

# Von der Selbst-Diagnose zum Verlässlichkeits-Benchmark

## Tübingen, Frankfurt, Erlangen

Mario Dal Cin  
Institut für Informatik  
Universität Erlangen-Nürnberg

Der Workshop ‚Self-Diagnosis and Fault Tolerance‘, der am 9. und 10. Juli 1981 in Tübingen stattfand, markiert einen ersten Höhepunkt unserer Arbeit auf dem Gebiet der Fehlertoleranz. Das Autorenverzeichnis dieses Workshops konnte zur damaligen Zeit durchaus als ‚Who is Who in Fehlertoleranz‘ im deutschsprachigen Raum (und darüber hinaus) gelesen werden. Die Workshop-Beiträge der Tübinger Forschungsgruppe hatten die Mehrprozessor-Selbstdiagnose, vor allem lernende und verteilte Diagnosealgorithmen, zum Thema [1-4].

### ATTEMPTO

In Tübingen wurde auch der Versuch gestartet, die Implementierbarkeit und damit den Nutzen solcher Diagnosealgorithmen zu demonstrieren. Dies führte zur Konzeption und Entwicklung des fehlertoleranten Multiprozessor-Systems ATTEMPTO (A TesTable Experimental Multi Processor with fault Tolerance ) [5,6]. Es erlaubte, unterschiedliche Strategien einer verteilten Selbstdiagnose zu implementieren und dann auch zu bewerten. Um dieses Ziel der integrierten Überwachung und Diagnose zu verwirklichen, wurde ATTEMPTO als Testbett für Parallelität, Fehlertoleranz und Diagnoseverfahren entwickelt. Der Rechner bestand aus Standardkomponenten. Mehrere Single-Board Computer (SBC) auf 68xxx Basis, verbunden über VME-Bus, und ein UNIX-Derivat als Betriebssystem bildeten die Hard- und Softwaregrundlage des Systems. Das Betriebssystem wurde durch eine für den Benutzer transparente Fehlertoleranzschicht abgeschirmt. Sie erlaubte eine zu UNIX binärkompatible und benutzertransparente parallele und fehlertolerante Abarbeitung von Jobs.

1985 wechselte der größte Teil der Tübinger Forschungsgruppe an die Universität Frankfurt. In Frankfurt entstanden im Rahmen der Weiterentwicklung von ATTEMPTO auch Arbeiten über Verfahren zur expliziten, robusten und fehlertoleranten Programmierung von Parallelrechnern, die sich unter anderem auf verteilte Selbst-Diagnosealgorithmen für Programm-Module abstützen [7].

### FTMPS

In Erlangen wurde dann im Rahmen des Esprit-Projekts FTMPS (Fault-Tolerant Massively Parallel Systems) ein Verfahren zur Fehlerdiagnose für Multiprozessoren mit regulärer Hardware-Topologie entwickelt und implementiert [8,9]. Bei der Erstellung des Diagnosekonzepts wurden sowohl Aspekte der Skalierbarkeit als auch der Effizienz und der Portierbarkeit berücksichtigt, die für große, massiv parallele Systeme von besonderer Bedeutung sind. Die Diagnose ist hierarchisch in drei Schichten aufgebaut. Die unterste Ebene wird von Test-Moduln gebildet, die gegenseitige Tests benachbarter Prozessoren erlauben; im einfachsten Fall werden <I am alive> - Nachrichten ausgetauscht. Die Testergebnisse werden an die zweite Ebene (Moduln für lokale Diagnose) gesendet, welche die Testergebnisse analysiert. Bei Änderung des Systemzustands, erkannt durch die Testergebnisse, wird eine verteilte Diagnose gestartet. Falls zur Diagnose Daten über das Fehlverhalten des Rechners in der Vergangenheit notwendig sind, diese jedoch auf den ersten

zwei Ebenen nicht zur Verfügung stehen, wird die Diagnose auf der obersten Ebene gestartet. Dieser Teil der Diagnose (globale Diagnose) ist auf dem Host plaziert und hat Zugang zu einer Datenbank, die die benötigten Informationen zur Verfügung stellt.

## MEMSY

In Rahmen des Sonderforschungsbereichs ‚Multiprozessor- und Netzwerk-konfigurationen‘ wurde in Erlangen der massiv parallele Rechner MEMSY (Modular Expandable Multiprocessor System) für numerische Anwendungen entwickelt und realisiert. MEMSY ist ein hierarchisch organisierter MIMD – Rechner, der sowohl einen verteiltem physikalischen als auch gemeinsame lokale Speicher besaß. Der Prototyp besaß 20 Knoten (mit jeweils vier Motorola MC88100 RISC-CPU's) verbunden über ein mehrschichtiges Verbindungsnetzwerk, in das die gemeinsamen lokalen Speicher (Koppelmodule) eingebunden waren [10-19,22]. Die Entwicklung dieses Rechners bot uns die Gelegenheit, unterschiedliche Mechanismen einer hardware-basierte Fehlertoleranz für massiv parallele Rechnern zu realisieren und zu erproben. Besonderer Wert wurde dabei darauf gelegt, dass diese Mechanismen skalierbar sind. Deshalb (und aus Kostengründen) wurde auf eine rein fehler-maskierende Redundanz, die relativ leicht zu implementieren gewesen wäre, verzichtet. Das Grundprinzip der Fehlertoleranz in MEMSY war ‚Rekonfiguration und Rücksetzen‘. Rekonfiguration erforderte ein flexibles Verbindungsnetzwerk mit redundanten Pfaden, Rücksetzen eine effiziente Fehler-Selbstdiagnose. Die Selbstdiagnose, in anderen Worten, die automatische Fehlerlokalisierung, musste so schnell erfolgen, dass ein problemloses Wiederaufsetzen der Knotenzustände möglich wurde. Deshalb wäre eine rein software-basierte Lösung unzureichend gewesen. Die Selbstdiagnose muss also durch Hardware-Mechanismen unterstützt werden. Für MEMSY wurden deshalb mehrere hardware-basierte Verfahren für selbst-diagnostizierende Rechnerknoten entwickelt, unter anderem: Verdopplung jedes Knotenprozessors zu einem Duplex-Prozessor mit Vergleichslogik, Einsatz von so genannten Watchdog-Prozessoren und Einsatz von dedizierten Diagnoseknoten, die jeweils einen Cluster von Arbeitsprozessoren überprüfen. Die Entwicklung dieser Mechanismen erfolgte in enger Zusammenarbeit mit unseren ungarischen Partnern [19,21]. Erwähnt sei auch noch, dass im Rahmen der Entwicklung das Konzept eines stabilen Multiport-Speichers für zuverlässigkeitskritische Systemdaten realisiert wurde [20].

## VERIFY

Nun sind insbesondere Hardwarefehler in modernen Rechnern (Gott sei Dank) recht selten. Deshalb muss, um die erwähnten Fehlertoleranz-Mechanismen auch auf ihre Leistungsfähigkeit hin überprüfen zu können, eine Möglichkeit bestehen, Fehler gezielt in das System einstreuen zu können. Mit anderen Worten, es werden Werkzeuge benötigt, die es erlauben, Fehlerinjektions-Experimente durchzuführen. VERIFY (VHDL-based Evaluation of Reliability by Injecting Faults efficientlY) ist eine auf VHDL basierende Plattform für Fehlerinjektions-Experimente [23-26]. Mit dem Projekt konnten zwei Modellierungs-Paradigmen erfolgreich miteinander verknüpft werden. So ist es möglich, mit Hilfe dieses Werkzeugs Modelle von Systemen aufzustellen, die sich im Normalfall deterministisch verhalten (Modellierung mit der Sprache VHDL). Zusätzlich können statistische Eigenschaften dieser Systeme (z.B. Fehler) berücksichtigt werden. Der Vorteil dieser Modellierungsart ist, dass in den Modellen alle Informationen enthalten sind, die für ihre Auswertung bezüglich des Verhaltens unter dem Einfluss von Fehlern gebraucht werden. In den in der Literatur beschriebenen Verfahren sind dagegen jeweils mehrere Modelle notwendig (z.B. Hardware- und Fehlermodell). Mit VERIFY erstellte Modelle sind daher wesentlich einfacher zu gewinnen, besser wartbar und damit auch konsistenter (Hardware- und Fehlermodell können nicht versehentlich voneinander abweichen). Dies ist gerade bei sicherheitskritischen Anwendungen ein wesentlicher Vorteil.

## PANDA

In Erlangen wurde auch das auf Petri-Netzen basierende Werkzeug PANDA (Petri Net Analysis and Design Assistant) für die Modellierung und Analyse von Fehlertoleranz-Strukturen konzipiert und entwickelt [27-29]. Da dieses Werkzeug seinen Schwerpunkt in der Analyse von Fehlertoleranzstrategien hat, wurde eine Methode integriert, die es erlaubt, verallgemeinerte Fehlerbäume zu analysieren. Außerdem besitzt es eine Optimierungskomponente, welche die Parameter eines modellierten Systems nach einer wählbaren Zielfunktion unter Berücksichtigung von Randbedingungen optimiert. Dieses Analysewerkzeug wurde und wird weiterhin in verschiedenen Projekten eingesetzt [33,34], so z.B. auch für die Verlässlichkeitsanalyse von Geschäftsprozess-Modellen [30-32]. Bei der Geschäftsprozess-Modellierung spielen Ressourcen und deren durch Aktivitäten gesteuerte Zustandsübergänge eine zentrale Rolle. Das Verhalten der Ressourcen, inklusive deren Fehlverhalten und Ausfälle, kann mittels der UML-Zustandsdiagramme detailliert modelliert werden. Die UML-Aktivitätsdiagramme ermöglichen die Beschreibung der Geschäfts-Aktivitäten und erlauben durch einige Erweiterungen auch die zielgerichtete Beeinflussung von Zustandsübergängen der Ressourcen. Die UML-Diagramme werden in Generalisierte Stochastische Petri Netze übersetzt und mit PANDA ausgewertet. PANDA wurde um eine Komponente zur Rücktransformation der Ergebnisse erweitert. Damit sollen nun reale Geschäftsprozesse in Zusammenarbeit mit Partnern der Wirtschaftswissenschaften und der Industrie untersucht werden.

## HIDE

Neben der Konzeption, Implementierung und experimentellen Erprobung von Fehlertoleranz-Mechanismen war immer auch deren Modellierung ein Ziel unserer Tätigkeiten, um derartige Mechanismen auch analytisch oder simulativ untersuchen zu können. PANDA gab uns schon die Möglichkeit, dies auf der Basis von Generalisierten Petri-Netzen zu tun. Realistische Petri-Netz-Modelle sind jedoch nur schwierig zu erstellen und zu verstehen. Sie sind für den ‚täglichen Gebrauch‘ kaum geeignet. Deshalb haben wir nach einer Möglichkeit gesucht, Fehlertoleranz-Mechanismen und fehlertolerante Systeme anschaulicher und für ‚jedermann‘ verständlich zu modellieren. Hier bot sich uns die UML (Unified Modeling Language) an. Dieser Ansatz wurde in Rahmen des EU-Forschungsprojekts HIDE [33,34] verfolgt. Ziel des Projekts war die Schaffung einer Entwicklungsumgebung für verlässliche Systeme, basierend auf UML. Sowohl die Anforderungen des Benutzers an das zu entwickelnde System als auch die Spezifikation und Analyse des Systems selber werden mit einem UML-Tool (derzeit Innovator von MID) beschrieben. Die Projektidee besteht in der Transformation von einem visuellen UML-Modell zu einem mathematischen Modell, das die automatische Analyse von Leistungs- und Zuverlässigkeitsdaten sowie die formale Verifikation des Systems ermöglicht. In der ersten Projektphase wurde gezeigt, dass diese Idee realisierbar ist und zu brauchbaren Ergebnissen führt. Die Erlanger Forschungsaufgaben umfassten einerseits die Entwicklung der Modellierungsumgebung mit Transformationen in Petri-Netze und andererseits die Modellierung von Demonstrationsbeispielen **sowie** deren Auswertung mit PANDA.

Begleitend zu diesen Arbeiten haben wir auch untersucht, ob sich formale Methoden, wie beispielsweise das Model-Checking, für die Verifikation von Fehlertoleranz-Mechanismen verwenden lassen. Dafür wurde ein Modellierungsparadigma geschaffen, das auf dem Konzept endlicher Toleranzautomaten basiert. Entsprechende Modelle eignen sich sowohl für die formale Untersuchung mittels relationaler Algebra wie auch für das Model-Checking [36-38].

## DBENCH

Fehlerinjektions-Experimente können zwar Schwachstellen eines Systems hinsichtlich seiner Verlässlichkeit aufdecken. Sie eignen sich jedoch im allgemeinen nicht dafür, die Verlässlichkeit unterschiedlicher Systeme zu vergleichen, da die injizierten Fehler meist nicht portabel sind. Folglich müssen für das Verlässlichkeits-Benchmarking zusätzliche Randbedingungen erfüllt sein. Im Dbench-Projekt der EU [39-41] soll nun ein konzeptionelles Framework und eine experimentelle Plattform geschaffen werden, die es erlaubt COTS and COTS-basierte Systeme hinsichtlich ihrer Verlässlichkeits-Eigenschaften Benchmarks zu erstellen. Verlässlichkeits-Benchmarking ist Performance-Benchmarking erweitert um Verlässlichkeits-Aspekte. Mit anderen Worten, ein Verlässlichkeits-Benchmark dient dazu, die Leistung von Systemen unter Vorhandensein von Fehlern und bei Fehlbedienung vergleichen zu können. Benchmarks müssen deshalb portabel, relevant und fair sein. Ein Benchmark besteht aus Spezifikationen so genannter Benchmark-Elemente, wie Arbeitslast, repräsentative Fehlerlast, Messgrößen, und Regeln für das Durchführen und Auswerten der Benchmark-Experimente. Deshalb wird besonderes Augenmerk darauf gelegt, Systemkonfiguration und Benchmark-Elemente so eindeutig und möglichst formal zu spezifizieren, dass ein fairer Vergleich möglich wird. Als Spezifikationssprache wird von uns VHDL eingesetzt. Im Rahmen des DBench-Projekts wurde von uns das User-Mode-Linux-System (UMLinux) [42-45] entwickelt, das die Basis für unsere Benchmark-Prototypen bilden wird.

## Referenzen

### ATTEMPTO

- [1] E. Ammann und M. Dal Cin, Algorithms for comparison-based self-diagnosis , Workshop Tübingen, 1981
- [2] R. Brause, E. Dilger und Th. Risse, Diagnosing algorithms and learning, Workshop Tübingen, 1981
- [3] H. Greilich, An algorithm for distributed self-diagnosis – a computer simulation, Workshop Tübingen, 1981
- [4] M. Dal Cin und F. Florian, Analysis of a fault-tolerant distributed diagnosis algorithm, Proc. FTCS-15, 1985
- [5] E. Ammann, R. Brause, M. Dal Cin, E. Dilger, J. Lutz, and T. Risse, ATTEMPTO: a fault-tolerant multiprocessor working station, design and concepts, Proc. FTCS Milano, 1983
- [6] M. Dal Cin, R. Brause, E. Dilger, J. Lutz and T. Risse, ATTEMPTO: An experimental Fault-tolerant multiprocessor system, Special Issue: Fault Tolerant Computing Microprocessing and Microprogramming, Vol. 20, 1987
- [7] M. Dal Cin, Zur expliziten fehlertoleranten Programmierung von Parallelrechnern, in Proc. Pars-Workshop, München, 1989

## FTMPS

[8] Deconinck, G.; Vounckx, J.; Cuyvers, R.; Lauwereins, R.; Bieker, B.; Willeke, H.; Maehle, E.; Hein, A.; Balbach, F.; Altmann, J.; Dal Cin, M.; Madeira, H.; Silva, J.G.; Wagner, R.; Viehöver, G.: Fault-Tolerance in Massively Parallel Systems, *Transputer Communications*, Vol 2(4), 1994.

[9] Vounckx, J.; Deconinck, G.; Lauwereins, R.; Viehöver, G.; Wagner, R.; Madeira, H.; Silva, J.G.; Balbach, F.; Altmann, J.; Bieker, B.; Willeke, H.: The FTMPS-Project: Design and Implementation of Fault-Tolerance Techniques for Massively Parallel Systems. In: *Proceedings of the HPCN Conference, Lecture Notes in Computer Science 797*, Springer Verlag, München, 1994.

## MEMSY

[10] M. Dal Cin, Fehlertoleranz in universellen Hochleistungsrechnern, in *Entwurf und Betrieb verteilter Systeme, Informatik-Fachberichte 264*, 1990

[11] M. Dal Cin, U. Hildebrandt, W. Hohl, L. Lehman, E. Michel, Mechanismen zur Fehlerdiagnose und -behebung für die Memsy-Hochleistungsarchitektur, Workshop Pommersfelden, 1989 Hildebrand, U.: "A Fault Tolerant Interconnection Network for Memory-Coupled Multiprocessor Systems", *Proc. 5th Int. Conf. Fault Tolerant Computing Systems, Informatik-Fachberichte 283*, Springer-Verlag, 1991

[12] Dal Cin, M.; Grygier, A.; Hessenauer, H.; Hildebrand, U.; Höning, J.; Hohl, W.; Michel, E.; Pataricza, A.: "Fault Tolerance in Distributed Shared Memory Multiprocessors", *Springer LNCS 732*, 1993

[13] Dal Cin, M.; Hohl, W.; Michel, E.; Pataricza, A.: "Error Detection Mechanisms for Massively Parallel Multiprocessors", *Proc. Euromicro Workshop on Parallel and Distributed Processing, Gran Canaria, 27. - 29. Jan. 1993*

[14] Hofmann, F.; Dal Cin, M.; Grygier, A.; Hessenauer, H.; Hildebrand, U.; Linster, C.-U.; Thiel, T.; Turowski, S.: "MEMSY - A Modular Expandable Multiprocessor System", *Springer LNCS 732*, 1993

[15] M. Dal Cin, A. Grygier, H. Hessenauer, U. Hildebrand, J. Höning, W. Hohl, E. Michel, A. Pataricza, Fault tolerance in distributed shared memory multiprocessors, *Springer LNCS 732: Parallel Computer Architectures*, 1993

[16] Pataricza, A.; Majzik, I.; Hohl, W.; Höning, J.: "Watchdog Processors in Parallel Systems", *Microprocessors and Microprogramming 39*, 1993

[17] Dal Cin, M.; Hohl, W.; Höning, J.; Pataricza, A.: MEMSY - A Modular Expandable Multiprocessor System with Fault Tolerance, *Proc. Parallel Systems Fair of the 8th Int. Parallel Processing Symposium, Cancun, April 26-29, 1994, IPDS'94, IEEE Publication*, 1994

[18] Dal Cin, M.; Hohl, W.; Dalibor, S.; Eckert, T.; Grygier, A.; Hessenauer, H.; Hildebrand, U.; Hönig, J.; Hofmann, F.; Linster, C.-U.; Michel, E.; Pataricza, A.; Sieh, V.; Thiel, T.; Turowski, S.: Architecture and Realization of the Modular Expandable Multiprocessor System MEMSY. Proc. First Intl. Conf. on Massively Parallel Computing Systems (MPCS'94), Ischia, May 1994, IEEE 1994

[19] I. Majzik, A. Pataricza, M. Dal Cin, W. Hohl, J. Hönig, V. Sieh, V.: Hierarchical Checking of Multiprocessors using Watchdog Processors, Proc. EDCC-1, Berlin, 4.-6.10. 1994, Springer LNCS 852, 1994

[20] A. Grygier, M. Dal Cin, A Stable Storage Unit for Multiprocessors, Proceedings of Pacific Rim International Symposium on Fault-Tolerant Systems (PRFTS'95) in Newport Beach, California, USA, Dezember 1995

[21] I. Majzik.; Hohl, W.; Pataricza, A.; Sieh, V.: "Multiprocessor Checking Using Watchdog Processors", Computer Systems Science & Engineering, Vol. 11, No 5, 1996

[22] M. Dal Cin; Hessenauer, H.; Hohl, W.: "The Modular Expandable Multiprocessor System MEMSY", Computer Systems Science & Engineering, Vol. 11, No 4, 1996

## VERIFY

[23] V. Sieh; Tschäche, O.; Balbach, F.: "Comparing Different Fault Models Using VERIFY", Proceedings 6th Conference on Dependable Computing for Critical Applications (DCCA-6), Grainau, März 1997

[24] V. Sieh.; Balbach, F.; Tschäche, O.: "VERIFY: Zuverlässigkeitsanalyse unter Verwendung von VHDL-Modellen mit integrierter Fehlerbeschreibung", Tagungsband 9. Workshop „Testmethoden und Zuverlässigkeit von Schaltungen und Systemen“, Universität Bremen, Bremen, März 1997

[25] V. Sieh; Tschäche, O.; Balbach, F.: "System Dependability Analysis using VHDL Models with Integrated Fault Descriptions", Extended Abstracts 8th European Workshop on Dependable Computing (EWDC-8), Göteborg, Schweden, 1.-4. April 1997.

[26] V. Sieh; Tschäche, O.; Balbach, F.: "VERIFY: Evaluation of Reliability Using VHDL-Models with Embedded Fault Descriptions", Proceedings 27th Symposium on Fault Tolerant Computing (FTCS-27), Seattle (WA), USA, Juni 1997

## PANDA

[27] Allmaier, S.; Dalibor, S.: "PANDA - Petri net ANalysis and Design Assistant" In: Tools Descriptions, 9th Int. Conference on Modelling Techniques and Tools for Computer Performance Evaluation, St.Malo, Juni 1997

[28] Dal Cin, M.; Hohl, W. ; Sieh, V.: "Hardware-Supported Fault Tolerance for Multiprocessors", Proc. Architektur von Rechensystemen ARCS '97, Rostock, VDE-Verlag, 1997

[29] S. Dalibor and M. Dal Cin, Generation test plans for distributed systems with stochastic decision models, Proc. IEEE Int. High-Assurance System Engineering Symposium , 2001

[30] D. Kreische, Performance and Dependability in Business Process Modeling . In: Proc. 5th Int. Workshop on Performability Modeling of Computer and Communication Systems (PMCCS 5). Erlangen, 2001

[31] D. Kreische, Modellierung von Geschäftsprozessen in der Unified Modeling Language und ihre Transformation in Petrinetze. Promise 2002, Potsdam, Germany, Oct. 9.-11. 2002. In Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen, Lecture Notes in Informatics (LNI), 2002

[32] Dal Cin, Linking Business Process Models with Performance and Dependability, in: Proc. IFIP WG 10.4 Meeting, 2001

## HIDE

[33] Bondavalli, A.; Dal Cin, M.; Latella, D.; Pataricza, A.: High-level Integrated Design Environment for Dependability (HIDE). In Proc. IEEE WORDS'99 Monterey, 1999.

[34] A. Bondavalli, M. Dal Cin, D. Latella, I. Majzik, A. Pataricza, and G. Savoia, Dependability analysis in the early phases of UML-based system design, Computer Systems: Science and Engineering, Vol. 16, September 2001

[35] M. Dal Cin, Verifying fault-tolerant behaviour of sate machines, Proc. IEEE Int. High-Assurance System Engineering Symposium, 1997

[36] M. Dal Cin, G. Huszerl, and K. Kosmidis, Quantitative evaluation of dependability-critical systems based on guarded statecharts, Proc. IEEE Int. High-Assurance System Engineering Symposium, 1999

[37] M. Dal Cin, Modeling fault-tolerant system behaviour, in Systems: Theory and Praxis, Springer Verlag WienNewYork, 1998 Proc. IEEE Int. High-Assurance System Engineering Symposium, 2000

[38] M. Dal Cin, Structural language for specification of quantitative requirements, Proc. IEEE Int. High-Assurance System Engineering Symposium, 2001

## DBENCH

[39] Kanoun, K. ; Costa, D. ; Dal Cin, Mario ; Gil, P. ; Laprie, J.-C. ; Madeira, H. ; Suri, N.: DBench: Dependability Benchmarking . In: IEEE Suppl. Proc. Int. Conference on Dependable Systems and Networks, Göteborg, 2001

[40] Buchacker, Kerstin ; Sieh, Volkmar: Framework for Testing the Fault-Tolerance of Systems Including OS and Network Aspects . In: IEEE Proc. High-Assurance System Engineering Symposium HASE-2001, Boca Raton

[41] K. Buchacker, M. Dal Cin, H. Höxer, O. Tschäche und V. Sieh, Reproducible dependability benchmarking experiments based on unambiguous benchmark setup descriptions, eingereicht DNS 2003, San Francisc 2003

[42] Im Rahmen des DBench-Projekts wurde ein User-Mode-Linux-System (UMLinux) entwickelt. Näheres s. <http://umlinux.de>

[43] Höxer, H.-J.; Buchacker, K.; Sieh, V.: „Implementing a User-Mode-Linux with Minimal Changes from the Original Kernel“, 9th International Linux System Technology Conference, Köln 4.-6.9.2002.

[44] Sieh, V.; Buchacker, K.: „UMLinux A Versatile SWIFI Tool“, Fourth European Dependable Computing Conference, Toulouse, Frankreich, 23.-25.10. 2002