

Enhanced Debugging Methods for Parallel and Metacomputing Applications Based on Macrosteps

PhD theses

Written by Róbert Lovas

Advisor: Péter Kacsuk (MTA SZTAKI)

*Faculty of Electrical Engineering and Informatics
Budapest University of Technology and Economics*

Budapest
2005

I. Introduction

Correctness debugging of non-deterministic parallel programs is a time-consuming and tedious task, particularly in interactive way. In this case, the software engineers must face the probe effect, the irreproducibility, the completeness problem, and also the large state-space to be discovered during the debugging phase of software development cycle. Moreover, emerging high-performance applications require the ability to exploit heterogeneous and geographically distributed resources as well, which poses new challenges for application development tools.

While the importance of debugging (and testing) is highly accepted in the software engineering domain, there is still a lack of widespread and user-friendly debugging methods and tools. This work tries to overcome the limitation of existing debugging methods and combines the debugging methods with automated modelling and formal verification of parallel and metacomputing programs.

The presented work is strongly tied to two software development frameworks; P-GRADE parallel programming environment (developed by MTA SZTAKI), and HARNESS metacomputing system (developed by Emory University, Oak Ridge National Laboratory, and University of Tennessee).

P-GRADE [4] provides an integrated, graphical solution for development and execution of parallel applications on clusters, supercomputers, and Grid systems. P-GRADE significantly accelerates the reengineering procedure of sequential and legacy programs including hierarchical design with a hybrid graphical language (GRAPNEL), debugging, testing, on-line monitoring, performance analysis, and visualization phases. P-GRADE's run-time environment provides dynamic load balancing for long-running GRAPNEL applications based on fully automatic checkpointing and migration mechanisms using Parallel Virtual Machine (PVM).









On the other hand, HARNESS [K] is a metacomputing system that attempts to overcome the limited flexibility of traditional distributed computing software frameworks (such as PVM [AA]) by defining a simple but powerful architectural model based on the concept of a software backplane. The fundamental abstraction in the Harness metacomputing framework is the Distributed Virtual Machine (DVM), where not only the number of resources can be reconfigured but the services itself offered by the DVM.

II. Related works

According to the literature [A], the distributed debugging methodologies can be classified according to the level of support they provide to the software developer concerning the activities of global predicate specification and detection, and the search for the causes of the distributed program bugs.

The “*Interactive debugging of remote sequential processes*” method is based on an extension of traditional sequential debugging commands. This basic approach allows the individual online observation and control of the execution of remote sequential processes. This feature is supported by almost all existing commercial and academic distributed debuggers.

In order to address the non-reproducibility issue the “*Trace, replay and debugging*” approach is based on collecting a trace of the relevant events generated by a distributed computation, during a first run of the program. If one or more erroneous situations are found, the distributed program can be re-executed under the control of a supervisory mechanism. The trace and replay technique has been in focus of intensive research in the past decades, mostly concerning the reduction of the perturbation (probe effect) and of the volume of the traced information. However, not all debuggers include such a facility. From the user’s point of view, there is an important drawback in the “*Trace, replay and debugging*” approach, since it does not provide support for the analysis of other distributed computation paths besides the actually traced one.

	<i>Approaches</i>			
<i>Tools</i>	Interactive debugging of remote sequential processes	Trace, replay and debugging	Integrated testing, active control and debugging	Automated detection of global predicates, active control and debugging
DDT [I]	✓	 conditional breakpoints and synchronization	 conditional breakpoints and synchronization	 cross-process comparison
TotalView [J]	✓	✓ checkpoint on SGI/IRIS & IBM/AIX	 barrier and evaluation point	 evaluation point
MAD [B][C][F]	✓	✓ with NOPE	✓ with NOPE	✗ under development
DDBG & STEPS [G][H]	✓ with DDBG	✓ with STEPS	✓ with STEPS	 evaluation functions, off-line analysis
P2D2 [D]	✓	✗	 control sets	 custom grid display editor
P-GRADE [E][4]	✓ with DIWIDE debugger	✓ with macrostep engine	✓ with GRSIM (CPN) simulator	✓ with TLC temporal logic checker

Notes: ✓ Fully supported ✗ Not supported  Limited support¹

¹ The user is responsible for appropriate use of the enrolled basic features.

The “*Integrated testing, active control and debugging*” attempts to overcome the above-mentioned limitation of a simple passive trace and replay approach. Multiple authors have proposed approaches for the active control of distributed program execution for distributed debugging purposes [L][M][N].

The “*Automated detection of global predicates, active control and debugging*” approach is an attempt to help the user increasing the confidence on the results of the previous approach, by allowing the specification of the correctness criteria in terms of global predicates. Such global predicates are then automatically evaluated by detection algorithms, working off-line or online.

The following table compares some of the most sophisticated debugger tools according to their supports for the four main debugging approaches. At the end of the table the P-GRADE environment is also enrolled. In my theses, the main goal is to give user-friendly and automated solutions for all approaches by means of the tools integrated in P-GRADE; such as DIWIDE distributed debugger, macrostep engine, GRSIM simulator and TLC temporal logic checker engine. I also address the generalisation of these elaborated methods towards metacomputing applications.

III. Methods for investigation

The first aim of my work was to prove the correctness of the new macrostep-based execution (an active control mechanism) of GRAPNEL programs in P-GRADE environment applying *formal methods* from the field of model verification.

In the first thesis, the formalism of coloured Petri-nets (CPN) [R][S][T] was chosen for *modelling* GRAPNEL programs from debugging aspects. The transformation to CPN is based on the class representation of GRAPNEL programs following its hierarchical design concept. The generated CPN model is specified by the XML description of a widespread CPN simulation tool.

The formal description of the macrostep-based execution relies on the state-space (Occurrence graph) of the introduced Petri-net model. Then, the correctness of macrostep concept has been proven formally by the help of partial ordering Kripke-structures [O][P][Q], which are derived from the state-space of the CPN model in case of uncontrolled running as well as macrostep-based execution.

As the second goal, the macrostep-based debugging methodology was improved, where further *model checking techniques* [Y] have been utilized in the field of parallel debugging. The introduced support for run-time evaluation of temporal logic specifications (detection of global predicates) [Z] has been defined by state machine description. The Petri-net *simulation* tool [V][W][X] can steer and optimise the traversal of state-space during the macrostep-based execution, a *static analyser* (a partitioning algorithm) classifies the processes into subclasses, and Rayleigh *error-model* is applied for the estimation of fault density in GRAPNEL applications.

Finally, the macrostep-based execution was *generalised* towards metacomputing applications. The described approach followed the novel design methods of the HARNESS metacomputing framework, and an adaptive and open architecture has been introduced for debugging of metacomputing applications. After the *investigation*

of the available debugging tools and the *requirement analysis* for debugging of metacomputing applications, new debugging mechanisms have been developed for the unification of local and remote method calls, for transferring the consistent global states of individual processes between arbitrary debugging tools, and for the macrostep-based execution of metacomputing applications.

IV. New scientific results²

1 *Macrostep-based debugging technique for GRAPNEL applications*

In P-GRADE development environment, the parallel applications are constructed based on the syntax and semantics of GRAPNEL hybrid programming language. GRAPNEL provides language elements to express graphically the parallelism, the distribution, the concurrency, and the communication between processes at different hierarchical design levels. Meanwhile the sequential code can be inherited from legacy sequential applications.

The goal of the first thesis is to define a formal framework for the proof the correctness of macrostep-based execution of GRAPNEL programs.

1. I introduced new limitation rules and language elements in the GRAPNEL language in order to automatically create its hierarchical coloured Petri-net model for debugging purposes. Then, my transformation methods for each element of this new GRAPNEL* language are described by coloured Petri-net patterns. Based on the transformation steps, I proved that ***a coloured Petri-net model exists and can be generated for any GRAPNEL* programs for debugging purposes.***

Relevant section in the dissertation: 2.1

Related publications: [1][2][5][7][9] (GRAPNEL) [3][22][25][27] (CPN)

2. In order to handle the non-deterministic behaviour of the GRAPNEL* programs in different debugging sessions, I presented the formal description for a novel, automatic generation of successive global consistent states, called “macrostep-based” execution. The traversal of the state-space with macrosteps (generation of the Execution Tree) is also introduced, and my formal description is based on the Occurrence graph of the Petri-net model. I presented that ***the traversal rules of Occurance graph can be formally defined as the selection of representative sets of state transitions, which result the macrostep-based execution of any GRAPNEL* program.*** Thus, the software developer can apply an execution mechanism for the GRAPNEL*

² The claims related to the theses are emphasized with bold characters.

parallel programs, which is similar to step-by-step execution of traditional sequential programs.

Relevant section in the dissertation: 2.2

Related publications: [8][13][27]

3. In this thesis, I presented the correctness of the macrostep based debugging methodology. As the first step, both the Execution Tree (the result of macrostep-based execution) and the Occurrence Graph (corresponding to the entire state space of GRAPNEL* program) are transformed into Kripke structures; KS_e and KS_p . Then, I proved that *the macrostep-based selection algorithm of representative state transitions is a kind of partial ordering on the Kripke structure KS_p , and the Kripke structures KS_e and KS_p are stuttering equivalents to each other*. Therefore, the software developer gets successive global states (macrosteps) without losing any relevant information about the behaviour of observed system.

Relevant section in the dissertation: 2.3

Related publication: [27]

2 Model checking methods in parallel debugging

Temporal logic (TL) and coloured Petri-net (CPN) have proved as adequate frameworks for describing the dynamic behaviour of a system (program) consisting of multiple asynchronously executing components (processes).

The main goal of this thesis is to improve the efficiency and usability of original macrostep-based debugging methodology by the automatic comparison of the expected and the observed behaviour of GRAPNEL* programs. Thus, the novel methodology combines ideas from parallel debugging methods and from model checking algorithms as well.

1. Relying on the theoretical background of temporal logic specifications, I described a method for the integration of a new debugging framework, where the actual GRAPNEL* program runs controlled by the macrostep-based debugger as the universe in which the user-defined temporal logic formulas are checked. My method includes the initialisation phase of the framework, the way of insertion and detection of run-time temporal logic assertions, the support for the evaluation of atomic predicates referenced by temporal logic formulas, and the communication protocol with a general purpose TL checker using state machine description. Based on these achievements, I presented that *a particular class of temporal logic expressions (LTL_x) can be evaluated on the paths of Execution Tree during macrostep based execution*. In this way, the required user-interaction (to detect erroneous situations) can be radically reduced by temporal logic assertions.

Relevant section in the dissertation: 3.1

Related publications: [14][25][27][33][34]

2. In this thesis, the integration of a coloured Petri-net (CPN) simulation engine into the debugging framework is described relying on the CPN model of GRAPNEL* programs. I presented that *the CPN simulation engine is able to steer the macrostep-based traversal of state-space* (the building of the Execution Tree) *towards erroneous situations during the debugging phase, and to detect the already traversed execution paths*. The presented method can assist the software developer to find programming bugs by its simulation & steering techniques.

As a part of my work the following techniques are also proposed and investigated in order to enhance the debugger framework. (1) A reduced CPN model for GRAPNEL programs if the program does not meet the limitation rules. (2) Some methods for the partitioning GRAPNEL programs in order to split up the application to deterministic and non-deterministic parts, making the debugging cycle more efficient. (3) A distributed configuration, where more execution paths (test scenarios) can be executed simultaneously. (4) An analyser tool based on Rayleigh model for estimation of error density, where the analyser is able to recommend the release of parallel software when the program's reliability achieves the satisfied level. (5) A method that can improve the reliability of the released program in certain cases forcing its states remaining inside the already tested state-space.

Relevant sections in the dissertation: 3.2

Related publications: [25][27]

3 Debugging of metacomputing applications

Emerging high-performance applications require the ability to exploit heterogeneous and geographically distributed resources. These applications use networks to integrate supercomputers, large databases, visualization devices, and scientific instruments to form networked virtual supercomputers or *metacomputers*, which can be accessed from simple desktop computers. While the underlying physical infrastructure to build metacomputing systems is becoming more and more widespread, the heterogeneous and dynamic nature of the metacomputing environment poses new challenges for application development and debugging tools.

The major aim of this thesis is to generalise the macrostep-based debugging methodology towards metacomputing applications.

1. I presented *an adaptive debugger framework (metadebugger) for HARNESS metacomputing applications* following a software backplane approach that *is able handle the dynamic behaviour of metacomputing systems during the debugging phase*. For this purpose, the metadebugger is equipped with “export” and “import” functionalities in order to transfer the consistent process states of multi-threaded applications between debuggers hence, the invocation of third-party debuggers becomes available in the heterogeneous environment. I introduced several novel features including the automatic context

management (i.e. “step into”) mechanism for remote method invocations (RMI) that unifies the debugging mechanism for local and remote calls even if the application is executed on a metacomputing platform. Relying on this RMI support, a monitoring and visualisation subsystem are also developed for metacomputing applications as a part of my work.

Relevant sections in the dissertation: 4.1

Related publications: [11][13][28][29]

2. This thesis focuses on the non-deterministic behaviour and architecture dependencies of metacomputing applications from the debugging point of view. For this purpose, I presented the way how the ***macrostep-based debugging methodology can be generalised towards HARNESS metacomputing applications***. The new methodology is based on modified collective breakpoints, macrosteps, and resource-translation tables but some limitation rules have been introduced regarding the communication possibilities of metacomputing applications. I developed further the architecture of metadefugger framework with a macrostep controller plug-in, which relies on the monitoring facilities of the metadefugger. A resource-translation mechanism is also developed as a part of my work due to the irreproducibility of execution environment as well as an algorithm for testing the environment dependencies of metacomputing applications.

Relevant sections in the dissertation: 4.2

Related publications: [10][28]

V. Dissemination of the results

The results described in the theses are published in conference proceedings, journals, and demonstrated at several scientific forums and exhibitions.

On the other hand, P-GRADE development environment is gaining more and more attention from universities and IT companies from the EU and USA. P-GRADE has been also applied successfully in meteorology for parallelising an ultra-short range weather prediction system (Hungarian Meteorology Service) [3][12][15], in engineering for simulation of urban traffic (University of Westminster), and in chemistry for modelling of reaction-diffusion systems (Eötvös Loránd University of Budapest) [23][26][30].

Most of the presented scientific results have been already implemented in P-GRADE environment, and the software developers are able to take the advantages of these new debugging methods in order to increase the reliability of their software products. Recently, P-GRADE has been further developed to support the seamless migration from traditional parallel and distributed platforms towards grid environments [17][20] and workflow-based complex applications [4][16][18][21][24]. Therefore the designed and debugged application can be deployed on these new platforms as well providing even more new opportunities for the end-users.

The results, related to metacomputing applications, can be applied in other metacomputing or grid computing frameworks (such as [19]), which are finding acceptance as standard platforms for high performance and data intensive applications.

VI. Publications

Journal papers

- [1] P. Kacsuk, G. Dózsa, T. Fadgyas and R. Lovas: *GRADE: a Graphical Programming Environment for Multicomputers*, Journal of Computers and Artificial Intelligence, Slovak Academy of Sciences, Vol. 17, No. 5, pp. 417-427, 1998
- [2] P. Kacsuk, G. Dózsa, T. Fadgyas and R. Lovas: *The GRED Graphical Editor for the GRADE Parallel Program Development Environment*, Journal of Future Generation Computer Systems, Vol. 15, No. 3, pp. 443-452, Elsevier Science, 1999
- [3] R. Lovas, P. Kacsuk, A. Horvath, A. Horanyi: *Application of P-GRADE Development Environment in Meteorology*, Distributed and Parallel Systems. Special issue of Scalable Computing: Practice and Experience. Vol. 6, No. 2, pp. 13-22. 2005 July (ISSN 1895-1767)
- [4] P. Kacsuk, G. Dózsa, J. Kovács, R. Lovas, N. Podhorszki, Z. Balaton, G. Gombás: *P-GRADE: a Grid Programming Environment*, Journal of Grid Computing, Volume 1, Issue 2, 2004, Pages 171 - 197

Book chapter

- [5] P. Kacsuk, G. Dózsa, R. Lovas: *The GRADE Graphical Parallel Programming Environment*, In the book: *Parallel Program Development for Cluster Computing: Methodology, Tools and Integrated Environments* (Chapter 10), Editors: P. Kacsuk, J.C. Cunha and S.C. Winter, pp. 231-247, Nova Science Publishers New York, 2001

Conference papers

- [6] A. Bäcker, D. Ahr, O. Krämer-Fuhrmann, R. Lovas, H. Mierendorff, H. Schwamborn, J. G. Silva, K. Wolf: *WINPAR, Windows-Based Parallel Computing*, In: *Parallel Computing: Fundamentals, Applications and New Directions*, Series of Advances in Parallel Computing, Vol. 12, pp. 495-502, Elsevier Science, 1998
- [7] P. Kacsuk, G. Dózsa, T. Fadgyas and R. Lovas: *The GRED Graphical Editor for the GRADE Parallel Program Development Environment*, In: *High-Performance Computing and Networking*, Lecture Notes in Computer Science, Vol. 1401, pp. 728-737, Springer Verlag, 1998
- [8] P. Kacsuk, R. Lovas, J. Kovács: *Systematic Debugging of Parallel Programs in DIWIDE Based on Collective Breakpoints and Macrosteps*, In: *EuroPar '99 Parallel Processing*, Lecture Notes in Computer Science, Vol. 1685, pp. 90-97, Springer-Verlag, 1999
- [9] G. Dózsa, D. Drótos, R. Lovas: *Translation of a High-Level Graphical Code to Message-Passing Primitives in the GRADE Programming Environment*, In: *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, Lecture Notes in Computer Science, Vol. 1908, pp. 258-265, Springer-Verlag, 2000

- [10] R. Lovas, V. Sunderam: *Extension of macrostep debugging methodology towards metacomputing applications*, In: Computational Science - ICCS 2001, Lecture Notes in Computer Science, Vol. 2074, p. 263-272, Springer Verlag, 2001
- [11] R. Lovas, V. Sunderam: *A Metadebugger Prototype for the HARNESS Metacomputing Framework*, Proc. of the Tenth IEEE International Symposium on High Performance Distributed Computing, pp. 427-428, San Francisco, CA, USA, 2001
- [12] Lovas R., Horváth Á.: *Ultrarövidtávú meteorológiai előrejelző rendszer párhuzamosítása a P-GRADE fejlesztőkörnyezettel*, Proc. of NETWORKSHOP '2002, pp. 56 + CD-ROM, Eger, Hungary, 2002
- [13] R. Lovas, V. Sunderam: *Debugging of Metacomputing Applications*, Proc. of the 16th International Parallel and Distributed Processing Symposium (IPDPS-JPDC), pp. 119 + CD-ROM, Fort Lauderdale, FL, USA, 2002
- [14] J. Kovacs, G. Kusper, R. Lovas, W. Shreiner: *Integrating Temporal Assertions into a Parallel Debugger*, In: EuroPar 2002 Parallel Processing, Lecture Notes in Computer Science, vol. 2400, pp. 113-120, Springer-Verlag, 2002
- [15] R. Lovas, P. Kacsuk, A. Horvath, A. Horanyi: *Application of P-GRADE Development Environment in Meteorology*, In: Distributed and Parallel Systems, Cluster and Grid Computing, pp. 109-116, Kluwer Academic Publishers, 2002
- [16] P. Kacsuk, R. Lovas, J. Kovacs, G. Dozsa, N. Podhorszki: *Metacomputing Support by P-GRADE*, GGF8 Workshop on Grid Applications and Programming Tools, 2003
- [17] P. Kacsuk, R. Lovas, J. Kovács, F. Szalai, G. Gombás, N. Podhorszki, A. Horváth, A. Horányi, I. Szeberényi, T. Delaitre, G. Terstyánszky, A. Gourgoulis: *Demonstration of P-GRADE job-mode for the Grid*, In: EuroPar 2003 Parallel Processing, Lecture Notes in Computer Science, Vol. 2790, pp. 1281-1286, Springer-Verlag, 2003
- [18] G. Dozsa, P. Kacsuk, Sz. Illes, Cs. Nemeth, Gy. Rabai, Z. Farkas, G. Gombas, R. Lovas: *Constructing and executing Grid workflow applications by Grid portal technology*, IEEE International Conference on Cluster Computing, pp. 19-22, Hong Kong, 2003
- [19] Z. Juhasz, R. Lovas, and P. Kacsuk: *JGrid: A Jini-based Service Grid*, IEEE International Conference on Cluster Computing, pp. 28-30, Hong Kong, 2003
- [20] R. Lovas, J. Kovacs, G. Gombas, N. Podhorszki, Z. Balaton, P. Kacsuk, I. Szeberenyi, T. Delaitre, and A. Gourgoulis: *Migration and monitoring of P-GRADE parallel jobs in the Grid*, IEEE International Conference on Cluster Computing, pp. 8-11, Hong Kong, 2003
- [21] R. Lovas, G. Dózsa, P. Kacsuk, N. Podhorszki, D. Drótos: *Workflow Support for Complex Grid Applications: Integrated and Portal Solutions*, In: Grid Computing – Second European AcrossGrids Conference, AxGrids 2004, Nicosia, Cyprus, Lecture Notes in Computer Science, Vol. 3165, pp. 129-138, Springer-Verlag, 2004

- [22] B. Vécsei, R. Lovas: *Debugging method for parallel programs based on Petri-net representation*, Proceedings of MicroCAD 2004, Miskolc, Hungary, pp. 413-420, 2004
- [23] R. Lovas, P. Kacsuk, I. Lagzi, T. Turányi: *Unified development solution for cluster and grid computing and its application in chemistry*, In: Computational Science and Its Applications – ICCSA 2004: International Conference, Assisi, Italy, Lecture Notes in Computer Science, Vol. 3044, pp. 226-235, Springer-Verlag, 2004
- [24] Cs. Németh, G. Dózsa, R. Lovas and P. Kacsuk: *The P-GRADE Grid portal*, In: Computational Science and Its Applications – ICCSA 2004: International Conference, Assisi, Italy, Lecture Notes in Computer Science, Vol. 3044, pp. 10-19, Springer-Verlag, 2004
- [25] R. Lovas, B. Vécsei: *Integration of formal verification and debugging methods in P-GRADE environment*, In: Distributed and Parallel Systems: Cluster and Grid Computing, Kluwer International Series in Engineering and Computer Science, Vol. 777, pp. 83-92, 2004
- [26] I. Lagzi, R. Lovas, T. Turányi: *Development of a grid enabled chemistry application*, In: Distributed and Parallel Systems: Cluster and Grid Computing, Kluwer International Series in Engineering and Computer Science, Vol. 777, pp. 137-144, 2004

Conference/workshop presentations and posters

- [27] R. Lovas, P. Kacsuk: *Enhanced Macrostep-based Debugging Methodology for Parallel Programs*, The Third Conference of PhD Students in Computer Science, pp. 72, Szeged, Hungary, 2002 (honored with *Excellent Talk Award*)
- [28] R. Lovas: *A Debugger for Metacomputing Applications*, 2nd US/Hungarian Workshop on Cluster Computing, Metacomputing and Grid Computing, Budapest, February 6, 2002
- [29] R. Lovas: *Debugging and Visualization in the Harness Metacomputing Framework*, 1st US/Hungarian Workshop on Cluster Computing, Metacomputing and Grid Computing, Madison, Wisconsin, USA, March 15, 2001
- [30] T. Turányi, F. Izsák, R. Lovas, P. Kacsuk: *Parallelisation of an algorithm for reaction-diffusion systems applying P-GRADE environment*, Workshop 5 of the ESF Programme “REACTOR”, Prague, Czech Republic, 2004

Non-refereed workshop papers

- [31] P. Kacsuk, G. Dózsa, R. Lovas, T. Fadgyas: *Enhancing GRADE towards a professional parallel programming environment*, Proc. of the 3rd Workshop of Stimulation of European Industry Through High Performance Computing, Madrid, 1998

Technical reports

- [32] Gabor Dozsa, Robert Lovas and Peter Kacsuk: *A Target Set of Parallel Architectures for the Design of the Semi-Automatic Urine Analyser*,

<http://www.cpc.wmin.ac.uk/~ahmed/reports.html>, AHMED/3 project report, October 1997.

[33] Gabor Kuster, Wolfgang Schreiner, Robert Lovas: *Integrating Temporal Specifications as Runtime Assertions into Parallel Debugging Tools*, RISC-Linz Report Series No. 02-07, <http://www.risc.uni-linz.ac.at/library/>, March 2002

[34] Jozsef Kovacs, Gabor Kuster, Robert Lovas, Wolfgang Schreiner: *Integrating Temporal Assertions into a Parallel Debugger*, RISC-Linz Report Series No. 02-12, <http://www.risc.uni-linz.ac.at/library/>, May 2002

VII. REFERENCES

- [A] José C. Cunha, João Lourenço, Vitor Duarte: Debugging of Parallel and Distributed Programs. In the book: *Parallel Program Development for Cluster Computing: Methodology, Tools and Integrated Environments* (Chapter 5), pp. 101-136, Nova Science Publishers New York, 2001
- [B] Dieter Kranzlmüller, Axel Rinnac: Parallel Program Debugging with MAD - A Practical Approach. International Conference on Computational Science 2003, pp. 201-212
- [C] D. Kranzlmüller and J. Volkert. NOPE: A Nondeterministic Program Evaluator. In P. Zinterhof et al., editors, *Parallel Computation, Proceedings of ACPC'99*, 4th International ACPC Conference, volume 1557 of Lecture Notes in Computer Science, pages 490-499, Salzburg, Austria, February 16-18, 1999. Springer, Berlin.
- [D] Robert Hood. The p2d2 project: building a portable distributed debugger. Proceedings of the SIGMETRICS symposium on Parallel and distributed tools, May 22 - 23, 1996, Philadelphia, PA USA
- [E] J. Kovacs, P. Kacsuk. The DIWIDE Distributed Debugger on Windows NT and UNIX Platforms, *Distributed and Parallel Systems, From Instruction Parallelism to Cluster Computing*, Eds.: P. Kacsuk and G. Kotsis, Cluwer Academic Publishers, 2000.
- [F] D. Kranzlmüller, S. Grabner, J. Volkert. Event Graph Visualization for Debugging Large Applications. Proc. SPDT'96, ACM SIGMETRICS Symp. on Parallel and Distr. Tools, Philadelphia, USA, pp. 108-117, 1996
- [G] Henryk Krawczyk, Piotr Kuzora, Marcin Neyman, Jerzy Proficz and Bogdan Wiszniewski: STEPS - a Tool for Structural Testing of Parallel Software. In the book: *Parallel Program Development for Cluster Computing: Methodology, Tools and Integrated Environments* (Chapter 16), pp. 334-354, Nova Science Publishers New York, 2001
- [H] José C. Cunha, João Lourenço and Vitor Duarte: The DDBG Distributed Debugger. In the book: *Parallel Program Development for Cluster Computing: Methodology, Tools and Integrated Environments* (Chapter 13), pp. 292-303, Nova Science Publishers New York, 2001
- [I] Allinea Software Ltd.: Distributed Debugger Tool v1.8, User Guide, 2004
- [J] Etnus, LLC. TotalView debugger. Available online at <http://www.etnus.com/TotalView/MPI.html>
- [K] M. Migliardi, V. Sunderam, A. Geist, J. Dongarra. Dynamic Reconfiguration and Virtual Machine Management in the Harness Metacomputing System, Proc. of ISCOPE98, pp. 127-134, Santa Fe', New Mexico (USA), December 8-11, 1998.

- [L] A. Tarafdar and V. K. Garg: Predicate control for active debugging of distributed programs, Proceedings of the 1st Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing (IPPS/SPDP-98), pages 763-769, Los Alamitos, March 30-April 3 1998
- [M] João Lourenço, José C. Cunha: Fiddle: A Flexible Distributed Debugging Architecture, ICCS 2001, San Francisco, CA, USA, 2001, pp. 821-830
- [N] Frey and Oberhuber, M.: Testing and Debugging Parallel and Distributed Programs with Temporal Logic Specifications, Proc. of Second Workshop on Parallel and Distributed Software Engineering 1997, pages 62-72, Boston, May 1997
- [O] Alberto Lluch-Lafuente, Stefan Leue and Stefan Edelkamp: Partial Order Reduction in Directed Model Checking, In: Proceedings of the 9th International SPIN Workshop on Model Checking Software, Springer LNCS, Grenoble, April 2002
- [P] E. M. Clarke, O. Grumberg, M. Minea, D. Peled. State space reduction using partial order techniques. Software Tools for Technology Transfer, vol. 3, no. 1, Springer Verlag, 1999, pp. 279-287.
- [Q] R. Kurshan, V. Levin, M. Minea, D. Peled, H. Yenigün. Static partial order reduction. Proceedings of the 4th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Lisbon, Portugal, March/April 1998, pp. 345-357
- [R] K. Jensen: Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1, Basic Concepts. Monographs in Theoretical Computer Science, Springer-Verlag, 1992
- [S] K. Jensen: Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 2, Analysis Methods. Monographs in Theoretical Computer Science, Springer-Verlag, 1994
- [T] M. Software “Desgin/CPN. A Tool Package Supporting the Use of Colored Petri Nets” Tech. Rep., Meta Software Corporation, Cambridge, MA, USA, 1991
- [U] P. Rondogiannis, M.H.M Cheng. Petri-net-based deadlock analysis of Process Algebra programs. Science of Computer Programming, 1994. Vol. 23 (1), pp. 55-89
- [V] Matthew B. Dwyer, Lori A. Clarke, and Kari A. Nies. A compact petri net representation for concurrent programs. Technical Report TR 94-46, University of Massachusetts, Amherst, 1994
- [W] Jim Greene: Ensuring Delivery of Highly Reliable, Complex Software Releases, QSM White Paper, October 2003
- [X] I. Majzik, A. Pataricza and A. Bondavalli: Stochastic Dependability Analysis of System Architecture Based on UML Models. In R. de Lemos, C. Gacek and A. Romanovsky (eds.): Architecting Dependable Systems, LNCS-2667, Springer Verlag, Berlin, 2003, pp 219-244
- [Y] E. M. Clarke, Jr., O. Grumberg, and D. A. Peled. Model Checking. MIT Press, Cambridge, MA, 1999.
- [Z] Z. Manna and A. Pnueli. The Temporal Logic of Reactive and Concurrent Systems Specification. Springer, Berlin, 1992.
- [AA] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, B. Mancheck and V. Sunderam. PVM: Parallel Virtual Machine a User’s Guide and Tutorial for Networked Parallel Computing, MIT Press, Cambridge, MA, 1994.