### *6.3.1.1 Functions*

There are three groups of functions:

- Traditional arithmetic: *abs, exp, integer, log, max, min, normal, random, real, sqrt, unicontig,* and *unipoints*;

- Aggregating functions, described in 6.2.1: *sum, mean* and *count*;

- Functions designed for description and measuring of processes. The *area, distance, length, population, sqrdist, time, trials* and *width* describe general process conditions. The *changes, changesin, changesout, cohesiveness, connections, groups, isolated, number, selected, shiftin, shiftout, shifts* and *surrounded* were designed to measure performance of simulation.

Some of these functions are described in section 9.

### 6.3.2 Conditional statements

The conditional (IF) statement in the WSL is a multi-branch one. It means that for one IF header there may be one or more ELSE IF parts. Every IF statement may be terminated with the OTHERWISE part. Logical conditions within conditional statements are constructed over arithmetic expressions (see above) with use of relational ( < > >= <= = ~=) and Boolean (AND OR NOT) operators.

Every logical expression, starting from the first is evaluated. The statement(s) standing after first such expression which evaluates to true, is executed. If there is no such expression and there is an OTHERWISE part, this part will be executed. If there is no true logical expression and no OTHERWISE part, nothing is executed.

Conditional statements may be nested, IE. another IF statement may be used as one of statements after every logical condition. They may be nested with loop statements, too.

**Example 5**

```
IF TIME = 1 THEN
 Threshold = 0.9;
 WorkVar = 50;
ELSE IF TIME < 10 THEN
 Threshold = 0.7;
 WorkVar = 60;
```

```
OTHERWISE
 Threshold = 0.3;
 WorkVar = 50;
END IF;
```

### 6.3.3 Loops

There is a simple construction for multiple execution of sets of statements within a pass. This is a *loop*. A set of statements making a loop (a *body* of a loop) is repeated while a logical expression, stated in its header, evaluates to true. This test of logical expression is carried out after each termination of the body of a loop.

Loops may be nested within other loops. They may be nested with conditional statements, too.

One of statements within a loop may be an *exitloop* statement. Execution of this statement causes immediate leaving of the loop, IE. the next executed statement will be the first statement after the loop.

**Example 6**

```
Index = 0;
Found = 0;
LOOP WHILE Index < 10 AND Found = 0;
 SELECT 1 REFNO 2 WINDOW 1;
 IF SELECTED (2) = 1 THEN
  Found = 1;
 OTHERWISE
  Index = Index + 1;
 END IF;
END LOOP;
```

## 6.4 Step statements

Every step consists of many passes - every statement used to describe subject's behavior is executed many times. A set of *step statements* is executed only once for every step, before the first pass of a step. All kinds of statements described in point 6.3 may be used as step statements. But only values of working variables may be modified there. There is no possibility of reference to any particular subject.

**Example 7**

**STEP STATEMENTS;**

StepIncome = 1000;

MeanImpact = **MEAN** (**ANY**, Impact);

**END;**

## 6.5 Stop rule

The way to precise a condition of process termination is a *stop rule*. The only thing which must be stated in this rule is just a logical condition for process termination. In this condition, like in step statements, there may be no reference to any particular subject.

The stop rule is tested after termination of every step, or, if the interaction is divided into several phases, after every step of the last phase (chapter 6.6, below).

The stop rule is optional. If it is absent, simulation may be interrupted by an action from the keyboard (refer to manual of the WSS system).

**Example 8**

**STOP WHEN TIME = 10 OR SHIFTS (ANY) = 0;**

## 6.6 Switches, segments and phases

A process may be divided into several *phases*. Description of a phase is called a *segment* of WSL code. Segments are delimited by *switches*, in which a logical condition of passing from one phase to the next must be stated (again, as in stop rule, without reference to specific subjects). This condition, if present, is tested upon completion of every step from the phase, preceding given switch.

If there is no switch statement, we may say that a process is built from one phase.

The homogeneity (on/off) and method (parallel/Monte Carlo) may be changed as result of positive testing of a switch.

Every segment may be headed by a set of step rules.

**Example 9**

**SWITCH WHEN MEAN** (**ANY,** Money) < 1000;

(insert figure 6 here)

# 7 Output

The results section is optional. If it is absent, the only way of presentation of simulation is

displaying values of the main characteristic on the screen. But there are four other kinds of monitoring results of simulation.

## 7.1 Snapshots file

The snapshot file is the basic working file for the WSS. Listing of process description, together with error diagnosis (if needed) is stored there. During simulation this file is filled with data after initialization and completion of every n-th step. The default interval is one step.

The snapshots file is stored in a form of an ASCII file, thus allowing for passing it to an analysis with use of any statistical package.

The substantial contents of a snapshot file are snapshot statistics. Arithmetic expressions are used to construct such statistics. There is no possibility of using values of characteristics of particular subjects for construction of these statistics.

Snapshots of graphical representation of the main characteristic optionally may have their place there, too.

**Example 10**

**SNAPSHOTS STATISTICS TIME, NUMBER** (0)**, NUMBER** (1), GlobalResource**;**

## 7.2 Variables

Pure values of any variable may be stored to an external ASCII file, too. They are stored in the traditional, subject/variable shape. An easy access to individual characteristics of subjects is supplied this way. Indices not present in the WSL may be easily constructed over these data for further analysis. Variables are stored with the same frequency as snapshots statistics.

**Example 11**

**SNAPSHOTS CHARACTERISTICS** Message, Strength, GlobalResource**;**

## 7.3 Final statistics

Final results of simulation are called *final statistics*. They are described by user defined arithmetic expression in the same way as snapshot statistics. Final statistics are stored in a separate, external ASCII format file; they may be easily submitted for further analysis. The final statistics file is loaded twice every simulation: after initialization and after termination of a simulation.

The mechanism of final statistics is especially useful in connection with apparatus of multiple execution (chapter 8). Results of repeated simulations may be aggregated. Influence of particular parameters over final results of simulation may be tested.

**Example 12**

**FINAL STATISTICS TIME,** @Parameter1, @Parameter2, **CHANGES (ANY)**;

## 7.4 Situations

Output described in 7.1 - 7.3 may be used as an input for further analysis. A *situation file* contains variables coded in the internal, WSS format. Thus, it may be used only for the WSS system itself, only. One or more characteristics or working variables or all of them from given simulation may be stored there. A situation, or its part (selected variables) may be loaded for further processing (chapter 5.2) in the initialization part of process description.

**Examples 13**

**SAVE ALL TO** 'Sit1';

**SAVE** Char1, Char2, WorkVar **TO** 'Sit2';

# 8 Multiple execution

Given process description may be automatically run (simulated) once or more times. It may be run without any modification in process description - this is called *repetition*. The other way of multiple simulation is *parameterization*. Arithmetic expressions in process description may be substituted with *parameters*, which may be varied in the way stated by the author of process description. If there is more than one such parameter, process is simulated in full factorial plan.

Repetition and parameterization may be joined in one simulation. Then for every combination of parameters simulation is repeated stated number of times.

The mechanism of final statistics (chapter 7.3) is especially useful in analyzing results of multiple process simulations.

**Example 14**

**REPETITION SPECIFICATION;**

**RUN FOR** @Parameter1 = (1, 2, 3),

@Parameter2 = (0.1, 0.5)

**REPEAT** 5 **TIMES;**

# 9 Selected functions

Many of the below presented functions refer to values of the main characteristic. The special keyword **ANY** may stand for any value of this characteristic there.

Another common parameter is *neighbours*. There are two possible kinds of neighbourhood relation describable by this parameter. When *neighbours* is equal to 4 then only the left, right, down and upper subjects are treated as having direct contact. When *Neighbours* is equal to 8 (the default for this optional parameter), or it is absent, then all subjects from a window of size 1 will be considered as having direct contact with given subject.

**Changes** (*Value*)

Number of main characteristic changes from the beginning of given simulation within given *value*. This is a sum of **changesin** and **changesout** (below).

**Changesin** (*Value*)

Number of subjects, which accepted given *Value* as the value of main characteristic from the beginning of the simulation.

**Changesout** (*Value*)

Number of subjects, which rejected given *Value* of the main characteristic from the beginning of the simulation.

**Cohesiveness** (*Value*, *Neighbours*)

For every subject with the main characteristic equal to *Value*, number of its direct neighbours with the same value of main characteristic is counted. These numbers are summed up to give the cohesiveness value.

**Distance**

May be used within **Mean** or **Sum** aggregating functions. Stands for the distance between the active subject, for which the function is evaluated, and a subject over whose characteristics the function operates.

**Groups** (*Value, Neighbours*)

Number of topological groups (clusters) formed by subjects with the main characteristic equal to *Value*.

**Isolated** (*Values, Neighbours*)

Number of subjects with specified *Value* of the main characteristic, having no direct contact with subjects with any other value of main characteristic.

**Length**

Horizontal size of process area,

**Number** (*Value*)

Number of subjects with the main characteristic equal to *Value*.

**Population**

The total number of subjects in a process.

**Random**

A real, pseudo - random number from the <0, 1) interval.

**Shifts** (*Value*), **Shiftin** (*Value*), **Shiftout** (*Value*)

As **Changes**, **Changesin** and **Changesout** but counts number of main characteristic's changes within given step.

**Surrounded** (*Value*, *Neighbours*)

Number of subjects with specified main characteristic *value* having no direct contact with any other subjects with the same *Value* of the main characteristic.

**time**

Starts from 0, and is increased by 1 before every step.

**trials**

Number of passes executed from the beginning of the simulation.

**width**

Vertical size of process area.

# 10 Possible applications

There may be several kinds and fields of application for a language like WSL. The first kind is using a formal language as a tool for description of models of social processes. A formal language forces its user to formulate ideas in more precise, unambiguous way, thus constituting more rigorous requirements for all the research process. A very important by-product of using such tool for describing ideas is providing a tool for communication for the society involved in research, practice and education in given field of science.

The second kind of applications, strictly related to the first, is providing a tool for simulations of social processes. The WSL seems to be much simpler, than common programming languages, for social scientists and hence easier to use for them. The relative simplicity has been achieved by incorporating into it only concepts from social sciences. Implementation of basic concepts from this domain has been done by the author and is hidden from the user. Users of the WSL should handle concepts which are meaningful and unique for given process.

Simulation, from the other hand, is a tool for verification of models. Every process described in the WSL may be very easily run on a computer equipped with the WSS system. All kinds of output may be analyzed and compared with experimental data in order to verify a model. Results of such verification may be applied for refinement of the model.

The WSL may be applied in every domain of science where behavior of single subjects within a population is of interest. We may mention psychology, sociology, economy, demography, epidemiology, political science or even geography.

# 11 Implementation- Warsaw Simulation System

The WSL has been implemented by the author on IBM/PC and runs under DOS operating system as the Warsaw Simulation System (WSS). The WSS is a working environment. It is menu-driven and equipped with help system, available during system's operating.

The WSS consists of three main components:

- a *compiler* of processes description,

- an *executor* of a code generator by the compiler,

- an *editor* of situations,

- the above-mentioned *help* subsystem.

An external editor, producing output in an ASCII format, should be used to edit a process description. It may be accessed from within the WSS. The same editor may be used for reviewing any ASCII-type output of the WSS.

The compiler converts process description from the source WSL code into the internal WSS code. It checks correctness of process description and produces an error report, if needed, too.

The executor runs simulations, displaying graphical representation of the main characteristic on the screen. Some basic statistics, like NUMBER, SHIFTSIN, SHIFTSOUT for every value of the main characteristic are presented on the screen, too.

Simulations may be interrupted from the keyboard any time. Value of every characteristics of every subject may be checked and modified. The current situation may be stored in an external file and further modified, by situations editor, later, or may be input to another process.