SIMULATION OF TELETRAFFIC SYSTEMS: SPECIFICATION LANGUAGES, SIMULATION LANGUAGES, SOFTWARE

Gerard JUNIN, Janis SEDOL, Victor SUPE
Computing Center of Latvian State University, Riga, USSR

ABSTRACT

Some extensions and further development are offered for the specification language SDL, which is recommended by CCITT for the functional description of the telephone exchanges with stored program control.

Simulation system SPALM85 and SDL/PL, developed in Computing Center of the Latvian State University, are based on the extended SDL. The software is implemented on 50 computers, compatible with IBM/360.

Basic terms, tools, characteristics and facilities are described for both simulation systems, as well as their application field during the project stage and investigations of communication systems.

1. INTRODUCTION

With the increase of complexity of telecommunication systems, the design of telephone exchanges with stored program control, the developing computer technique and engaging of computers in the structure of communication systems and networks, the simulation method becomes more important and frequently it is the only way of the investigation of the characteristics of many systems. Simulation languages and applied program packages are developed to process the simulation. To the current moment more than 500 simulation systems are designed throughout the world and new systems are developing [1-3]. The development is caused by new application fields and the introduction of the tools and technologies, which are giving new facilities and are decreasing the expenses. This may be related both to the design and investigations of communication systems, and to the design, debugging and maintenance of simulation programs.

The efficiency of investigations is considerably dependent on the choice of simulation language and it's software. The modern requirements for program tools suppose the presence of the specification language, too, for simplification of the project development, the functional description of the system to be simulated and the better understanding between the programmers and system investigators. The tradicional simulation on systems and languages either do not use such tools for specification, or the tools are too close to the syntax of the simulation language (as, for example, in GPSS). As a result, the difficulties arise in communication between the designers, the investigators of the real systems and the programmers. This is the main source of errors and the lack of correspondence between the real object and its model.

The proposed simulation systems SPALM85 and SDL/PL are based on the specification language SDL, which is recommended by CCITT [4]. At the same time the simulation languages and their software have some features, which determine their conveniences and possibilities of their use, as well as their application fields.

The system SPALM85 for the first time provides the common way for both the simulation model design and the algebraic calculations by solving state equations using iteration method. The building of the simulation program in the form of the single module gives a possibility to obtain very high technical and economical showings (the resources of main storage of computer, the processor time for experiment). The system is easy to learn and simple in the usage.

The system SDL/PL is provided for simulation of complex systems, where the components of the model may be developed separately, using independent translation, data input and output of results, as well as separate debugging for each component. The libraries of model components may be stipulated. The complexity of models are limited only by the resources of the available computer. The software for SDL/PL is implemented taking into account special debugging tools for simulation programs. The system is applied to the simulation of communication systems and for the design of debugging and testing tools for the software of quasielectronic and electronic telephone exchanges. The system has the facilities for the exchange of signals and data with the programs, designed in other programming systems, as it is stipulated by PL/1 programming language.

In addition, both simulation languages have different coding style
2. SPECIFICATION LANGUAGES

CCITT has recommended Specification and Description Language (SDL) [4] for the functional description of telephone exchanges with stored program control. The symbols, used in this language, are presented in Fig. 1.

![SDL Symbols](image)

Fig. 1 SDL Symbols.

The delay of signals for time $t$ (external and internal) 

$$ S(t) $$

The delay of process for time $t$ 

$$ S(t) $$

The delay of signal $S$ for time $t$, the type of signal - $\alpha$

The delay of signals and the signals, which get lost, is following: if the process instance exists in the associated states, then the signals of both types transfer the process instance into transition. In the case of absence of process instance, the waiting signal moves to the queue of signals for the given state and is handled, when the process instance arrives.

The signal of type L in this case is got lost and no actions upon the state are processed. The cancelling signal has an influence only upon the queue of signals. This signal activates the waiting signal with the same name and then both signals get lost. The absorbing state is invented, too. If the process instance enters such a state, then it leaves the system.

The addition of such tools makes the interaction of process instances more exact and widens the application field of SDL. The observed extended SDL is directly used in system SDL/PL.

To develop the language SDL further, the original and the generalized states are introduced. The original state (Fig. 4) is invented for the creation of the very first process instance, which begins its actions at the initial time moment. This state needs not the associated inputs. The generalized state combines the functions of state and delay.

The realization of a generalized state is shown in Fig. 5a, and the proposed abbreviation - in Fig. 5b.

![Generalized State](image)

Fig. 3 The broadened state.

![Generalized State](image)

Fig. 4 The original form of the delay of process instance.

When using such a figure the process instance may proceed to the transition also in case, when no signal is accepted, but the delay time has run out. In this case the process instance leaves the state along the line or arrow, which has not associated input symbol. With the introduction of the generalized state there is no need for the special delay symbols. The delay then is only particular case of a state, when there are not associated inputs.

![Generalized State](image)

Fig. 5 The generalized state.

The expedience of the introduction
of both the original and the generalized states is corroborated by the experience of the design and application of the system SPALM85, which is the further development of the SPAL system [5,7], based on SDL.

According to the established traditions, the system SPALM85 offers the graphical symbols for the generalized state, the original state and the absorbing state, as well as for the decision, and they are different from those of SDL (Fig.6).

![Graphical symbols of SPALM85](image)

Fig.6 The graphical symbols of SPALM85. Moreover, SPALM85 does not distinguish external and internal signals (input, output). It uses only two types of signals (G,L). "Save" operation is not used, and the delay is not distinguished from generalized state.

There is zero state introduced in SPALM85. It serves as the source of process instances for generating signals. For frequently used diagram elements there are new designations (derivative actions). Fig.7 shows the source of process instances in its full (Fig.7a) and abbreviated form (Fig.7b).

![Diagram elements of SPALM85](image)

3. SIMULATION SYSTEM SPALM85

3.1 The Structure

Simulation System SPALM85 contains the specification language SITA, the programming language PAL85 and the simulation languages SITA-ITER and SITA-SIM (for Monte-Carlo simulation). A model in language SITA is represented in the form of a diagram, built from the actions of the extended SDL (the symbols shown in Fig.6 and Fig.7). The simulation languages SITA-ITER and SITA-SIM are designed in a common way (SITA-ITER is the subset of SITA-SIM), and the program in SITA-ITER may be used both for the algebraic calculations and simulation. The further discussion concerns the language SITA-SIM only. This language is the expansion of the programming language PAL85 by the procedures for simulation elements.

3.2 The Principles of PAL85

The program in PAL85 consists of statements, which are separated by blanks. Identifiers are used to designate arrays, blocks, procedures and other objects. Identifiers consist of letters only. Arithmetical and logical statements are designated by standard operators: "+,-,*,*", for arithmetical operations and "+,>,<,=,>,<" for logic operations. The operator "+" designates the transfer of control, as well as the address to a block or a procedure. Statements, containing "#" symbol, are called "titles" and are used to separate the logical parts of the program. They are also used as labels.

Let us consider the basic types of statements. The declaration part begins with the statement ":t=" and contains the declarations of all the variables and arrays, used in the program. The declaration statement begins with the identifier and contains the information on the type and dimensions of the array (in particular case it is single variable). The type REAL is designated by the operator "+", the INTEGER type - by the absence of "#". The declaration of the initial values consists of the symbol "=" and the number record. The initial values are declared for the array, declared by the preceding declaration statement.

The block head statement contains the block identifier and the symbol ":t=". The label statement consists of the number, which is called "label number", and of the character ":t=". The assignment statement has the form "variable=expression". There is abbreviated form for the assignment statement, where the left side is omitted. This gives the opportunity to write ":t=A+B" instead of ":t=A;:t=B;:t=A+B,". The value of the expression is assigned to the first variable in the expression. The variable from the array is designated by the identifier of the array and the number of the variable in this array. For the indexed variables the indices are given in the square brackets. The statement for control transfer to label has the form ":t=#label-number", the conditional transfer statement has the form "the-correlation-of-two-expressions #label-number". The transfer is executed when the correlation is satisfied. The omission of the label number means the exit from the current block (in case of both types of transfer statements). The address to block has the form ":t=#block-identifier". The address to procedure has the form of "parameter-identifier (parameter-list)". Parameters are separated by commas. Numbers, variables, blocks and labels may be used as parameters. In case of parameter-block or parameter-label this parameter be-
3.3 Simulation Procedures

The procedures for simulation operations are given in Table 1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Identifier</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generator (source)</td>
<td>GEN</td>
<td>Flow intensity</td>
</tr>
<tr>
<td>State</td>
<td>ST</td>
<td>State number, intensity of exits</td>
</tr>
<tr>
<td>Output of signal</td>
<td>OUT</td>
<td>Label of input statement</td>
</tr>
<tr>
<td>Input of signal</td>
<td>IN</td>
<td>State number</td>
</tr>
<tr>
<td>Transfer by prob-</td>
<td>PR</td>
<td>Probability, label for transfer</td>
</tr>
</tbody>
</table>

3.4 Model Examples

Let us consider two model examples, built in SITA-SIM.

Example 1. Fully available schema with limited waiting. The call flow with intensity \( L \) arrives in the schema with \( V \) lines. There are \( M \) places for waiting. The waiting call leaves the queue with intensity \( N \) and gets lost. The call, which arrives when all the places for waiting are full, gets lost, too. The average service time is equal to 1.

The diagram for this model in SITA language is shown in Fig. 8.

![Fig. 8](image)

Fig. 8 The diagram for fully available schema with limited waiting.

Simulation program uses the following standard designations: \( Q \) - the number of states, \( X \) - the employment vector for states, \( Y \) - the vector of state capacities, \( SET \) - the preparation block, \( ALG \) - the block for simulation algorithm. The program, including initial data \( V=3 \), \( L=2, M=4, N=1/2 \), is following:

```
:: V=3 L=2 M=4 N=1/2
1: #SP(1,1) #OUT(#3) #
2: X[2]=N # ST(#2,N) #
3: #IN(2) #
```

Example 2. The system with internal link, repeated calls and preliminary service. There are \( N \) subscribers. The free subscriber sends the call with intensity \( L/N \) to any other subscriber. There are \( V \) lines. The calling subscriber seize the line for the preliminary service with \( V \) average time \( 1/A \). If there is no free line available, or if the called subscriber is busy after preliminary service, then the call goes to the state, from which the repeated call is made with intensity \( N \) or the call gets lost with intensity \( S \). The repeated call is handled in the same way as the new one. If the called subscriber is available, then the conversation takes place between the two subscribers after the preliminary service. The average conversation time is equal to \( 1/B \). After this the subscribers and seized lines are released. Fig. 9 shows the diagram for the discussed model.

![Fig. 9](image)

Fig. 9 The diagram for the model with repeated calls.

The simulation program including initial data \( N=4, V=2, L=1/2, A=2, B=1, M=4, S=1/10 \), is the following:

```
:: N[4] V=2 L=1/2 A=2
Q=1 M=4 N=4
```
4. SIMULATION SYSTEM SDL/PL

4.1 General Information

The simulation system SDL/PL is the tool for the discrete event simulation of the complex systems, described or designed, using specification language SDL. This system is provided for the debugging of real time control programs by the method of simulation of the controlled equipment.

The system consists of the simulation language SDL/PL and its software. The language SDL/PL is built on the basis of the specification language SDL with supplements and the programming language PL/1. The same principles of the model design may be implemented as well as on the basis of other programming languages (for example, SIMULA-67 [8]).

4.2 The Structure of Simulation Program

Unlike the representation of the model in the form of a single module in system SPAIM85, the simulation program in SDL/PL contains the collection of the model components - process descriptions, which is supplemented by the model description and the set of variables for the processes of simulation. The declaration part consists of the DECLARE statements of PL/1 for all the variables in common use for all the process instances in this description.

The functional part corresponds to the SDL diagram for this PD, and describes the functional algorithm, common to all the process instances in this PD. Special statements are used for the representation of SDL actions, the syntax for those statements is close to the language SDL/PR [4] - the program form of SDL. Some actions, such as tasks and decisions, require flexible algorithmic facilities, and they are represented by PL/1 statements. Other SDL symbols are represented by Special Statements (STATE, INPUT, OUTPUT, DELAY, etc.). Taking into account such a way of model design, when the components of the model - process descriptions are developed by different programmers, PD provides special blocks for independent actions for the data input, the processing of Statistical operations, the output of the simulation results. To decrease the number of errors in the debugging process of the simulation program and its components, and to standardize these operations, the language SDL/PL provides special facilities for debugging. The principal difference from the other programming languages is the possibility to specify the debugging operations outside the functional algorithm - in the special DEBUG-block in PD. The sequence of the processing of such actions is controlled by the software according to the specifications stipulated by the user. When the debugging tools are not necessary, they are deleted automatically by the software.

4.2.2 Model Description

The model description serves for the consolidation of separate PD into united model. Special operations are provided for the description of model structure, data input/output and preparation sequence, the sequence of the starting of PD's by signals, the description of the initial state of model, the conditions and orders for applying the debugging tools, as well as the sequence of the experiment (the duration of time for it). Moreover, actions may be provided for various algorithmic purposes (preparation of data, calculation of the experiment conditions) in terms of PL/1 statements, including the calling of external procedures in other programming languages.

Several process instances of the same kind may interact in the model according to one PD. Each of those process instances may have the individual characteristics - the passport of process instance. The head of the PD records the name for the description and the set of variables for the passports of process instances. The declaration part consists of the DECLARE statements of PL/1 for all the variables in common use for all the process instances in this description.

The functional part corresponds to the SDL diagram for this PD, and describes the functional algorithm, common to all the process instances in this PD. Special statements are used for the representation of SDL actions, the syntax for those statements is close to the language SDL/PR [4] - the program form of SDL. Some actions, such as tasks and decisions, require flexible algorithmic facilities, and they are represented by PL/1 statements. Other SDL symbols are represented by Special Statements (STATE, INPUT, OUTPUT, DELAY, etc.). Taking into account such a way of model design, when the components of the model - process descriptions are developed by different programmers, PD provides special blocks for independent actions for the data input, the processing of Statistical operations, the output of the simulation results.

To decrease the number of errors in the debugging process of the simulation program and its components, and to standardize these operations, the language SDL/PL provides special facilities for debugging. The principal difference from the other programming languages is the possibility to specify the debugging operations outside the functional algorithm - in the special DEBUG-block in PD. The sequence of the processing of such actions is controlled by the software according to the specifications stipulated by the user. When the debugging tools are not necessary, they are deleted automatically by the software.

4.2.2 Model Description

The model description serves for the consolidation of separate PD into united model. Special operations are provided for the description of model structure, data input/output and preparation sequence, the sequence of the starting of PD's by signals, the description of the initial state of model, the conditions and orders for applying the debugging tools, as well as the sequence of the experiment (the duration of time for it). Moreover, actions may be provided for various algorithmic purposes (preparation of data, calculation of the experiment conditions) in terms of PL/1 statements, including the calling of external procedures in other programming languages.
PROCESS SOURCE;
DCL T_AVER FLOAT(6) STATIC;
/* AVERAGE TIME FOR INTERVAL */
STATE BEGIN;
INPUT START(L);
OUT: OUTPUT CALL(C);
DELAY INTERVAL ('EXP', T_AVER);
GOTO OUT;
ENDSTATE BEGIN;
/* THE BLOCK FOR DATA INPUT */
SET: GET LIST (T_AVER);
/* INPUT OF AVERAGE TIME */
ENDSET;
ENDPROCESS SOURCE;

b) PD in SDL/PL

Fig. 71 The description of process-source.

PROCESS CALLS PARM(NRCALL, LINENR);
DCL CALLNR BIN FIXED STATIC INIT(0);
DCL LINE(1000) STATIC INIT(0);
/* THE LINE CONTROL ARRAY */
DCL DELTIME STATIC /* SERVICE TIME */, ERR(2) BIN FIXED STATIC /* ERRORS */;
NR BIN FIXED(31);
STATE ARRIVAL;

INPUT CALL(C);
NRCALL=CALLNR; /* REMEMBER CALL NUMBER */
CALLNR=CALLNR+1; /* ACCOUNT OF CALLS */
CALL ATSLINE('SL', NRCALL, NR);
IF LINE(NR)=1
THEN DO;
/* ERROR-1 */
ERR(1)=ERR(1)+1;
PUT SKIP EDIT
('ERROR1-DOUBLE CONNECTION',
'ON LINE', NR)(A, A, P(4));
STATE OUT;
END;
LINE(NR)=1;
LINENR=NR; /* REMEMBER THE LINE NR. */
DELAY SERVICE ('EXP', DELTIME);
LINE(LINENR)=1;
CALL ATSLINE('RL', NRCALL, NR);
IF NR = LINENR
THEN DO;
/* ERROR-2 */
ERR(2)=ERR(2)+1;
PUT SKIP EDIT
('ERROR2-WRONG LINE RELEASED',
NR)(A, P(4));
END;
STATE OUT;
ENDSTATE ARRIVAL;
SET: GET LIST (DELTIME), ERR(2) ENDSET;
RESULT: /* BLOCK FOR SIMULATION RESULTS */
PUT SKIP EDIT ERR(1), 'DOUBLE CONNECTIONS',
ERR(2), 'WRONG RELEASED', P(4), A, SKIP;
ENDRESULT;
ENDPROCESS CALLS;

b) PD in SDL/PL

Fig. 12 The description of process-call.
4.3 The example: the debugging of the program for line selection

Let us consider the following example of the application of the system SDL/PL. Let us debug the program (from the control software of the telephone exchange), provided for the selection of line for the connection through the telephone exchange. Let us develop the model for the "environment" of such a program. In the model, the exponentially distributed flow of incoming calls arrives at the exchange, and in the moments of the establishment of the connection and of the ringing off the model calls the proper program in the control software (let us name this program ATSLINE). Three parameters are given to this program - the function ('SL' - seize line, 'RL' - release line), the number of the line and the parameter to return the number of the line to be seized or released. The model will consist of two process instances simultaneously. Each of process instances will remember in its passport the individual characteristics - the number of the call (NCALL) and the number of the line, chosen for this call (LINESR).

The PD of call will execute the diagnostic check for the program ATSLINE (the printout of messages and account of errors). Such a model (the model description is shown in Fig.13) may be used for the debugging of various algorithms of data maintenance and selection of line for the call. The implementation of the program ATSLINE may be in any programming language, which is accessible from PL/1.

MODEL DBGATS;
  INCLUDE SOURCE,CALLS;/*THE STRUCTURE*/
  MAKE SOURCE(BEGIN);/*CREATE SOURCE*/
  SET DATA INPUT/*DATA INPUT*/
  OUTPUT START(L);/*START THE SOURCE*/
  SIMULATE 50; /*EXPERIMENT LENGTH*/
  RESULT; /*OUTPUT OF RESULTS*/
ENDMODEL DBGATS;

Fig.13 The model description for the debugging of the line selection program.

4.4 The Software: Structure And Application

The software of SDL/PL is implemented in operation system OS for computers 2030 (the analogue of IBM/360). It contains: the preprocessor SDL/PL, which generates the PL/1 text for the model; the program for the analysis of PL/1 compiler output to bring it to accord to the listing of SDL/PL preprocessor; the programs for model control; the programs for debugging regime support; service programs.

The preprocessor of SDL/PL generates the program in PL/1 for each PD or model description (either in standard or debugging regime). Then the compiler of PL/1 works on the generated text. The further stages are the work of the program of analysis and link editing by the means of operation system. As a result, we obtain the load module either for process description, or model description, or the collection of them.

According to the principles of SDL/PL, the components of the model may be separately translated and debugged. The goal model is formed by the linkage editor of the operation system. At this stage the external procedures and the programs from the SDL/PL run-time software are added to the model.

5. CONCLUSIONS

The experience has shown, that the described in the paper supplemented SDL provides the convenient means for the description of the communication systems to be simulated. The simulation systems SPALL85 and SDL/PL are developed on the basis of the supplemented SDL.

SPALL85 system is applied to the investigations of teletraffic systems and their probability characteristics in cases when no program is involved (in other programming languages). The system SDL/PL is oriented for the simulation of complex systems and is used for complex debugging of the software of telephone exchanges with stored program control.

REFERENCES