MANAGING THE RESOLUTION OF SIMULATION MODELS

Gábor Lencse
Department of Telecommunications
Széchenyi István University
Egyetem tér 1.
H-9026 Győr, Hungary
e-mail: lencse@sze.hu

László Muka
Elassys Consulting Ltd.
Bég utca 3-5.
H-1022 Budapest, Hungary
e-mail: muka.laszlo@elassys.hu

KEYWORDS

discrete-event simulation, model resolution management, information and communication technology systems, business process systems

ABSTRACT

A novel approach based on inflation and deflation is proposed for managing the resolution of simulation models. Different methods are proposed for manual or automatic deflation. An example is given how a topology description language can be extended to support the inflation/deflation concept. Dynamic management of the model resolution is introduced using the method called inflate-the-next and also two of its possible improvements.

INTRODUCTION

Resolution of the Models

Discrete-Event Simulation (DES) is a widely used method for the performance analysis (Jain 1991; Pidd 1991) of Information and Communication Technology (ICT) systems and Business Process (BP) systems. There is a large number of various methods used to describe the behaviour of complex systems (Banks et al. 1996; Bratley et al. 1986; Jávor 1985; Jávor 1993). The simulation of large and complex systems requires a large amount of memory and computing power that is often available only on a supercomputer or on computer grid. The resolution of the models is an important factor from the point of view of efficiency of simulation: if the resolution is too low the necessary results cannot be reached but if the resolution is too high then we loose both modelling work and computing time. It seems to be logical that the most efficient resolution of a model is a compromise between these two requirements. Time decomposition method described in (Muka and Lencse 2007) may help the modeller to choose the relevant systems to be modelled, to manage the appropriate model resolution and to support the decision about parallel and sequential simulation. The appropriate level of resolution may be found by applying a set of transformations to the different parts of the model. However, it is many times desirable that the resolution of the model can be adjusted *dynamically*. Why? On the one hand, the modeller may not have enough information for determining the optimal resolution in the model building stage – the situation may improve later during experimenting with the model. On the other hand, the different experiments performed on the model may require different resolution of the different parts of the model.

In this paper, it is shown how the resolution of the simulation models can be made volatile giving freedom to the modeller to set the appropriate resolution for each part of the model for each experiment individually.

The remainder of this paper is organised as follows: first, the idea of inflating and deflating is presented, second, some methods for deflation are presented, third, an example is given how a topology description language can be extended to support the inflation/deflation concept, fourth, the dynamic management of the resolution is introduced.

THE IDEA OF INFLATION AND DEFLATION

Before presenting the inflation/deflation concept, we briefly consider the possible elements of the models we deal with.

Modelling Concept

In order that our results can be widely used in the world of modelling and simulation of information and communication technology (ICT) and business process (BP) systems, we do not intend to make any unnecessary restrictions on the models. We believe that it is a rational expectation that a contemporary modelling and simulation environment should give the possibility of hierarchical system description. We call the key element of the hierarchical description *Compound Module*. A compound module may contain further compound or simple (that is: not compound) modules and other elements depending on the modelling and simulation environment.

Changing the resolution by inflation/deflation

To support the flexibility of the resolution of our models, we simply add a new attribute (some modelling systems call it parameter) named *Resolution* to all the compound modules. This attribute is of an enumerated type and may take the values either *Deflated* or *Inflated*. These are the *states* of the compound module. How do we interpret these states? If a compound module is inflated, its internal structure as well as the operation of its parts are modelled in detail. (In the same way as it is done without the resolution attribute.) If it is deflated, we *ignore its internal structure and operation* and *imitate its behaviour* for the outside world by a simpler algorithm that acts similarly but of course not completely the same as the original compound module. As a compound module may contain several elements – even compound modules of arbitrary hierarchical levels –, we expect that the simulation of a compound module in the deflated state *requires much less computing power* and *models the real world system less precisely* than in its inflated state. The advantage of the flexibility is that the modeller may decide experiment by experiment how much precision is necessary in the modelling of the given compound module. In this way we must pay the price (in computing power) of the necessary precision only.

The concept above means that at a given hierarchical level the modeller may decide about all the compound modules to be inflated/deflated as he/she wishes, but if a compound module is deflated, no more decision can be made about the contained compound modules. However, if a compound module is inflated, some of its contained compound modules may still be deflated, as due to encapsulation the compound module "knows" nothing about them. Thus, it is meaningful to define a new expression: a compound module is *fully inflated* if it is inflated and all the contained compound modules are fully inflated. (Illustrated in Fig. 1.)

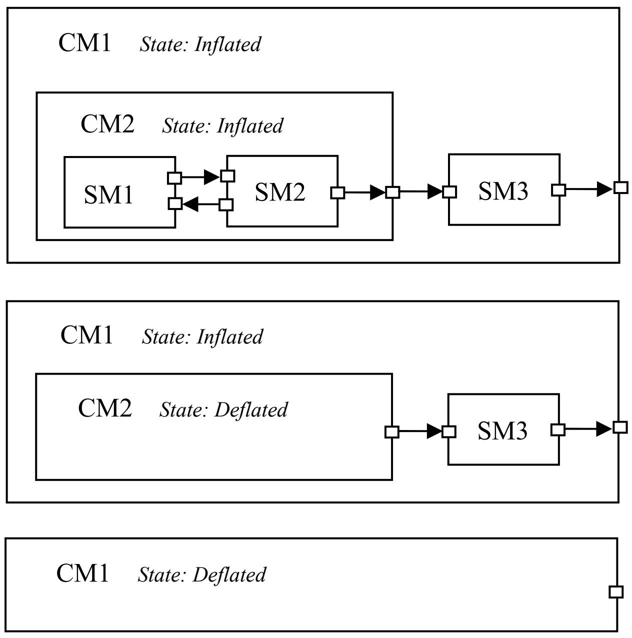

Figure 1. CM1 is: a) Fully Inflated, b) Inflated, c) Deflated

Manual or automatic deflation

In the simplest case, the modeller has to write two descriptions (codes) for each compound module: one for its inflated state and one for its deflated state. We call this approach *manual deflation* and it has the following advantages: the modeller has full control over the model in both states and no extra features are required from the modelling and simulation system. The price is that the modeller has to make two models for each compound module.

It is possible that the modeller prepares the model of the compound module for the inflated case only. We will show different methods that may be suitable for *automatic deflation*, where the model of the compound module in the deflated state is generated automatically. The advantage of this approach is that the modeller does not have to work on the model of the compound module in the deflated state. This approach requires either support from the modelling and simulation system or some extra work from the modeller, however this extra work probably may be automated quite well. The details depend on the methods used for automatic deflation.

METHODS FOR DEFLATION

The methods presented here are based on our previous research results in modelling and simulation. Even though the essence of these results will be briefly summarized (as space permits) the reader is encouraged to read the original papers to get deeper understanding of the methods used here.

Substitution by Statistical Interfaces

The basic idea of this solution has its roots in the Statistical Synchronisation Method (SSM) invented by György Pongor (Pongor 1992) and further developed under the name SSM-T by Gábor Lencse (Lencse 1998a).

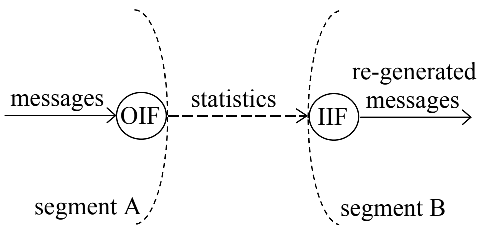

Figure 2. An OIF - IIF Pair

Statistical Synchronisation Method

SSM is a Synchronisation Method for Parallel Discrete Event Simulation (PDES). A synchronisation method is responsible for keeping causality between the virtual times of the segments. More well-known methods for this task are the conservative and the optimistic methods – more information on them can be found in (Fujimoto 1990). Now, we summarize the essentials of SSM in a nutshell. Similarly to other parallel discrete-event simulation methods, the model to be simulated — which is more or less a precise representation of a real system — is divided into segments, where the segments usually describe the behaviour of functional units of the real system. The communication of the segments can be represented by sending and receiving various messages. For SSM, each segment is equipped with one or more input and output interfaces. The messages generated in a given segment and to be processed in a different segment are not transmitted there but the *output interfaces* (OIF) collect statistical data of them. *The input interfaces* (IIF) generate messages for the segments according to the statistical characteristics of the messages collected by the proper output interfaces. (See Figure 2.) The segments with their input and output interfaces can be simulated separately on separate processors, giving statistically correct results. The events in one segment do not have the same effect in other segments as in the original model, so the results collected during SSM are not exact. The precision depends on the partitioning of the model, on the accuracy of statistics collection and regeneration, and on the frequency of the statistics exchange among the processors.

Deflation with Statistical Interfaces
It is natural that SSM can be applied in the following way: the contents of the compound module will create one segment and all the other parts of the model will create the other segment. In this way SSM is applied at the boundaries of the compound module. Let us consider the message routes that lead from inside the compound module to the outside world. Let us wait until all the OIFs collect enough data for their first statistics. They send the statistics to the proper IIF that will regenerate the message flow for the outside world. Now the simulation of the inside part of the compound module may be stopped and the IIFs will still sustain the message flow. Of course, this approach works only in the steady state of the modelled system. This is what the original SSM can be used for.

Non-steady State Behaviour
Here, SSM should be replaced by SSM-T. Besides the one that was mentioned before, we have published a number of papers on the different issues of this method, such as dealing with its statistics collection methods (1998b) applicability criteria (Lencse 1999a) and statistics exchange control algorithm (Lencse 1999b). Using these results, it seems to be possible both to automatically stop the detailed simulation of the inside parts of the compound module and to sustain the message flow, and also automatically restart it when necessary. This topic can be a subject of further research.

Substitution by Flow-based Methods

By flow-based methods we mean the Traffic-Flow Analysis (Lencse 2001) for ICT systems and the Entity Flow-Phase Analysis (Lencse and Muka 2006) for BP systems.

Traffic-Flow Analysis
The Traffic-Flow Analysis (TFA) is a simulation-like method for fast performance analysis of communication systems. TFA uses statistics to model the networking load of applications.

In the *first part*, the method distributes the traffic (the statistics) in the network, using routing rules and routing units.

In the *second part*, the influences of the finite line and switching-node capacities are calculated.

The important features of TFA:

- The results are approximate but the absence or the place of <u>bottlenecks</u> is shown by the method.
- The execution time of TFA is expected to be significantly less than the execution time of the detailed simulation of the system.
- TFA describes the steady state behaviour of the network.

As TFA is a less well-known method, it has only one partial implementation, which is a part of the ImiNet network expert system (Elassys 2008).

Entity Flow-Phase Analysis
The Entity Flow-phase Analysis has been derived from TFA. This derivation is based on the formal similarity of the ICT and BP models. EFA uses the same two phase method as TFA, only the interpretation of the model elements is different. The statistics represent entities (not messages) and the interpretation of the routing is also different. While the packets of a network usually do not multiply, the entities may fork (and the descendants must meet somewhere) or split (and the descendants live their own life separately); see more details in the aforementioned paper.

An implementation of EFA is planned as an extension for the ImiFlow system (Elassys 2008).

Deflation with TFA or EFA
On the basis of our results presented in (Lencse and Muka 2007) it is trivial that the detailed simulation of the internal parts of a compound module can be replaced by TFA for ICT systems and by EFA for BP systems. The different modelling methods still should work together. Thus the manual deflation seems to be quite simple. For the automatic deflation, we need a method to automate the transformation of the detailed ICT or BP models to TFA or EFA models, respectively. This topic seems to be a very interesting and promising research area.

TOPOLOGY DESCRIPTION LANGUAGE SUPPORT

We show an example how a network topology description language can be extended to support the inflation/deflation concept. We selected the NeD language[1] (Varga and Pongor 1997) for this purpose. The EBNF description of the compound module is the following:

```
moduledefinition ::=
    module compoundmoduletype
      [ paramblock ]
      [ gateblock ]
      [ submodblock ]
      [ connblock ]
    endmodule [ compoundmoduletype ]
```

Now we extend it in the following way:

```
moduledefinition ::=
    module compoundmoduletype
      [ paramblock ]
      [ gateblock ]
      [ submodblock ]
      [ connblock ]
      [ deflatedblock ]
    endmodule [ compoundmoduletype ]
```

[1] The EBNF grammar of the NeD language can be found in the User Manual of OMNeT++ (http://www.omnetpp.org).

```
deflatedblock ::=
    deflated:
      [ paramblock ]
      [ gateblock ]
      [ submodblock ]
      [ connblock ]
    enddeflated
```

In simple words, the EBNF description above means that after the optional keyword "**deflated:**" the modeller may give the description of the compound module for the deflated case by the same way (using the same types of blocks) as he/she could describe the contents of the compound module for the inflated case. (And the description for the inflated case remained the same as it was originally.) As for the **Resolution** attribute, the NeD grammar requires no modification, the modeller can provide a parameter with this name. Let us see the EBNF of the parameter description:

```
paramblock ::=
    parameters: { parameter ,,, } ;

parameter ::=
    parametername
    | parametername : const [ numeric ]
    | parametername : string
    | parametername : bool
    | parametername : char
    | parametername : anytype
```

If we do not insist on the enumerated type with the values *Deflated*, *Inflated*, it seems to be a very simple solution to implement these values by logical values of the Boolean type: *false* and *true* respectively.

RESOLUTION MANAGEMENT

Now, we introduce a highly cost effective solution for simulation of large and complex systems. Let us suppose that we have the detailed model of the system containing many elements that would require a huge cluster of processors for simulation experiments, but we have only limited computing capacity.

Inflate-the-next Method

At the highest hierarchy level of the model, we divide the set of modules into two parts: the set of inflated modules and the set of deflated modules. The set of inflated modules should contain all the modules that we focus on at a given step of the given experiment. And all the rest of the modules are deflated and put in the deflated set of modules. In a simple case, we assign the two sets of modules to two separate processors for execution. (Of course, an arbitrary number of processors may be used.) Between the processors we may use any synchronisation methods for PDES mentioned before. For the next step of the experiment, we deflate the contents of the inflated set of modules and move them into the deflated set of modules and we select to inflate the next set of modules from the deflated set of modules and inflate them and put them into the inflated set of modules. This is why we call this algorithm *inflate-the-next*. The experiment ends when all the relevant sets of modules have been inflated and executed. See Figure 3.

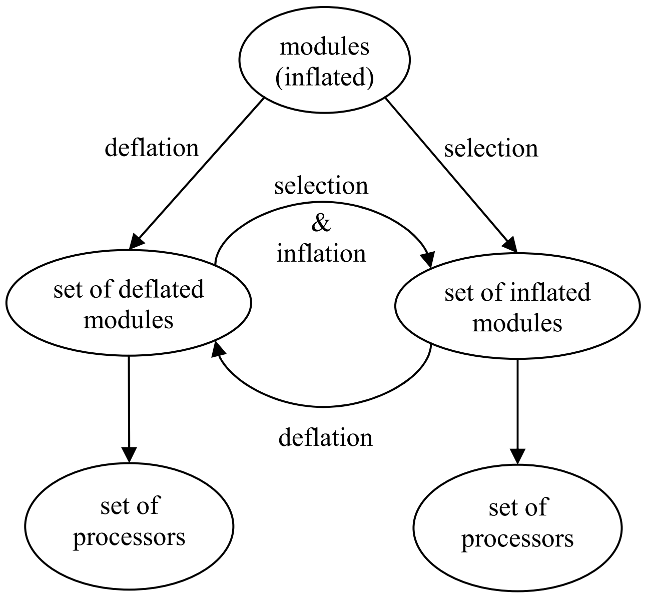

Figure 3. Inflate-the-next Method

The Order of Selection

We may have two approaches:

- If we choose a set of compound modules to be inflated with a minimum number of connections to the rest of the modules then hopefully we will get relatively precise results for the inflated set.
- If we choose a set of compound modules to be inflated with the maximum number of connections to the rest of the modules then the results of execution can be better used to model them when they are deflated.

Using these considerations, the inflate-the-next algorithm can be improved in the following way. All the experiments should contain two phases: in the first phase we use the approach with maximum number of connections to improve the precision of the model, in the second phase we use the approach with the minimum number of connections to increase the precision of the simulation results.

The Size of the Inflated Set

On the one hand, the size of the inflated set is limited by memory and computing power; on the other hand, possibly all the modules with intensive communication with each other should be put into a single inflated set of modules. This idea has at least two advantages:

- It eliminates the problem of decreasing precision that would be caused by putting modules with intense communication with each other into separate inflated set of modules.

- It speeds up the parallel simulation of the two sets by decreasing the number of messages between them.

This consideration should be used together with the previous one.

Directions of Future Research

The ideas presented in this paper seem to be promising. Both the research of further algorithms for deflation and the investigation of the presented ones can be a hot research topic. The resolution management algorithm and its improvements should be tested and refined.

CONCLUSIONS

We have introduced a concept based on inflation and deflation for maintaining the optimal resolution of simulation models.

We have presented different methods for manual and automatic deflation.

We have shown a way how the grammar of a typical topology description language can be modified to support the inflation/deflation concept.

We have given a resolution management method called inflate-the-next as well as two possible improvements for this method.

We conclude that both the topic of resolution management and the methods we presented deserve further research.

REFERENCES

Banks, J.; J. S. Carson and B. L. Nelson. 1996. *Discrete-Event System Simulation* Prentice Hall, Upper Saddle River, New Jersey

Bratley P.; B. L. Fox and L. E. Schrage. 1986. *A Guide to Simulation.* Springer-Verlag, New York

Elassys Consulting Ltd. 2008. *ImiNet and ImiFlow Systems* http://www.elassys.hu

Fujimoto, R. M. 1990. “Parallel Discrete Event Simulation” *Communications of the ACM* 33, no 10, 31-53

Jain, R. 1991. *The Art of Computer Systems Performance Analysis..* John Wiley & Sons, New York

Jávor, A. (editor) 1985. *Simulation in Research and Development.* North-Holland, Amsterdam

Jávor, A. 1993. *Petri Nets in Simulation* EUROSIM Simulation News Europe, 1993, no. 9, pp. 6-7.

Lencse, G. 1998a. “Efficient Parallel Simulation with the Statistical Synchronization Method” *Proceeding of the Communication Networks and Distributed Systems Conference* (CNDS'98), (San Diego, CA, USA, January 11-14) SCS, 3-8

Lencse, G. 1998b. “Statistics Collection for the Statistical Synchronisation Method” *Proceedings of the 10th European Simulation Symposium* (ESS'98) (Nottingham, England, Oct. 26-28.) SCS-Europe, 46-51

Lencse, G. 1999a. “Applicability Criteria of the Statistical Synchronization Method” *Proceedings of Communication Networks and Distributed Systems Conference* (CNDS'99), (San Francisco, CA, USA, January 17-20.) SCS, 159-164

Lencse, G. 1999b. “Design Criterion for the Statistics Exchange Control Algorithm used in the Statistical Synchronization Method” *Proceedings of the 32nd Annual Simulation Symposium* (San Diego, CA, USA, April 11-15.) IEEE Computer Society, 138-144

Lencse, G. 2001. “Traffic-Flow Analysis for Fast Performance Estimation of Communication Systems” *Journal of Computing and Information Technology* 9, No. 1, 15-27.

Lencse, G. and L. Muka. 2006. “Expanded Scope of Traffic-Flow Analysis: Entity Flow-Phase Analysis for Rapid Performance Evaluation of Enterprise Process Systems” *Proceedings of the 2006 European Simulation and Modelling Conference* (ESM'2006) (Toulouse, France, Oct. 23-25.) EUROSIS-ETI, 94-98.

Lencse, G. and L. Muka. 2007. “Combination and Interworking of Four Modelling Methods for Infocommunications and Business Process Systems” *Proceedings of the 2007 Industrial Simulation Conference* (ISC'2007) (Delft, The Netherlands, June 11-13.) EUROSIS-ETI, 350-354.

Muka, L. and G. Lencse. 2007. “Decision Support Method for Efficient Sequential and Parallel Simulation: Time Decomposition in Modified Conceptual Models” *Proceedings of the 2007 European Simulation and Modelling Conference* (ESM'2007) (Malta, Oct. 22-24.) EUROSIS-ETI, 574-581.

Littlechild, S. C. and M. F. Shutler. 1991. *Operations Research in Managemen*t Prentice Hall, London

Pidd, M. 1991. “Computer simulation methods” in *Operations Research in Management*, Edited by Littlechild, S., and Shutler. M., Prentice Hall, UK.

Pongor, Gy. 1992. “Statistical Synchronisation: a Different Approach to Parallel Discrete Event Simulation” *Proceedings of the 1992 European Simulation Symposium* (ESS’92), (Dresden, Germany, Nov. 5-8) SCS Europe, 125-129.

Varga, A. and Gy. Pongor. 1997. “Flexible Topology Description Language for Simulation Programs” *Proceedings of the 9th European Simulation Symposium* (ESS'97) (Passau, Germany, Oct. 19-22.) SCS Europe, 225-229.

BIOGRAPHIES

GÁBOR LENCSE received his M.Sc. in electrical engineering and computer systems at the Technical University of Budapest in 1994 and his Ph.D. in 2000. The area of his research is (parallel) discrete-event simulation methodology. He is interested in the acceleration of the simulation of info-communication systems. Since 1997, he has been working for the Széchenyi István University in Győr. He teaches computer networks and networking protocols. Now, he is an Associate Professor. He is a founding member of the Multidisciplinary Doctoral School of Engineering, Modelling and Development of Infrastructural Systems at the Széchenyi István University. He does R&D in the field of the simulation of communication systems for the Elassys Consulting Ltd. since 1998. Dr Lencse has been working part time at the Budapest University of Technology and Economics (the former Technical University of Budapest) since 2005. There he teaches computer architectures.

LÁSZLÓ MUKA graduated in electrical engineering at the Technical University of Lvov in 1976. He got his special engineering degree in digital electronics at the Technical University of Budapest in 1981, and became a university level doctor in architectures of CAD systems in 1987. Dr Muka finished an MBA at Brunel University of London in 1996. Since 1996 he has been working in the area of simulation modelling of telecommunication systems, including human subsystems. He is a regular invited lecturer in the topics of application of computer simulation for performance analysis of telecommunication systems at the Multidisciplinary Doctoral School of Engineering, Modelling and Development of Infrastructural Systems at the Széchenyi István University of Győr.