A COMPUTER-AIDED SIMULATION ANALYSIS TOOL FOR SIMAN MODELS AUTOMATICALLY GENERATED FROM PETRI NETS

Albert Peñarroya, Francesc Casado and Jan Rosell
Institute of Industrial and Control Engineering
Technical University of Catalonia, Barcelona, Spain
E-mail: jan.rosell@upc.edu

KEYWORDS
Petri nets, SIMAN, Computer-aided tools.

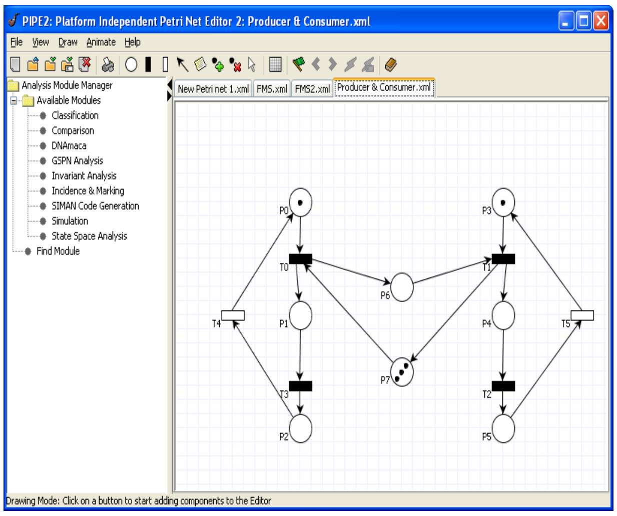
ABSTRACT

The analysis of the logic correctness of the system and its performance evaluation are usually carried out using, respectively, the Petri nets formalism and the discrete-event simulation. Several tools exist for both. The Platform Independent Petri Net Editor (PIPE) is a free software tool developed in Java for the modeling, simulation and qualitative analysis of Petri nets. It has been designed with an open philosophy so that extensions can be easily incorporated. SIMAN is one of the first discrete-event simulation languages developed. It has extensively proven its power. This paper first presents a module for the PIPE software that allows the automatic generation of SIMAN code from a Petri net. Then, a tool is proposed to aid the performance analysis of manufacturing systems from its SIMAN model. These tools are designed as a support for students in the understanding of the simulation methodology.

INTRODUCTION

The two main objectives when analyzing a manufacturing system are the evaluation of the logical correctness (i.e. the qualitative analysis) and the evaluation of its performance (i.e. quantitative analysis). Petri nets formalism and discrete-event simulation are used to carry out these objectives and, therefore, both must be included in the engineering students' curricula (Desel, 2000). Taking into account this, the objective of this paper is to introduce an aid to help students in the understanding of the use of simulation techniques as a methodology for the analysis of manufacturing systems.

Petri nets are a formalism that allows the modelling of systems involving concurrency, resource sharing, synchronization and conflict, and allows the validation of the correctness of the system by analyzing the qualitative properties of the net modelling the system (Murata, 1989). There are several software tools (see the Petri Nets World web www.informatik.uni-hamburg.de/ TGI/PetriNets/tools/) that allow the modeling, simulation and analysis of Petri nets.
The Platform Independent Petri Net Editor (PIPE, http://pipe2.sourceforge.net/), is a Java based, open



Drawing Mode: Click on a button to start adding components to the Editor

Figure 1 PIPE GUI

source, graphical tool for drawing and analyzing Petri nets developed at the Department of Computing at Imperial College London (Figure 1). Some of its available modules include invariant analysis, simulation, state space analysis and comparison and classification. New modules can be developed and easily incorporated.
SIMAN is a general purpose simulation language which incorporates special purpose features for modeling manufacturing systems (Pedgen, 1986). It is one of the best and first developed simulation languages extensively used. Taking into account this and the use of the extensibility property of PIPE, this paper will introduce both:

1. The development of a PIPE module able to automatically generate SIMAN code from a Petri net model of a system.

2. The development of a software tool to aid in the performance analysis of a system described with its SIMAN model. The tool must help the user in the specification of the warm-up period, the computation of the number of replication required, the validation process, the comparison between models, and the specification and execution of factorial designs.

As extra requirements the developed tools must be open source, developed in Java, and must provide the capability of executing the SIMAN models on a remote simulation server through the WEB. This will allow the

sharing of software and hardware resources over the Internet, independent of the user's platform (Guru et al., 2000).
After this introduction the paper is structured around two main sections describing, respectively, the PIPE module and the software tool for the simulation analysis.

PLATFORM INDEPENDENT PETRI NET EDITOR

Description

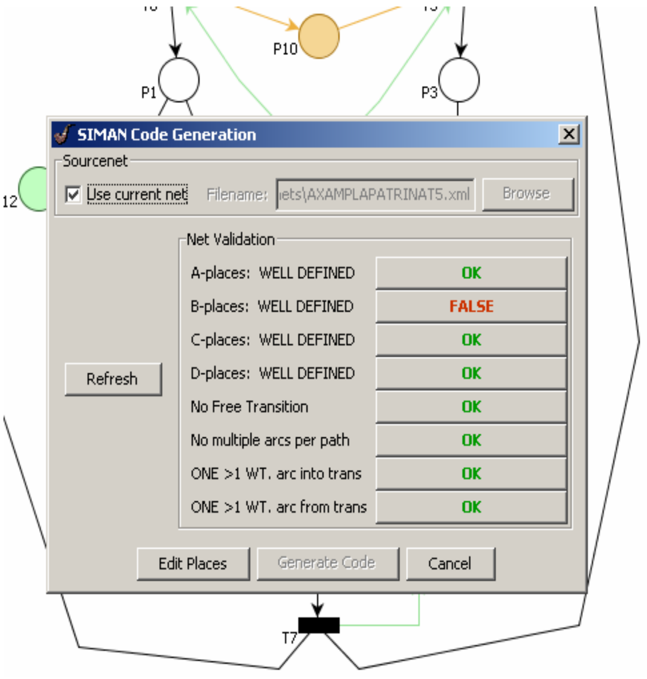
The Platform Independent Petri Net Editor (PIPE) is a graphical tool for the modelling and analysis of ordinary Petri nets. It allows invariant analysis, state space analysis and comparison and classification. PIPE also offers simulation capability that illustrates the token game through the evolution of the net markings.

Its modular architecture and open source philosophy allows the development of new features. In this paper we propose a module to automatically generate SIMAN code from Petri nets. This module requires some kind of coloring to the ordinary Petri nets managed by PIPE, as explained in the next subsection.

SIMAN code generation module

In order to include this new module, the following changes have been introduced to the basic PIPE functionality. First, the capacity to distinguish between different types of places: type-A places to represent activities, type-B places to represent finite resources like machines or robots, type-C places for control places, and type-D places to represent the system input or variable resources like pallets or fixtures. Second, the capacity to introduce code into the net places in order to specify some parameters and values needed when translating to SIMAN.

In order to generate SIMAN code from a Petri net first it is necessary to specify the type of places and the initial marking. Then the code associated to each place must be introduced. For type-A places it is necessary to specify the delay time of the activity; for type-B places the time between failures and the downtimes; for type-C places the group to which they pertain since type-C places are grouped in sets; for type-D places the time between arrivals whenever they represent the system input. Moreover if there is a conflict in type-A or type-D places it is necessary to specify how it is to be solved (i.e. by chance or by the type of entity which is defined in the corresponding type-D place).



Sourcenet
☑ Use current net Filename: ets\AXAMPLAPATRINAT5.xml Browse

Net Validation

A-places: WELL DEFINED	OK
B-places: WELL DEFINED	FALSE
C-places: WELL DEFINED	OK
D-places: WELL DEFINED	OK
No Free Transition	OK
No multiple arcs per path	OK
ONE >1 WT. arc into trans	OK
ONE >1 WT. arc from trans	OK

Figure 2 SIMAN Code Generation Module

Once the Petri net is defined, the SIMAN code generation module can be executed (Figure 2). The module first makes a validation of the net in order to avoid future parsing problems. This validation consists in the following verifications:

1. Type-A places: Existence of at least an input arc and initial marking set to zero.
2. Type-B places: Non-null initial marking.
3. Type-C places: Specification of a family group and non-null initial marking per group.
4. Type-D places: Existence of at least one type-D place, existence of at least one output arc and either the existence of a non-null initial marking or the specification of the time between arrivals.
5. Non-existence of isolated transitions or places.
6. Non-existence of repeated arcs.
7. Non-existence of multiple input arcs if there are non-unitary weights.
8. Non-existence of multiple output arcs if there are non-unitary weights.

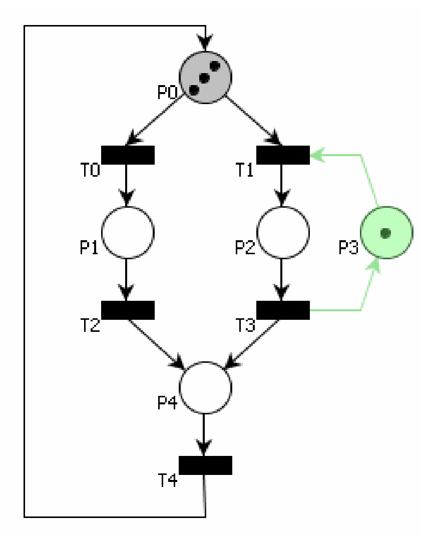


Figure 3 Example Net

Steps 1 to 5 are strictly necessary for the correct parsing to a SIMAN model. Step 6 solves a PIPE bug, and steps 7 and 8 greatly simplify the parsing process.

Whenever one of these steps fails, the module warns the user indicating where the problem is located. Figure 2 shows a case where there is a problem with the initial marking of type-B places. Once the problems are solved, the Generate Code button is activated and when pressed the SIMAN code is generated and a window is opened with the contents of the MOD and EXP SIMAN files which are, respectively, the model and the experiment components that correspond to the logic and data in the model. These files can then be stored to disk.

An Example

The following simple example illustrates some of the features of the SIMAN code generation module. The Petri net is shown in Figure 3. It is a cyclic net where type-D place P0 represents the availability of three parts to be processed. Type-A places P1, P2 and P4 represent three different activities. Parts are processed either by P1 and P4 or by P2 and P4. Activity P2 requires the use of the resource represented by type-B place P3.

The following code is introduced into the net places:
Place P0: Code indicating that the conflict is solved by chance (20% of parts go to place P1; 80% to place P2).

```
#
decide = probability
@T0=0.8
@T1=0.2
#
```

Place P1: Code indicating the delay time.

```
#
delay = EXPO(1.1)
#
```

Places P2 and P4: Code indicating the delay time, EXPO(0.5) and EXPO(2), respectively, in a similar way as place P1.
Place P3: Code indication the time between failures and the downtime.

```
#
failures
@ timeON = GAMMA(7,15)
@ timeOFF = GAMMA(2,3)
#
```

The SIMAN code obtained is shown in Table 1 and Table 2. The *MOD* file includes the program flow while the EXP file includes the definitions of the variables and resources.

Table 1. SIMAN MOD File

```
1$   CREATE,   3,HoursToBaseTime(0.0),Entity P0:
     HoursToBaseTime(1),1: NEXT(2$);
2$   ASSIGN:   Create P0.NumberOut = Create
     P0.NumberOut + 1:NEXT(3$);
3$   BRANCH,   1:
        With,0.8,4$,Yes:
        With,0.2,5$,Yes;
4$   DELAY:    0: NEXT(6$);
5$   QUEUE,    Seize T1.Queue;
7$   SEIZE,    2,Other:
        Resource P3, 1:NEXT(8$);
6$   DELAY:    EXPO(1.1):NEXT(9$);
9$   DELAY:    0: NEXT(10$);
8$   DELAY:    EXPO(0.5):NEXT(11$);
11$   RELEASE:
          Resource P3, 1:NEXT(10$);
10$   DELAY:   EXPO(2):NEXT(12$);
12$   DUPLICATE:
        1,3$;
     ASSIGN:   Dispose T4.NumberOut=Dispose
        T4.NumberOut+1;
     DISPOSE: Yes;
```

Table 2. SIMAN EXP File

```
PROJECT,  "Unnamed Project", "Siman Code
 Generation" ,,,No,Yes,Yes,Yes,No,No,No,No,No;
FAILURES:
 Failure 0,Count(GAMMA(7,15),GAMMA(2,3));
RESOURCES:
 Resource P3,Capacity(1),,,,, FAILURE(Failure
0,Ignore),;
REPLICATE,   10,,HoursToBaseTime(160),Yes,Yes,,,,
 24,Hours,No,No,,,Yes;
VARIABLES:
 Create P0.NumberOut, CLEAR(Statistics), CATEGORY
 ("Exclude"): Dispose T4.NumberOut,
CLEAR(Statistics),
  CATEGORY("Exclude");
ENTITIES:
 Entity P0;
QUEUES:
 Seize T1.Queue,FIFO,,AUTOSTATS(Yes,,);
DSTATS:
 Create P0.NumberOut, aCreate P0.NumberOut:
 Dispose T4.NumberOut,aDispose T4.NumberOut;
```

CASA - COMPUTER AIDED SIMULATION ANALYSIS

Specification

The CASA software has as objective the aid in the performance analysis of discrete-event simulations. The program has different features like model validation or factorial design. The results of the simulations are obtained by executing the SIMAN models, or in some cases they can be provided by data text files. The ARENA simulation software (www.arenasimulation.com) must be available in order to use its SIMAN compiler, linker and simulation programs. The input SIMAN files can be either generated by the PIPE module described in the previous section or by the ARENA software.

The CASA software has the following options: model simulation, specification of the warm-up period, computation of the number of replications required, model validation, comparison between two models and factorial design. Theoretical expressions used are obtained from (Banks et al. 1996) .

Model simulation

Once the SIMAN model is loaded, the simulation utility allows the simulation of the model. A command window is opened where the different calls to the compilation, linking and execution programs are automatically performed. The output data is shown in text format.

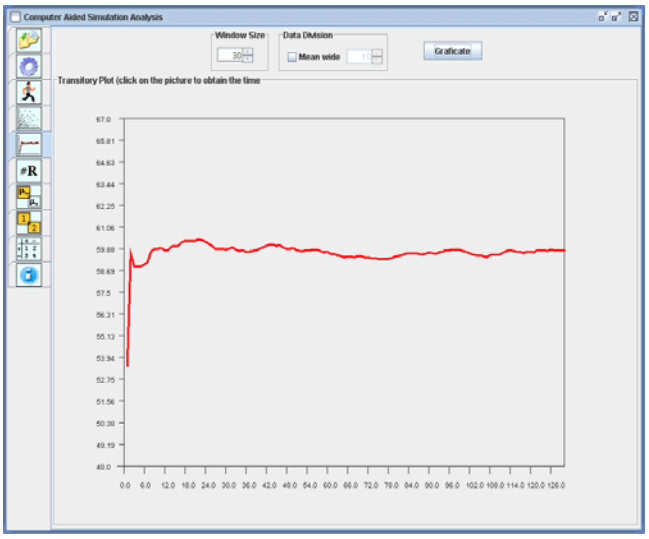
Specifying the warm-up period

This option allows the determination of the warm-up period. The MOD file is shown to the user where he specifies the variable to be used for determining the warm-up period. It can either be a variable computed at the exit of a block (e.g. parts produced) or a variable computed between to blocks (e.g. time-in-queue or WIP). In either case, the software allows the use of the moving average method to filter the data, using a chosen window size *w*:

$$\tilde{Y}_i = \begin{cases} \dfrac{\sum_{s=-w}^{w} \tilde{Y}_{i+s}}{2w+1} & if \;\; i = w+1,\ldots,m-w \\[2ex] \dfrac{\sum_{s=-(i-1)}^{i-1} \tilde{Y}_{i+s}}{2i-1} & if \;\; i = 1,\ldots,w \end{cases} \qquad (1)$$

Computing the number of replications required

This option computes the number of replications needed in order to obtain confidence intervals with a specified precision. If the desired half-width of the confidence interval is ε, then the following iterative procedure is



Window Size: 30

Figure 4 Warm-up Period Panel

programmed to compute the required number of replications:

1) Starting with R_0 replications, estimate σ^2 by S_0^2:

$$R \geq \left[\frac{t_{\frac{\alpha}{2},(R-1)} S_0}{\varepsilon} \right]^2 \qquad (2)$$

2) Estimate a first value of R by substituting

$t_{\frac{\alpha}{2},(R-1)}$ by $z_{\frac{\alpha}{2}}$ in expression (2).

3) Increment R until (2) is satisfied (using $t_{\frac{\alpha}{2},(R-1)}$).

The program shows a text window with the final result and all the intermediate results of each iterative step performed.

Model Validation

This option is an aid for the validation of the model. The real system data used for the validation is the average of one of the performance measures selected by the user (μ_0). The statistical *t*-test is performed . First it determines:

$$t_0 = \frac{\bar{y} - \mu_0}{S / \sqrt{n}} \qquad (3)$$

Then, if $|t_0| < t_{\frac{\alpha}{2},(n-1)}$ the model is accepted with a probability α of having rejected a valid model. Then, the probability of having accepted a non-valid model is computed as (Ferris et al. 1946):

$$\beta = e^{-\frac{1}{2}n\lambda^2} \sum_{r=0}^{\infty} \frac{\left(\frac{1}{2}n\lambda^2\right)^r}{r!} I\left[(r+1/2), \frac{1}{2}(n-1); \frac{t_\varepsilon^2}{n-1+t_\varepsilon^2}\right] \tag{4}$$

where λ is the allowed difference between the model and system means, *n* is the number of replications and I(p,q;x) is the incomplete beta function. This value is computed graphically in (Banks, 1996).

The user can specify a desired maximum risk β and then the program outputs the number of replications required to achieve it.

Comparison between two models

This option allows the comparison between the loaded SIMAN model and another one that is loaded when the comparison module is opened. The comparison is done considering independent sampling. The user specifies which variables are to be compared and then the simulation of the models is performed. Then the confidence interval of the difference of means is computed:

$$\left(\left(\overline{Y}_1 - \overline{Y}_2\right) \pm t_{\frac{\alpha}{2},\nu} \cdot s.e.\left(\overline{Y}_1 - \overline{Y}_2\right)\right) \tag{5}$$

Whenever this confidence interval contains zero there is not strong statistical evidence that one system design is better than the other.

To compute this confidence interval, the test of equal variances is performed. This test uses the Fisher-Snedecor distribution, i.e. if :

$$F = \frac{S_2^2}{S_1^2} < F_{\alpha, R_1-1, R_2-1} \tag{6}$$

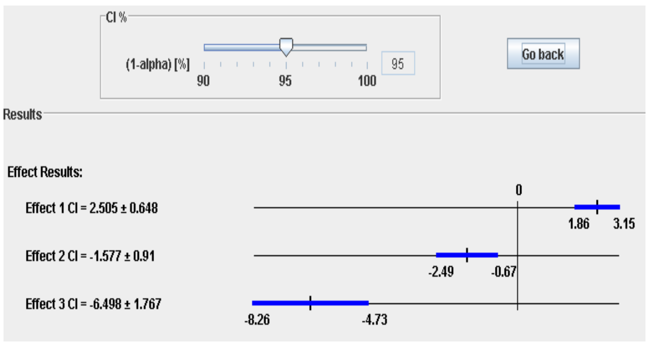
then both variances are considered equal. In this case the standard error of the difference is computed as follows:

$$S_P = \frac{(R_1-1)\cdot S_1^2 + (R_2-1)\cdot S_2^2}{R_1+R_2-2} \tag{7}$$

$$s.e. = S_P \cdot \sqrt{\frac{1}{R_1} + \frac{1}{R_2}} \tag{8}$$

And the degrees of freedom are: $\nu = R_1 + R_2 - 2$.

Otherwise, when variances are considered unequal, standard error of the difference is computed as follows:



Results

Effect Results:

Effect 1 CI = 2.505 ± 0.648

Effect 2 CI = -1.577 ± 0.91

Effect 3 CI = -6.498 ± 1.767

Figure 6 Detail of the Factorial Results

$$s.e. = \sqrt{\frac{S_1^2}{R_1} + \frac{S_2^2}{R_2}} \tag{9}$$

and the degrees of freedom are:

$$\nu = \left\lfloor \frac{\left(\frac{S_1^2}{R_1} + \frac{S_2^2}{R_2}\right)^2}{\frac{\left(S_1^2 / R_1\right)^2}{(R_1-1)} + \frac{\left(S_2^2 / R_2\right)^2}{(R_2-1)}} \right\rfloor \tag{10}$$

As in other program options, all the computations are shown to the user in a text window.

Factorial design

The last option of the program allows the performance of a factorial design using up to six factors. Each factor is assigned two values. The user selects the number of factors to consider and introduces two values to be considered for each of them. The program automatically computes all the possible combinations (design points) and executes the corresponding replicates of each one.

This option graphically outputs the confidence intervals of each of the principal effects of the chosen factors (Figure 6). When the confidence interval does not include zero then the factor is considered significant.

CONCLUSIONS

The availability of software tools for the understanding of the simulation methodology in the analysis of manufacturing systems is a key aspect for engineering studies. This paper has proposed two tools that cover both Petri nets and discrete-event simulation.

First, a module that automatically generates SIMAN code from Petri nets has been incorporated to the Platform Independent Petri Net Editor (PIPE), an open source graphical tool for drawing and analyzing Petri nets. Although PIPE allows the simulation of Petri nets, the translation to SIMAN allows a better simulation of manufacturing systems since the new incorporated module permits, among other features, the introduction of different time distributions or the definition of failures. Moreover, the obtained SIMAN code facilitates the use of the second tool for the performance analysis of the system.

The second tool introduced in this paper is the software CASA (Computer Aided Simulation Analysis). It has been developed as an aid in the performance analysis of manufacturing systems modeled using SIMAN. It has several features not encountered in other simulation packages, like the capability of performing factorial designs or model validation.

Both tools are currently being used in undergraduate courses at the Industrial Engineering School of Barcelona (Technical University of Catalonia). They are available at lafarga.cpl.upc.edu/.

REFERENCES

Banks, J., Carson J. and B. Nelson, 1996. *Discrete-Event System Simulation.* Prentice-Hall, Upper Saddle River, NJ, USA.

Desel, J. 2000. “Teaching System Modelling, Simulation and Validation”, in *Proceedings of the 2000 Winter Simulation Conference*, pp. *1669-1675*.

Ferris, C.L., Grubbs, F. E. and C. L. Weaver. 1946. “Operating Characteristics for the Common Statistical Tests of Significance*,” Annals of Mathematical Statistics*, June 1946. The Institute of Mathematicals Statistics

Guru, A., P. Savory and R. Williams. 2000. “A web-based interface for storing and Executing Simulation Models”, in *Proceedings of the 2000 Winter Simulation Conference*, pp. *1810-1814*.

Murata, T. 1989. "*Petri Nets: Properties, Analysis and Applications*," Proc. IEEE, vol. 77, No. 4 (Apr.), pages. 541-580.

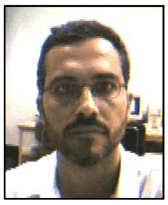
Pedgen, C. D. 1986. “Introduction to SIMAN”, in *Proceedings of the 1986 Winter Simulation Conference*, pp. *95-103*.

ACKNOWLEDGEMENTS

This work was partially supported by the Comisíon Interministerial de Ciencia y Tecnología (CICYT) projects DPI2004-03104 and DPI2005-00112.

ALBERT PEÑARROYA was born in Barcelona, Spain and went to the Technical University of Catalonia, where he studied at the Industrial Engineering School of Barcelona and obtained his degree in 2006.

FRANCESC CASADO was born in Barcelona, Spain and went to the Technical University of Catalonia, where he studied at the Industrial Engineering School of Barcelona and obtained his degree in 2006.



JAN ROSELL received his B.S. in telecomunication engineering and Ph.D. in advanced automation and robotics from the Technical University of Catalonia, Barcelona, Spain, in 1989 and 1998, respectively. In 1992 he joined the Institute of Industrial and Control Engineering, where he has developed research activities in robotics. He has been involved in teaching activities in automatic control and modelling and simulation as an assistant professor since 1996 and as an associate professor since 2001. His current technical areas include automatic programming, robotic assembly, and modelling and simulation of manufacturing systems.