23

EPOCA: status and prospects

S. Donatelli[a], *N. Mazzocca*[b] *and S. Russo*[b]

[a] Dipartimento di Informatica, Università di Torino

Corso Svizzera 185, 10149 Torino - Italy

Tel.: +39 (0)81 7429246 Fax: +39 (0)81 7429 E-mail: susi@di.unito.it

[b] Dipartimento di Informatica e Sistemistica, Università di Napoli "Federico II"

Via Claudio 21, 80125 Napoli - Italy

Tel.: +39 (0)81 7682893/5 Fax: +39 (0)81 7683186 E-mail: mazzocca,russo@nadis.dis.unina.it

Abstract

In this paper we present a review of the aims, achievements and prospects of the EPOCA project. EPOCA is a Petri net based system for performance evaluation and analysis of concurrent applications. Research issues, current outcomes and future directions of the project are described.

Keywords

Parallel software engineering, Petri nets, CSP, performance evaluation

1 PROJECT AIMS AND MOTIVATIONS

EPOCA (Environment for Performance evaluation and analysis Of Concurrent Applications) is a CASE system that supports the development, the qualitative analysis and the predictive performance analysis of parallel and distributed programs. EPOCA is a joint project of the Performance Evaluation group at the Department of Computer Science of the University of Torino, and of the Parallel Architectures group at the Department of Computer Science and Systems of the University of Napoli. The project had two main objectives:

- the investigation of issues in the inclusion of Petri net based formal techniques for both program validation and performance evaluation into parallel software development;
- the definition and the construction of related appropriate CASE support tools.

To meet these goals, a methodology has been defined, centred on the use of a class of timed Petri nets, namely Generalized Stochastic Petri Nets (GSPNs). GSPNs (Ajmone-Marsan, 1984) have been chosen because they are a formalism suited to study time independent (correctness), as well as time dependent (performance) properties of concurrent programs,

I. Jelly et al. (eds.), *Software Engineering for Parallel and Distributed Systems*

and because they allow performance figures to be computed either by solving the Markov chain isomorphic to a GSPN, or, for large models, via simulation (Donatelli, 1994-a),

The methodology iterates through the following steps, until an implementation is obtained, which is correct and meets desired performance requirements:

- application implementation in a C-based CSP-like language;
- construction of a qualitative GSPN representation of the program;
- program behaviour analysis and validation via net analysis;
- introduction of program quantitative parameters in the untimed model;
- definition and computation of performance indices.

2 PROJECT OUTCOMES

The project has resulted at this stage in the following outcomes:

- a methodology for modelling and analysing concurrent programs;
- an integrated prototype CASE support system for the development, modelling and analysis phases;
- scientific publications: the EPOCA bibliography below lists the major journal and international conference papers and technical reports;
- scientific collaborations: they have been established with other groups active in the area of parallel software engineering, to cover research issues at the intersections with other projects.

In the following, the features of the EPOCA prototype system and the experience with it are summarized.

3 THE CASE SYSTEM EPOCA

In EPOCA results and proofs about a concurrent program are derived from results and proofs on a GSPN model of it. This approach is similar to the one of Shatz and Cheng for Ada (Shatz, 1987), which is limited to qualitative analysis. The EPOCA system results from the integration of different, and in same way complementary, experiences of two research groups, which resulted in the development of two tools: DISC (Iannello, 1990), a message-passing language developed at the University of Napoli, and GreatSPN (Chiola, 1991), a GSPN editing and analysis tool, developed at the University of Torino. The main functionalities of EPOCA are: translation of DISC programs into GSPN models, analysis of time-independent properties (based on structural or state space analysis), and program quantitative characterization through automatic definition and computation of net performance indices. Of course the overall EPOCA environment includes also all the functionalities of DISC (makefile generator, compiler, linker tools, run time support, monitor, post-mortem analyzer and window-based interface), and those of GreatSPN (graphical GSPN editor, net analysis tools).

EPOCA allows to build a model of the process interactions. Program flow control statements are modelled only if they contain inter-process communications. Variables are

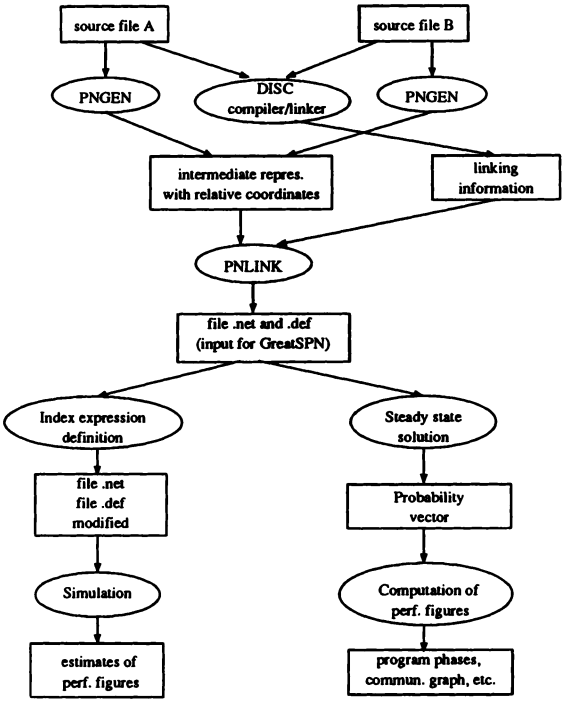

Figure 1 The program translation and analysis process in EPOCA.

not modelled automatically. Purely sequential parts of the code are collapsed into timed transitions, whose timing parameter represents the code execution time. Using this level of abstraction, the GSPN model is appropriate to analyze qualitative and quantitative properties, avoiding to deal with the detailed modeling of the sequential part of the code. Since the application does not need to be complete, for instance only the basic structure of process communication and activation may have been already coded, EPOCA allows the evaluation of certain design decisions without requiring the full implementation to be available.

Figure 1 summarizes the main features of the translation and solution process applied to a DISC program (assumed to be composed of 2 files), showing the role of the main EPOCA tools. The PNGEN module compiles each source file into an intermediate representation, that contains the translation of each process: all cross-references still have to be solved. The PNLINK module combines the intermediate files to produce the global GSPN model (defined by a ".net" and a ".def" file). The final format suitable for GreatSPN, generated by PNLINK, consists of the topological description and of the graphical layout of the net. This is useful for net visualization, animation and further editing. Proper tools provide an automatic definition of the GreatSPN expression of the desired performance indices. The model is then solved either exactly or approximately through a discrete event simulator.

4 USING EPOCA

The analysis of an application in EPOCA can start from a program skeleton (prototype), that defines the synchronization and the interactions between processes, abstracting from the algorithmic details. This gives the possibility to investigate the properties of the application using an incremental validation approach; for instance, the presence of a deadlock should be detected independently and before the evaluation of performance indices, that requires the complete estimate of the quantitative parameters.

After the automatic derivation of the GSPN model, the analyst can start performing the qualitative program analysis. This can be done using algebraic techniques, or executing the net model, or by the inspections of the reachability graph. The *algebraic techniques* provide information on the correctness and the behavior of the system. The *reachability graph* allows to investigate the dynamic evolution of the concurrent program through its state space, and it is the basis for constructing a performance model.

The performance figures computed by EPOCA can be grouped in three classes: phases, communication costs and interference costs. A *program phase* is a period of the execution in which a given set of processes is active. EPOCA computes the set of phases, the duration of each phase, the maximum/average number of processes concurrently active (degree of parallelism). *Communication costs* are used to study quantitatively the communication profile of the application: for instance, they allow to build the program communication graph. *Interference costs* express the probability of two processes requiring the CPU at the same time. All these metrics provide a quantitative characterization of the program running on the best possible platform.

The EPOCA methodology and tools have been applied to several case studies, representative of different common application classes. Table 1 reports a survey of the experiences with EPOCA, summarizing for each class the main goals of the program analysis, and the metrics computed by the EPOCA tools. These experiences show that EPOCA provide an invaluable aid for applications where formal analysis is of fundamental importance, such as safety critical ones, and that it is extremely useful for process control and distributed client-server applications, for which the dependency from input data is very well described by a stochastic model. The tools are also useful to reason on distributed algorithms, while for parallel numerical applications the Petri net-based approach to performance prediction gives results comparable to those of other methods (analytical models and monitoring techniques), but in many cases it is less effective.

Readers interested in getting a deeper insight in the different issues addressed in the project can refer to the EPOCA bibliography below. Balbo (1992) presents the net translation rules for the CSP constructs, while Donatelli (1994-b) describes the translation from DISC to GSPN. Balbo (1994) defines the program performance indices, and shows how they can be computed on the GSPN model. Donatelli (1994-a, 1994-b, 1996), Mazzeo (1996) and Jelly (1995) contain the description of some case studies (see Table 1). Donatelli (1994-a) discusses also the advantages and disadvantages of having chosen the GSPN formalism as the model formalism of EPOCA. Donatelli (1994-b) gives details on the architecture of the tool, while the role of EPOCA as a tool for computer aided distributed software engineering is discussed by Donatelli (1996).

Table 1 A survey of the experiences with EPOCA.

Application class	Description	Analysis' goals	Computed parameters
Client-server	Inter-departmental administrative computing center (Donatelli, 1996)	Deadlock freedom, program comprehension, bottleneck detection, performance tuning (with varying data transfer rates and remote data access probabilities)	*Qualitative*: deadlock states, reachable states, net animation. *Quantitative*: service time, communication graph, efficiency, degree of parallelism
Process control	Monitoring system (stochastic dependency from input data) (Balbo, 1994)	Program correctness and performance	*Qualitative*: deadlock and reachable states. *Quantitative*: mean service time, comm. graph, efficiency, degree of parallelism
Safety-critical systems	Railway transport system	Deadlock freedom, program comprehension, safety analysis, performance analysis	*Qualitative*: deadlock and reachable states, net animation. *Quantitative*: probab. of hazard states
Distributed algorithms	Circuit simulator (Balbo, 1992)	Program correctness	*Qualitative*: deadlock detection
Parallel numerical algorithms	FFT (Donatelli, 1994-a)	Comparison of data partitioning and communication strategies; program performance as a function of the data transfer rate and of the node computing power	*Qualitative*: deadlock detection. *Quantitative*: degree of parallelism, program phases, communication graph, completion time, CPU utilization

5 CONCLUSIONS AND FUTURE DIRECTIONS

The EPOCA project has addressed a problem which is perceived as increasingly important in the concurrent software engineering community, namely the definition of formal techniques and tools to support the analysis and the validation of the system implementation (i.e., the program). In EPOCA a methodology has been proposed, based on the use of the GSPN formalism. The project has shown that this methodology can be fully supported by automatic tools, and that an effective approach to do that is to integrate existing CASE tools for program development and net analysis. Although EPOCA has considered a specific CSP-like language, most of its features are independent of the details of the programming language used for system implementation. The project has pointed out that programs should have specific characteristics - such as a clear separation of structural and behavioural aspects - for static analysis tools to be capable of modelling them automatically.

The experience in EPOCA proved attractive and promising. This has suggested to extend the approach to the introduction of verification and performance evaluation techniques into the *design* refinement phase of parallel software development. Work has begun to investigate the benefits of introducing the EPOCA approach into the PARSE methodol-

ogy (Russo, 1995). PARSE (Gorton, 1995) is a design methodology is based on a graphical notation, that enables the developer to describe a parallel software system in terms of a collection of concurrent components interacting via message passing. The transformation from the PARSE design domain to the Petri net domain provides two fundamental advantages: (a) it provides formal support for design verification, (b) it allows to animate the system design specification, by executing the net. The refinement steps, until the analysis reveals that the design is error-free and that it can meet the performance requirements, will provide the software engineer with increased confidence in the final quality of the design, before proceeding to the implementation.

REFERENCES

Ajmone Marsan, M., Balbo, G. and Conte, G. (1984) A class of generalized stochastic Petri nets for the performance analysis of multiprocessor systems. *ACM Trans. Comp. Systems*, 2(1).

G. Chiola. (1991) GreatSPN 1.5 software architecture. *Proc. 5th Int. Conf. Modelling Techniques and Tools for Computer Performance Evaluation*, Torino, Italy.

Gorton, I., Gray, J. and Jelly, I.E. (1995) Object-Based Modelling of Parallel Programs. *IEEE Parallel and Distributed Technology*, **3(2)**.

Iannello, G., Mazzeo, A., Savy, C. and Ventre, G. (1990) Parallel software development in the DISC programming environment. *Future Generation Computer Systems*, **5(4)**, 365–372.

Shatz, S.M. and Cheng, W.K. (1987) A Petri net framework for automated static analysis of Ada tasking behaviour. *The Journal of Systems and Software*, **8**, 343–359.

EPOCA BIBLIOGRAPHY

Balbo, G., Donatelli, S. and Franceschinis, G. (1992) Understanding parallel program behavior through Petri net models. *Journal of Parallel and Distributed Computing*, **15(3)**, 171–187.

Balbo, G., Donatelli, S., Franceschinis, G., Mazzeo, A., Mazzocca, N. and Ribaudo, M. (1994) On the computation of performance characteristics of concurrent programs using GSPNs. *Performance Evaluation*, **19**, 195–222.

Donatelli, S., Franceschinis, G., Ribaudo, M. and Russo, S. (1994) Use of GSPNs for concurrent software validation in EPOCA. *Information and Software Technology*, **36(7)**, 443–448.

Donatelli, S., Franceschinis, G., Mazzocca, N. and Russo, S. (1994) Software architecture of the EPOCA integrated environment. *Proc. 7th Int. Conf. Modelling Techniques and Tools for Computer Performance Evaluation*, LNCS No.794, Springer-Verlag.

Mazzeo, A., Mazzocca, N., Russo, S. and Vittorini, V. (1996) A method for predictive performance evaluation of distributed programs. To appear in *Simulation: Practice and Theory*.

Russo, S., Savy, C., Jelly, I.E. and Collingwood, P. (1995) Petri net modelling of PARSE designs. Joint Tec. Rep. 7/95, Computing Research Centre, Sheffield Hallam Univ. and Univ. of Naples.