Proceedings of the 2004 Winter Simulation Conference
R .G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, eds.

AN AUTOMATIC DISTRIBUTED SIMULATION ENVIRONMENT

Sarita Mazzini Bruschi
Regina Helena Carlucci Santana
Marcos José Santana
Thais Souza Aiza

Av. Trabalhador São-Carlense, 400, Centro
Caixa Postal 668
Instituto de Ciências Matemáticas e de Computação (ICMC), Universidade de São Paulo (USP)
São Carlos, SP 13560-970, BRAZIL

ABSTRACT

Developing a sequential simulation program is not an easy task. Developing a distributed simulation program is harder than a sequential one because it is necessary to deal with mapping physical processes into logical processes, communication and synchronization problems and learn another simulation language/library. In literature, several simulation environments can be found but the great number are for sequential simulation, not using all the advantages of a distributed/parallel platform. This paper presents ASDA, an automatic distributed simulation environment that aims at providing several possibilities to users developing a distributed simulation. The automatic word can be understood in three diferent ways: the environment automatically generates a distributed simulation program code; the environment can automatically choose one distributed simulation approach; and the environment can automatically convert a sequential simulation program into a distributed simulation program using the MRIP (Multiple Replication in Parallel) approach.

1 INTRODUCTION

Simulation has been used as a tool that aims at helping users to foresee system behaviour. In this sense, two categories of simulation are constantly being developed: analytical simulation and virtual environment (Fujimoto 2000). Analytical simulations aims at performance system evaluation using metrics such as response time, number of clients that are queued, average client time in queue, etc. In this category, the user interaction is identified only at model definition moment. Conversely, virtual environment has total user interaction and aims at analysing system behaviour by means of system action/reaction.

Considering analytical simulations, the model parameters can be defined in a deterministic way (trace driven simulation) or in a random way using distributed probability functions (stochastic simulation). There is a problem in stochastic simulation: it is necessary either to run the simulation program several times or for a long time to obtain a statistically final result valid. When it is necessary to simulate a small system, this could not be a problem but, as system increases, simulation execution time becomes high.

Distributed Simulation has been developed aiming at decreasing the simulation execution time of a sequential simulation. Two approaches have been proposed in the literature: SRIP (Single Replication in Parallel) and MRIP (Multiple Replication in Parallel). These solutions basically differ in the way the system is modelled. In SRIP the model is decomposed into logical processes, each one running on different processors; on the other hand, in MRIP independent instances of the same sequential simulation program are replicated in parallel (Ewing, McNickle and Pawlikowski 1997).

Some factors can limit the usage of distributed simulation once they require advanced knowledge on parallel/distributed computing and on simulation from the user. In other words, the user needs (Bruschi 2002):

- to know the system deeply to be simulated and its model in order to exploit all the inherent parallelism;
- to analyze the best way to split the model with the aim at maximizing the load-balancing and minimizing the communication;
- depending on the model, to know which synchronisation protocol gives the best performance;
- to know the architectural features of the used platform, making it possible to evaluate the compromise between load-balancing and communication, i.e., to define when it is worth to spoil the load-balancing in favor of minimizing the communication and vice-versa.

This paper presents a general overview of the design of a novel distributed simulation environment called

ASDA. This environment removes the limit imposed by the need of knowledge on parallel/distributed computing and simulation, helping the users to develop simulation programs. The users will be able to choose between traditional sequential simulation programs and distributed simulation programs and can leave some difficult decisions to be taken by the environment.

The rest of the paper is divided into six sections. Section 2 presents an overview of two distributed simulation approaches (SRIP and MRIP) while section 3 shows an overview about automatic simulation environments. Section 4 explains the ASDA environment and section 5 the ASDA prototype. Section 6 contains our concluding remarks.

2 BACKGROUND

When user decides to use analytical distributed simulation, two approaches can be used: SRIP and MRIP.

The SRIP approach is based on the decomposition of the simulation model into logical processes, running on different processors and communicating with each other by means of message passing protocols. However, a critical SRIP problem is the warranty of synchronism for the several processes composing the simulation program. Several protocols have been developed and grouped into two wide categories: conservative and optimistic protocols (Fujimoto 2000).

The main feature of the conservative protocols is the execution of an event only when it is safe, i.e., when there is no possibility of a causality error occurrence. While this cannot be guaranteed, the process stays blocked, making it possible to have both loss of performance and deadlocks (Fujimoto 2000). The many conservative mechanisms available differ in the way the deadlock is handled. Some of them prevent the deadlock occurrence and other ones recover the system from deadlocks. The CMB protocol (Chandy and Misra 1979) prevents deadlock by means of the adoption of null messages.

The optimistic protocols do not avoid the causality errors, allowing all events to be processed. A detection and recover mechanism is adopted to recover the simulation from possible errors, leading to a consistent state. This protocol has the advantage of exploiting all the implicit parallelism where the conservative protocols could not proceed (Fujimoto 2000). The Time Warp mechanism is the most known optimistic protocol and it is based on the Virtual Time paradigm (Jefferson 1985).

Both classes of protocols have their own set of advantages and disadvantages. The choice between them is not an easy task and depends on the application features and on the computer architecture considered.

In MRIP approach the model is not decomposed. Independent instances of the same sequential simulation program are replicated in parallel and are executed based on different random seeds. Each replication sends its partial results at the end of the run to a global analyser, where the final results are evaluated. When the accuracy defined by the user is reached the simulation stops (Ewing, McNickle and Pawlikowski 1997).

In contrast to SRIP, MRIP can be easly applicable to any system, independent of the inherent system parallelism. However, there are some situations where the MRIP technique is inappropriate (Glynn and Heidelberger 1992):

- A single replication can not be executed on a unique processor;
- The output is almost deterministic.

Although this method seems very simple, some care has to be taken regarding to the number of processors (number of replications), the length of each replication and the length of the deletion period to generate a valid confidence interval (Glynn and Heidelberger 1992).

The SRIP and MRIP approaches are not exclusive, i.e., it is possible to use SRIP and MRIP in the same simulation program.

3 SIMULATION ENVIRONMENTS

The concept of simulation environment became very important mainly because of the difficulty found when the user decides to develop a distributed simulation. It is necessary to know about simulation (approaches and characteristic of each one), distributed/parallel concepts (communication, synchronization, process scheduling) and statistic concepts to analyse the simulation output.

Several simulation environments can be found in the scientific literature and in specific companies, but the great number are designed for a specific purpose and/or with a comercial aim. These environments offer to users a number of facilities that help simulation development, such as:

- a graphical editor, where the simulation can be built graphically;
- a simulation kernel, where the simulation will run;
- an output analyser, that allows to analyse the output and construct confidence intervals to analysed parametres.

Examples of these environments are: Arena (Swet and Drake 2001), VSE (Visual Simulation Environment) (Orca Computer 2004), UCLA Simulation Environment (Bagrodia 1998) and OMNet++ (Varga 2001; OMNeTpp 2004).

3.1 Arena

Arena is a general purpose environment with a comercial aim and it is based on the SIMAN V language simulation. Arena started the use of *simulation templates* and the software could easily be adapted to any industry, company or project (Swain 1995).

Nowadays, Arena is a suite with several tools for many business needs in modeling, simulation and optimization. It

can be used for strategic business decisions and operational planning improvements (Swet and Drake 2001).

The Arena software is avaliable in three versions: the Basic Version (Arena Basic Version), where the user can simulate business processes and other systems to support high-level analysis; the Standart Edition (Arena SE), where all facilities of Arena Basic Version are provided plus complete modeling flexibility; and the Arena Professional Edition (Arena PE) which add to Arena SE the capability to craft custom simulations that mirror components of the real system, including animation (Swet and Drake 2001).

3.2 VSE

The Visual Simulation Environment (VSE) is a comercial software and its development was conducted at Virginia Tech. In 1995, a technology transfer enabled the development of the VSE comercial version 1.0 at Orca Computer, Inc.

VSE is an integrated set of software tools, including (Orca Computer 2004):

- VSE Editor: allows the user to construct graphically the model using an object-oriented paradigm;
- VSE Simulator: provides animation and allows running experiments with the model;
- VSE Output Analyser: allows the user to construct confidence intervals and compute general statistic for simulation output data;
- VSE Teacher: lets the user lean how to use VSE by watching video clips.

3.3 The UCLA Simulation Environment

The UCLA Simulation Environment is an environment that adopts the process-interaction approach to discrete-event simulation. It was developed at UCLA University and attempts to provide the following features (Bagrodia *et. al* 1998):

- Implementation of both distributed and shared memory platforms and support for a diverse set of distributed simulation protocols;
- Support to visual and hierarchical model design.

The environment is composed of: a parallel simulation language, called Parsec; a GUI, called Pave; and a portable runtime system that implements the simulation algorithms.

3.4 OMNeT++

OMNeT++ (Objective Modular Network Testbed in C++), is a discret simulation environment, designed to simulate communication networks. However, due to its generic and flexible architecture, it can be used in other areas, like queuing networks or hardware architectures.

OMNeT++ provides a component architecture for models. The components (modules) are programmed in C++, and assembled into larger components and models using a high-level language (NED). OMNeT++ has extensive GUI support, and due to its modular architecture, the simulation kernel (and models) can be embedded easily into user applications (OMNeTpp 2004).

The OMNeT++ components are (OMNeTpp 2004):

- simulation kernel library;
- compiler for the NED topology description language (nedc);
- graphical network editor for NED files (GNED);
- GUI for simulation execution, links into simulation executable (Tkenv);
- command-line user interface for simulation execution (Cmdenv);
- graphical output vector plotting tool (Plove);
- utilities (random number seed generation tool, makefile creation tool, etc.);
- documentation, sample simulations, contributed material, etc.

Comparing the simulation environments described above, it can be seen that there are a lot of options to help users when needing to develop a simulation. Nevertheless, only one of these environments provides support to distributed simulation (Parsec), supporting parallel conservative algorithms (based on null messages, conditional events and a conservative protocol that combines null messages with conditional events), and a parallel optimistic algorithm. Although these environments provide all the features of a good simulation environment (visual graphical interface, simulation kernel, output analysis), no one offers the possibility of making use of all parallel/distributed advantages, and even if it is possible, it is difficult to know which protocol will give the best performance for a specific model and platform.

The ideal environment to distributed/parallel simulation must allow the user to choose which approach is best suitable for both a parallel or a distributed platform and the model and the simulation approach. Thus, aming at the ideal environment, an automatic distributed simulation environment called ASDA, was specified and it is described as follows.

4 ASDA

The ASDA (Ambiente de Simulação Distribuída Automático – in Portuguese) aims at taking the user away from the task of translating the model into a simulation

program and it goes further: the simulation program generated in this case is a distributed simulation program. Moreover, the user is taken away from the details of communication and synchronisation and the distributed simulation features (Bruschi 2002).

The ASDA purpose is to make an automatic environment available to the users, where distributed simulation can be developed in an easy and fast way. Users that deal with this environment can have different levels of knowledge, including both extremes: users with advanced simulation knowledge and users that have only superficial knowledge in both simulation and parallel computing.

Thus, ASDA also offers the necessary flexibility to help different user profiles, meeting to the following requirements (Bruschi 2002):

1. If offers an easy learning and using environment;
2. It allows the complete generation of a distributed simulation program by an inexperienced user;
3. It offers flexibility to more experienced users as they are allowed to modify the generated programs;
4. It allows the utilisation of previously developed sequential simulation programs;
5. If offers guidelines in a way users can choose between different distributed simulation approaches;
6. It makes it easy to obtain trust data;
7. It minimizes the final simulation program time, offering an efficient process scheduling.

ASDA is basically composed of 7 modules: User Interface, Code Generator, Software Interface, Evaluation, Replication, Execution and Schedule Modules as can be seen in Figure 1 (modular structure diagram).

In order to keep the flexibility eligible by the experienced users and still offering the facilities required by the inexperienced ones, ASDA has a friendly User Interface Module. From this interface, the user can choose among specifying a new simulation model, using a developed simulation program or defining the environment variables.

If the user's choice is to define a new simulation model, it is necessary to specify the model by means of the Graphic Interface sub module. When the graphic specification is ready, the program code is generated by the Code Generator Module and the replications are directed to the Replication Module. These possibilities attend 1, 2 and 3 requirements.

If the user chooses to use a previously developed simulation program, the Software Interface Module is

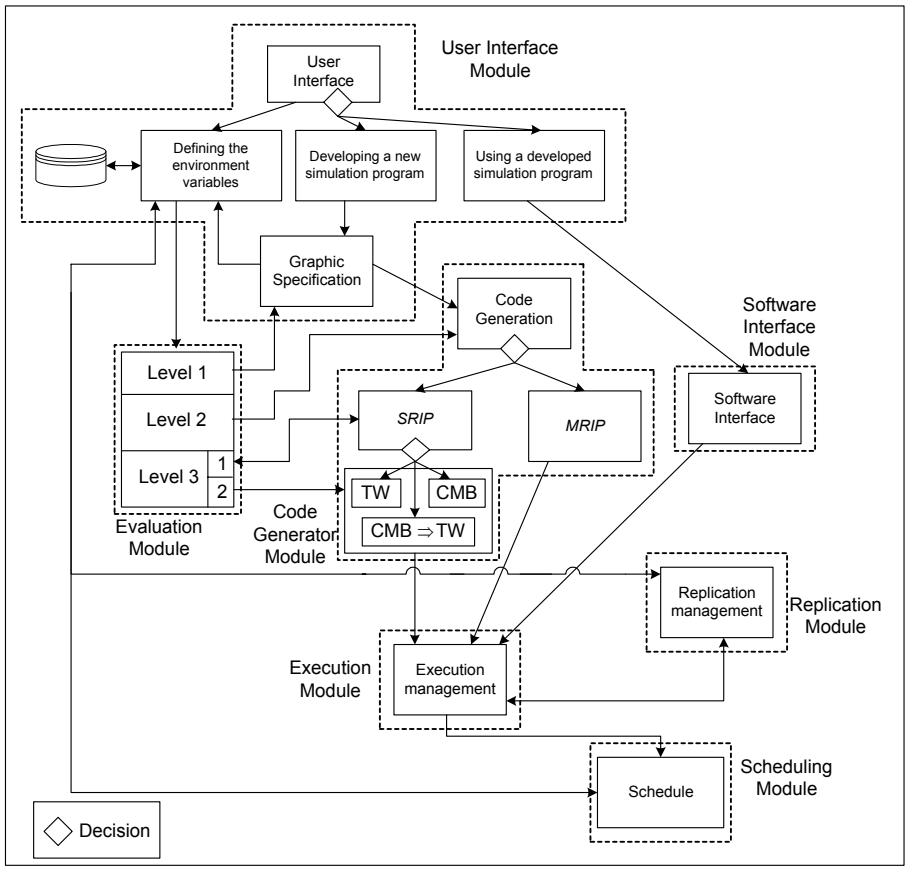

Figure 1: The ASDA Modular Interface

activated and the program is sent to the Replication Module. All simulation programs are started by the Execution Module that is responsible to communicate with the Scheduling Module in order to activate the schedule. This feature attends requirements 4 and 7.

The Evaluation Module has an important function: to help users in the whole process of constructing the simulation and it is divided in three levels: level 1, level 2 and level 3. This feature attends requirement 5.

In the next subsections, each ASDA module is detailed.

4.1 User Interface Module

The User Interface Module is responsible for allowing the user to use the facilities available to model the system. At the same time, it offers facilities to experienced users to take their own decisions. This module is responsible for offering a graphical editor to the user, where the model can be specified using a modelling approach.

Three system modelling approaches are widely used: queuing nets, petri nets and statecharts. Whereas petri nets and statecharts are able to represent parallelism, queue nets are able to represent the most important feature in computing systems: the queues. Petri nets and statecharts can not straightly represent this feature; on the other hand queuing nets can not represent system parallelism. Another possibility is Queuing Statecharts (Frances 2001), where the most relevant features from queue nets and statcharts are joined. ASDA is specified to allow two modelling approaches: extended queue nets (Soares 1992) and queuing statecharts (Frances 2001).

ASDA was specified to save all the model information by means of graphs. This characteristic will help Evaluation and Generator Modules, as will be seen. An example can be seen in Figure 2, where there is a model that was specified using queue nets and the correspondentgraph transcription (NE means entrance vertex, CS means service center vertex and NS means output vertex). The probabilities defined in model are the weights in the graphs.

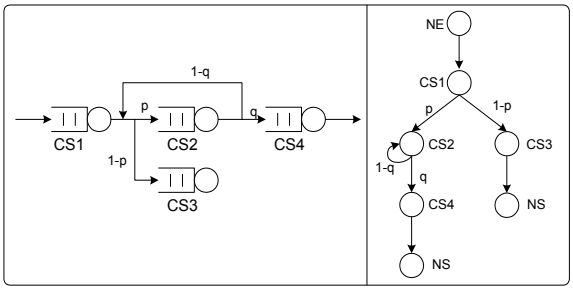

Figure 2: A Queue Net Model and its Graph Representation

4.2 Evaluation Module

This module is responsible for advising users in two ways: model specification and distributed simulation approach.

The module was divided in three levels to carry out this task:

- Level 1: verifies if the user model specification is correct;
- Level 2: helps the user to decide among SRIP, MRIP or SRIP and MRIP approaches;
- Level 3: if level 2 chooses SRIP, this level gives advices about the best SRIP protocol. The possibilities are CMB (conservative protocol), Time Warp (optimistic protocol) and CMB → Time Warp (adaptive protocol).

In order to accomplish the task, this module will use several parameters to decide an approach that will take to better performance, i.e, an approach that can result in a shorter program simulation time. For level 2, these parameters can be:

- parallel/distributed platform;
- number of processors;
- warm-up simulation length;
- model granularity;
- accuracy required by the user;

At level 3, the two most important characteristics analysed are: lookahead and possible number of rollbacks. Before this analysis, it is important to decide how the model will be decomposed into logical processes. At this point, the model representation using graphs is very useful.

A current research is implementing this module using Neural Networks (Silva 2004). Using this technique, the module will not be limited by the distributed simulations approaches defined at the implementation time. The neural network could learn about other approaches and use the knowledge to take the decisions.

4.3 Code Generator Module

This module is responsible for generating the simulation program based on the information given by the user (by means of the model). In this module the user must choose among the SRIP (stand-alone execution), MRIP (a sequential simulation program replicated) or SRIP + MRIP techniques.

After having the model fully specified, the simulation program code is automatically generated, which will depend on the simulation technique adopted. With SRIP, the code uses one of the synchronisation protocols (optimistic, conservative or adaptive), taking the model and the

architecture of the system where the simulation will be executed into consideration. With MRIP, a sequential simulation program is generated. The user can either define which is the technique appropriate to its application or follow the Evaluation Module suggestion.

In this module, a database with simulation libraries/languages allows the user to choose the target language used and, for each library/language considered, there is a pattern file with the program sequence to be generated (a template).

The MRIP code generation is simpler than the SRIP one because it is actually a sequential simulation program. In the SRIP technique the model division into logical processes and also the synchronisation protocol has to be considered.

4.4 Software Interface Module

This module is made active when the option to use a developed sequential simulation program is selected. The inclusion of this module in the environment was motivated by the large number of existing sequential simulation programs. Using the idea of replication (MRIP), it is possible to use these programs and decrease the simulation time required to obtain the results within the required accuracy.

The main idea of the interface module software is to have interfaces to functional-extensions of programming languages, such as SMPL (MacDougall 1987), ParSMPL (CMB protocol) (Ulson 1999) or SimPack (Fishwick 1992). The interfaces provide a way to modify the program in order to construct a new program, following the MRIP technique. The new program is then sent to the Replication Module.

4.5 Replication Module

The Replication Module starts the replications and controls the execution, making all the calculations to decide when the simulation must stop. Two basic components are part of this module: the MRIP master and the MRIP slave.

The main function of MRIP master is to activate the MRIP slaves, receive and process the information sent by the slaves. The MRIP slave starts the replication (the sequential simulation program), collects the data and sends them to the MRIP master.

4.6 Scheduling Module

The main purpose of the schedule is to distribute the services (processes, tasks, threads, etc.) to the process resources (processors, memory, disk, etc.) available. This task can be performed in many different ways, depending on the scheduling policies. The scheduling performance will depend on (Souza et al. 1999):

- The objective of the platform used;
- The available hardware;
- Multitask environments;
- Different application classes;
- Different workloads.

The Scheduling Module will join two research areas at the Distributed System and Concurrent Programming Group – ICMC - USP: parallel computing and performance evaluation. A novell process scheduling environment called AMIGO (DynAMical FlexIble SchedulinG Environment) has been developed (Souza et al. 1999) and will be also included in ASDA.

5 ASDA PROTOTYPE

In order to prove that ASDA is a viable environment, four modules were implemented: the User Interface, the Code Generator, the Replication and the Software Interface Modules. The User Interface module has been implemented using Java language and makes the interaction between the user and the whole environment possible. A ASDA screen can be seen in Figure 3.

The Replication module implements two basic components: the MRIP master and the MRIP slave. The master MRIP is responsible for collecting all the observations sent by the MRIP slaves, calculating the global mean and if the user's required precision was reached, the simulation stops. The slave MRIP is the simulation program added by some communication and analysis functions.

The Software Interface Module defines an interface between the developed simulation program and the Replication module. For each simulation language/library that can be used, one interface must be defined. The developed simulation program will be transformed in the MRIP slave and the function of the interface is to insert the communication and analysis functions at the correct place. At this moment, programs written in three simulation libraries can be replicated: SMPL (MacDougall 1987), ParSMPL (CMB protocol) (Ulson 1999) or SimPack (Fishwick 1992).

6 CONCLUSIONS

Many factors must be taken into consideration to develop a distributed simulation such as the model to be simulated, the available hardware and the technique used for synchronization.

ASDA was proposed aiming at helping both expert and inexperienced users with the task of developing a distributed simulation.

This new distributed simulation environment has potential to clarify the development and use of distributed simulation, offering an easy and straight automatic approach It is important to highlight four important characteristics of ASDA: it gives the possibility of using several distributed/parallel simulation approaches (SRIP, MRIP and SRIP+MRIP); the environment provides knowledege.

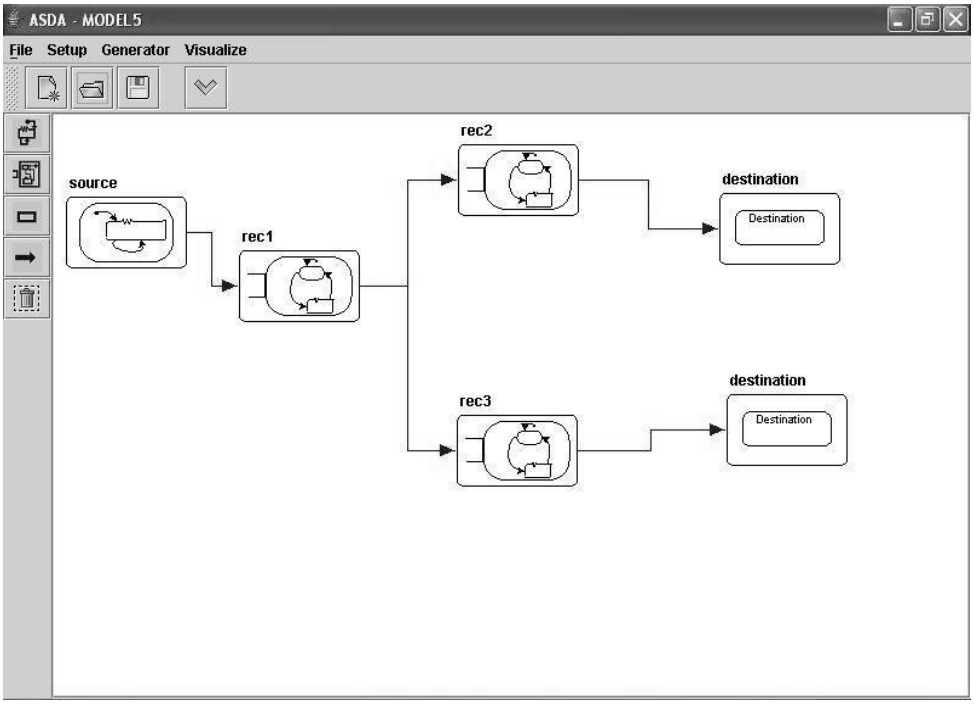

Figure 3: The Main Screen of ASDA Prototype

to help users to decide on the approach that can result in the better performace; the environment automatically generates the simulation program code, using different simulation languages/libraries; and it is possible to replicate an existing sequencial simulation program, using the MRIP approach.

ACKNOWLEDGMENTS

The authors would like to thank the financial support given to the Distributed System and Concurrent Programming Group – ICMC – USP by the Brazilian Foundation FAPESP.

REFERENCES

Bagrodia, R., R. Meyer, M. Takai, Y. Chen, Y., X. Zeng, J. Martin and H. Song. 1998. PARSEC: A Parallel Simulation Environment for Complex Systems. *IEEE Computer*, 3(10): 77-85.

Bruschi, S. M. 2002. ASDA – Um Ambiente de Simulação Distribuída Automático, Ph.D. Thesis, Department of Computer and Statistic, University of São Paulo, Brazil.

Chandy, K. M., and J. Misra. 1979. Distributed Simulation: A Case Study in Design and Verification of Distributed Programs. *IEEE Transactions on Software Engineering*, SE-5(5): 440-452.

Ewing, G. C., D. McNickle and L. Pawlikowski. 1997. Multiple Replications in Parallel: Distributed Generation of Data for Speeding up Quantitative Stochastic Simulation, In *Proc. of the 15th Congress of Int. Association for Matemathics and Computer in Simulation*, 397-402, Wissenschaft und Technik Verlag, Berlin.

Fishwick, P. 1992. SimPack: Getting Started with Simulation Programming in C and C++. In *Proceedings of the 1992 Winter Simulation Conference*, ed. R.C. Crain, J. R. Wilson, J. J. Swain, and D. Goldsman, 154–162. Arlington, Virginia.

Francês, C. R. L. 2001. Statecharts estocásticos e queuing statecharts: novas abordagens para avaliação de desempenho baseadas em especificação statecharts. Ph.D. Thesis, Department of Computer and Statistic, University of São Paulo, Brazil.

Fujimoto, R. M. 2000. *Parallel and Distributed Simulation Systems*, (New York, EUA)

Glynn, P. W. and P. Heidelberger. 1992. Analysis of initial Transient Deletion for parallel Steady-State Simulations. *SIAM J. Sci. Stat. Comput*, 13(4): 904-922.

Jefferson, D. R. 1985. Virtual Time. *ACM Transactions on Programming Languages and Systems*, 7(3): 404-425.

Macdougall, M.H. 1987. *Simulating Computer Systems Techniques and Tools*. The MIT Press.

OMNeTpp. 2004. Available online via <http://www.omnetpp.org/> [accessed July 15, 2004]

Orca Computer. 2004. Available online via <http://www.orcacomputer.com/vse/VSESet.htm> [accessed July 15, 2004]

Silva, M. P. 2004. Método para escolha de abordagem para simulação distribuída utilizando inteligência artificial.

MSc. Exam Qualification, Department of Computer and Statistic, University of São Paulo, Brazil.

Soares, L.F.G. 1992. *Modelagem e Simulação Discreta de Sistemas*, Editora Campus Ltda.

Souza, P. S. L., M. J. Santana, R. H. C. Santana, A. P. F. Araújo. 1999. PVM and a Viable and Flexible Scheduling. In *Proceeding of the 11th IASTED International Conference on Parallel and Distributed Computing and Systems* (PDCS'99), Massachusetts - USA.

Swain, J.J. 1995. Tools for process understanding and improvement: Simulation on Survey. *OR/MS Today*, 22(4): 64-79.

Swet, R. J.; and G. R. Drake. 2001. The Arena Product Family: Enterprise Modeling Solutions. In *Proceedings of the 2001 Winter Simulation Conference*, ed. M. Rohrer, D. Medeiros, B. A. Peters and J. Smith, 201–208. Arlington, Virginia.

Ulson, R. S. 1999. Simulação Distribuída em Plataformas de Portabilidade: Viabilidade de uso e comportamento do Protocolo CMB. Ph.D. Thesis, Department of Physics, University of São Paulo, Brazil.

Varga, A. 2001. The OMNeT++ Discrete Event Simulation System, In *Proceedings of the European Simulation Multiconference – ESM'2001*, Prague, Czech Republic, June 6-9.

AUTHOR BIOGRAPHIES

SARITA M. BRUSCHI is a lecturer in the Computer Science Department at the University of São Paulo (USP). She has a B.Sc. in Computer Science from UNESP (1994), and a M.Sc. and Ph.D. in Computer Science from USP, in 1997 and 2002, respectively. Since 2003 she has been working at the Distributed System and Concurrrent Programming Laboratory in the Institute of Mathematical and Computational Sciences. Her special fields of interest include distributed simulation, performance evaluation and parallel/distributed computing. She is a member of the Brazilian Computer Society (SBC). Her e-mail address is <sarita@icmc.usp.br>

REGINA H. C. SANTANA is an Associate Professor in the Computer Science Department at the University of São Paulo (USP) and co-ordinator of the B.Sc Computer Science course. She has a B.Sc. in Eletrical Engeneering from USP (1980), a M.Sc. in Computer Science from USP (1985) and a Ph.D. in Eletronics and Computer Science from University of Southampton, UK (1990). Her research interests include distributed simulation, modeling techniques and performance evaluation. She is a member of the Brazilian Computer Society (SBC). Her e-mail address is rcs@icmc.usp.br

MARCOS J. SANTANA is an Associate Professor in the Computer Science Department at the University of São Paulo (USP), Director of the Informatic Center of São Carlos (USP) and Education Director of the Brazilian Computer Society (SBC). He has a B.Sc. in Eletrical Engeneering from USP (1980), a M.Sc. in Computer Science from USP (1985) and a Ph.D. in Eletronics and Computer Science from University of Southampton, UK (1990). His research interests include modelling techniques, performance evaluation and parallel/distributed computing. He is a member of the SBC. His e-mail address is <mjs@icmc.usp.br>

THAIS S. AIZA is a postgraduate student in the Computer Science Department at the University of São Paulo (USP). She has a B.Sc. in Computer Science from Mato Grosso Federal University (1999) and a specialization course. in Information System from UNIVAG (2001). Her research interests include distributed simulation and performance evaluation. Her e-mail address is <thais@icmc.usp.br>