Network of Excellence in Distributed and Dependable Computing Systems
IST Contract No. IST-2000-25088

CaberNet Vision of Research and Technology Development in Distributed and Dependable Systems

Edited by Alexander Romanovsky

January 2004

List of Contributors

Chapter 2: David Powell (LAAS), Denis Besnard (Newcastle University)

Chapter 3: Sandro Etalle (Twente University), Pieter Hartel (Twente University), Heiko Krumm (Dortmund University), Paulo Verissimo (University of Lisbon), Peter Ryan (Newcastle University)

Chapter 4: Francois Armand (Jaluna), Gerhard Fohler (Maalardalen University), Peter Puschner (TU Vienna), Manuel Rodriguez (Critical Software), Andy Wellings (York University)

Chapter 5: Roberto Beraldi (Roma University), Barbara Hughes (Trinity College), Rene Meier (Trinity College), Hugo Miranda (University of Lisbon), Luis Rodrigues (University of Lisbon), Paulo Verissimo (University of Lisbon)

Chapter 6: Sacha Krakowiak (Joseph Fourier University), Francois Armand (Jaluna)

Chapter 7: Geoffrey Coulson (Lancaster University), Maarten van Steen (Vrije University)

Chapter 8: Markus Hillenbrand (Kaiserslautern University), Valerie Issarny (INRIA), Savas Parastatidis (Newcastle University), Ferda Tartanoglu (INRIA), George Vasilakis (FORTH), Marcus Venzke (Hamburg-Harburg TU), Friedrich Vogt (Hamburg-Harburg TU)

Chapter 9: Holger Peine (Fraunhofer IESE)

Chapter 10: Alberto Montresor (Bologna University), Matthias Wiesmann (EPFL), Dietrich Fahrenholtz (Hamburg-Harburg TU), Ricardo Jimenez-Peris (TU Madrid), Marta Patino-Martinez (TU Madrid)

Chapter 11: Dirk Henrici (Kaiserslautern University), Michael Kleis (GMD FOKUS), Paul Mueller (Kaiserslautern University), Bernd Reuther (Kaiserslautern University), Detlef Bosau (Stuttgart University)

Chapter 12: Miguel Castro (MS Research), Elisa Turrini (Bologna University)

Chapter 13: Emil Lupu (Imperial College), Mike Fisher (BT), Morris Sloman (Imperial College)

Chapter 14: Santosh Shrivastava (Newcastle University)

Chapter 15: Latella Diego (ISTI CNR), Mieke Massink (ISTI CNR), Gethin Norman (Birmingham University), Dave Parker (Birmingham University)

Chapter 16: Rogerio de Lemos (Kent University), Valerie Issarny (INRIA)

Foreword

CaberNet[1] is the Network of Excellence (NoE) in distributed and dependable systems. It is funded by the European Commission's ESPRIT Programme. Its mission is to co-ordinate top-ranking European research in distributed and dependable systems, to make it accessible to governments and industries and to enhance the quality of professional training with regards to such systems. CaberNet addresses all aspects of networked computer systems design. These systems range from embedded systems used to control an aircraft in flight to globe-spanning applications searching for information on the World-Wide Web.

This document presents a CaberNet vision of Research and Technology Development (RTD) in Distributed and Dependable systems. It takes as a basis the state-of-the-art (SOTA) Report[2] prepared by John Bates in 1998: this document was commissioned by CaberNet as a first step towards the definition of a roadmap for European research in distributed and dependable systems. This report overviewed the developments in the main areas to which the CaberNet members made outstanding contributions, which were the most important at the time of its preparation, and analysed the most important trends in R&D in those areas.

The NoE is the collective author of this new document, which was put together by integrating contributions coming from many CaberNet partners. A dedicated CaberNet workshop (November 2003, Porto Santo, Portugal) and a one-day meeting of the CaberNet Links-to-Industry Forum (December 2003, London, UK) were organised to consolidate the Network understanding of the RTD Vision.

The Vision document is intended to serve as a policy-directing document. But it is equally valuable as a high level overview of recent and current activities in the selected areas, emphasising directions in which R&D in distributed and dependable systems are likely to be moving in the future.

Since CaberNet started, several very extensive roadmapping activities have been initiated by the Commission on topics of direct relevance to CaberNet's interests. We have used our involvement in some of these activities and their results (in particular, those of the AMSD and ARTIST roadmapping projects) and this, clearly, adds value to our work. Moreover, we would not have been able to prepare the vision document as it is now without using and referring to these roadmaps. But we would like to emphasise that the aim of this document is to present a vision, something which is distinctly different from a roadmapping activity: it has a broader focus and contains a more general, visionary view of the future and of the main current trends; it focuses on a number of selected topics of CaberNet excellence without attempting to give a complete coverage of the very wide total area of distributed and dependable systems; and, lastly, the document looks into a longer-term future for which we believe it would not be reasonable to define concrete milestones. On a practical note, this activity had available to it far less in the way of resources than any of the recent roadmapping activities sponsored by the Commission, this document being just one of a number of CaberNet's outputs.

To ensure the continuity of this work with respect to the SOTA document we took the structure of the SOTA report as a starting point but a number of changes have been made to reflect recent trends.

My responsibility as the editor of the Vision document was mainly focusing on combining the contributions from a large number of individual CaberNet members, resolving minor mismatches, letting people express their views and integrating their opinions and visions into the document. I am grateful to all the contributors for their willingness to work on the document, to all members of the CaberNet Links-to-Industry Forum for sharing the industry views and concerns with us, to all participants of the 5th CaberNet Plenary workshop for helping me in shaping the document structure and for their many suggestions on how to improve it, to Brian Randell, Valerie Issarny and Rogerio de Lemos for their invaluable advice on how to approach and conduct this work, to Valerie Issarny and Brian Randell for their useful suggestions on how to improve the document structure, and, finally, to all members of the CaberNet Executive Board for their support and assistance.

Alexander Romanovsky

Newcastle upon Tyne, UK
January 2004

[1] http://www.newcastle.research.ec.org/cabernet/

[2] J. Bates. The State of the Art in Distributed and Dependable Computing. A CaberNet-Sponsored Report. October 1998. http://www.newcastle.research.ec.org/cabernet/sota/index.html

Table of Contents

15 Rigorous Design

This chapter is concerned with the modelling and analysis of distributed systems using formal methods. Formal methods cover a range of techniques based on rigorous mathematical reasoning designed for the analysis of system behaviour both to find errors and prove correctness.

Distributed systems have a high complexity through concurrent, distributed, real-time activities. This combination of factors is likely to be found in an increasing number of critical hardware and software systems as a consequence of the inevitable growth in scale and functionality and the trend to interconnect systems in networks. The likelihood of subtle, difficult to detect errors in this class of systems is much greater than in systems without concurrency and real-time requirements, in particular if human interaction with the system is also considered.

With traditional quality control measures such as peer review and testing alone it is extremely difficult and time consuming to obtain sufficient statistical information about the adequacy of software for complex and critical systems in the presence of rare but possibly catastrophic errors. This is because in order to find rare errors in general an exorbitant amount of testing or simulation is required if one is to have a chance to detect such errors, which is extremely costly and time-consuming [Liggesmeyer et al 1998][Rushby 1993].

Formal methods can be used as a complementary quality control measure that can reveal inconsistencies, ambiguities and incompleteness, and several other shortcomings of system designs early on in the development process. This reduces significantly the risk of accidental introduction of design errors in the development of the software leading to higher quality software and cost reduction in the testing and maintenance phases of system development. Formal methods and their related software tools have been used extensively and successfully in the past in a variety of areas including protocol development, hardware and software verification, embedded/real-time/dependable systems and human safety, demonstrating that great improvements in system behaviour can be realised when system requirements and design have a formal basis.

In the following we highlight some of the more exciting recent advances and challenges in the development of *model checking*. Model checking is a verification technique in which efficient algorithms are used to check, in an automatic way, whether a desired property holds for a finite model of the system. Very powerful logics have been developed to express a great variety of system properties and high-level languages have been designed to specify system models.

One of the major advantages of model checking, in comparison with e.g., theorem proving, is the possibility of automatically generating counterexamples that provide designers with important debugging information [Clarke et al 2002]. Another advantage is the fact that the verification is automated so that many of the mathematical details of the verification are hidden from the designer. This reduces the amount of training that is required in order to use model-checking tools. The cost effectiveness of this technique is further improved by the fact that it is used early on in the software development cycle and therefore allows early detection of errors avoiding the need for much more expensive error correction in later phases of the development cycle.

15.1 Current and Recent Work

Model checking is becoming widely used in industry (e.g. IBM, Intel, Microsoft, Motorola, SUN Microsystems, Siemens, Bell labs) and is being applied to numerous industrial case studies and standards ranging from hardware, operating systems, compilers, and communication protocols, to control systems, embedded real-time systems and multi-media application software. Evidence of the high relevance of model checking for the dependability community is the presence of a workshop on model checking in 2003 at the International Conference on Dependable Systems and Networks, the most important conference on dependability.

The main problem of model checking is the well-known state-space explosion problem. However, many techniques have been successfully developed to alleviate this problem. Most notable is the use of appropriate abstraction techniques that allow verification of models of systems with an essentially unlimited number of states.

A further important issue is the relation between the abstract models used for specification and verification and the final implementation of the system. Formal testing is an approach that forms a bridge between these models and the implementation. It re-uses formal models for automatically generating/synthesizing relevant test-suites for component, integration and system testing and for actually executing such tests against real system implementations [Brinksma and Tretmans 2001].

With the recent increase in efficiency of model checking techniques, there is renewed interest in its direct application to software written in real programming languages such as C and Java. This could enormously enhance the capability of error detection in software code but it depends critically on the development of appropriate heuristics and automatic abstractions to guide the search for possible errors in the enormous state space generated by the software [Groce and Visser 2002].

For an ever increasing class of systems their functional correctness can no longer be separated from their "quantitative correctness", e.g. in real-time control systems, multimedia communication protocols and many embedded systems.

Hybrid Systems

Hybrid systems are characterized by a combination of discrete and continuous components: they take the form of a discrete controller embedded in an analogue environment, where the analogue part of the system may concern variations of the ambient temperature of the system, the volume of coolant in a chemical process, or, more simply, the passage of time (between system events).

Formal models for hybrid systems typically take the form of a finite-state automaton (representing the discrete component) equipped with a finite set of real-valued variables (representing the continuous component) [Alur et al 1993]. The underlying model is therefore a state transition system; this differs from control theory which is based on continuous dynamical systems, in which state variables evolve according to differential equations. The values of the variables may influence the transitions among the states of the automaton (for example, by enabling particular transitions for choice), and, conversely, the transitions between such states and the passage of time when control remains within the states can affect the values of the variables (for example, a transition may reset the values of some continuous variables to a particular value, and differential equations describe how the variables change over time when control resides in a given state). A classification of models of hybrid automata along with their theory is introduced in [Henzinger 1996][Henzinger 2000]. A subclass of hybrid automata called the timed automata [Alur and Dill 1995], which admits real-valued clocks that increase at the same rate as time as the only continuous variables, has been applied successfully to model and verify real-time systems.

The presence of real-valued variables in formalisms for hybrid systems means that the underlying semantic model of such formalisms is infinite-state. This makes their automatic analysis difficult, if not infeasible. A breakthrough result concerning the development of automatic verification methods for real-time systems was made by Alur and Dill [Alur and Dill 1995], who presented a technique for obtaining a faithful finite-state representation of timed automata. Unfortunately, the basic sub-tasks of verification such as reachability are undecidable for many classes of hybrid automata (for example, the reachability problem for timed automata equipped with one clock which can be stopped in some states and restarted in others is, in general, undecidable), although some decidability results have been developed.

Model checking tools for timed automata, such as UPPAAL and KRONOS, implement decidable algorithms, whereas semi-decidable model checking algorithms are implemented in model checkers of hybrid automata, such as the tool HYTECH. The development and implementation of efficient algorithms for timed automata has made possible the verification of several (moderately-sized) industrial case studies.

Stochastic Systems

Many systems exhibit stochastic dynamics, either in the form of discrete probabilistic behaviour that is the outcome of coin tossing, or as described by continuous probability distributions (the probability of the system moving to a given state within a specified time is given by a probability density function). Probabilistic models, typically some variants of discrete or continuous Markov processes, are those whose transition relation is probabilistic. Probabilistic modelling is used to represent and quantify uncertainty; as a symmetry breaker in distributed co-ordination problems; to model unreliable or unpredictable behaviour; and to predict system behaviour based on the calculation of performance characteristics.

The now established field of performance evaluation aims to develop formalisms and tools for modelling systems and analysing their performance measures, as a means to support the process of design and engineering. The analysis involves building a probabilistic model of the system being considered, typically a continuous time Markov chain (CTMC), but often more general probability distributions, are needed. Such models can be derived from high-level descriptions in stochastic process calculi or Petri nets. The model serves as a basis for analytical, simulation-based or numerical calculations which result in steady-state or transient probabilities and the associated performance measures (resource utilisation, average call waiting time, etc). The focus is on quantitative characteristics, including measurement and testing, and covers a broad spectrum of issues.

Probabilistic model checking is an extension of model checking techniques to probabilistic systems. As in conventional model checking, a model of the probabilistic system, usually in the form of a discrete or continuous time Markov chain (DTMC/CTMC) or a Markov decision process (MDP), is built and then subjected to algorithmic analysis in order to establish whether it satisfies a given specification. The model checking procedure combines traversal of the underlying transition graph with numerical solutions. Model checking of non-probabilistic systems has developed very quickly from first algorithms into implementations of industrially relevant software tools. In contrast, model checking of probabilistic systems has not followed the same path: although the algorithms and proof systems have been known since the mid-1980s, little implementation work has been done until recently, culminating in the development of tools PRISM [Kwiatkowska et al 2002] (for DTMCs/CTMCs and MDPs) and ETMCC [Hermanns et al 2003] (for DTMCs/CTMCs), with the help of which a number of systems have been studied (IEEE 1394 FireWire, Crowds anonymity protocol, etc). As in the case of conventional model checking, state space explosion is a particular difficulty, and has been tackled through an adaptation of symbolic (BDD based), methods and parallel, distributed or disk-based techniques. These issues have been investigated with project Automatic Verification of Randomized Distributed Algorithms.

15.2 Future Trends

Formal methods and related tools can be used most profitably to analyse parts of systems that are most critical. The development of guidelines for the identification of those system parts, the level of formality that is required and the selection of the most appropriate method to analyse its properties need to be addressed.

Particular attention must be paid to the development of models that can take user interaction with the system into account. Also here quantitative models play an important role when it comes to descriptions of human performance aspects relevant for the overall evaluation of critical systems.

The use of different methods brings about the issue of the relation between models used for different verification purposes (e.g. simulation models, models used for model checking and theorem proving but also formalisms used for data intensive versus those for control intensive problems). This becomes particularly relevant in the case of combined verification.

Training of designers, integration of tools and methods in the design process and the development of relevant academic courses to allow a broad take-up of formal methods in industry and elsewhere are fundamental for the transfer of available knowledge to the daily practice of software developers.

Hybrid Systems

We expect significant advances concerning compositionality, hierarchical modelling, refinement and object orientation. In the timed automata case, understanding of the relationship with control theory and the unification of theories is an achievable goal. Much of the future effort will be directed towards methods of analysis of timed and hybrid systems, including verification by model checking and simulation, and the scaling up of these methods to industrial size examples. The successful development of timed automaton model checking tools such as UPPAAL will be consolidated in two respects: on one hand, there will be further advances in the development and implementation of efficient algorithms for timed automata verification; on the other hand, methods for the verification of extensions of the basic timed automata model for example, with parameters, probabilities, and costs/rewards will be implemented and applied to real-life systems. Progress in this direction has been made by project Verification of Quality of Service Properties in Timed Systems.

Stochastic Systems

The challenges for this area include the modelling of spatial mobility for example based on the spi-calculus or hybrid automata. This topic is being investigated by the projects: Probabilistic Model Checking of Mobile Ad Hoc Network Protocols and A Future Of Reliable Wireless Ad hoc networks of Roaming Devices.

Compositionality, modularity and hierarchical notations will be studied and we anticipated progress to be made. Furthermore, counterexamples generated by model checkers provide designers with valuable information about possible problems in their designs and the extension of these examples to probabilistic model checking is a key issues that need to be addressed. These topics are included in the aims of project Automated Verification of Probabilistic Protocols with PRISM.

The development of convenient and powerful logics to express relevant quantitative dependability measures in a concise and formal way is key to the future systematic and possibly automatic, analysis of dependability aspects of critical systems. In addition, the development of logics to express in addition cost and reward based measures can greatly enhance the number of interesting measures relevant for dependable systems.

References

Chapter 2

[Abdellatif 2001] O. Abdellatif-Kaddour, P. Thevenod-Fosse, H. Waeselynck. Adaptation of simulated annealing to property-oriented testing for sequential problems, *2001 International Conference on Dependable Systems and Networks (DSN'2001). Fast abstracts*, Göteborg (Sweden), 1-4 June 2001, pp. B82-B83.

[Abou El Kalam 2003a] A. Abou El Kalam, R. E. Baïda, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, C. Saurel, G. Trouessin. Security Models and Policies for Health and Social Information and Communication Systems. In *1st French-speaking Conference on Management and Engineering of Hospital Systems (GISEH 2003),* (Lyon, France), pp. 268-277, 2003 (in French).

[Abou El Kalam 2003b] A. Abou El Kalam, R. El Baïda, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel, G. Trouessin. Organization Based Access Control. In *4th IEEE Workshop on Policies for Distributed Systems and Networks (POLICY-2003),* (Como, Italy), pp. 120-131, IEEE CS Press, 2003.

[Aidemark 2002] J. Aidemark, J. Vinter, P. Folkesson, J. Karlsson. Experimental Evaluation of Time-redundant Execution for a Brake-by-wire Application. *International Conference on Dependable Systems and Networks (DSN-2002)*, Washington DC, USA, June 2002.

[AMSD 2003] A dependability roadmap for the information society in Europe. Accompanying Measure System Dependability. IST Project 2001-37553. 3 parts. 2003. http://www.am-sd.org/

[Anderson et al 2003] T. Anderson, M. Feng, S. Riddle, A. Romanovsky. Protective Wrapper Development: A Case Study. In *Proc. 2nd International Conference on COTS-Based Software Systems, ICCBSS 2003*. Ottawa, Canada, February 2003. pp. 1 - 14. LNCS 2580, Springer. 2003.

[Arlat 1999] J. Arlat, Y. Crouzet, Y. Deswarte, J.C. Laprie, D. Powell, P. David, J.L. Dega, C. Rabéjac, H. Schindler, J.F. Soucaille. Fault tolerant computing. *Encyclopedia of Electrical and Electronic Engineering*, Vol.7, Ed. J.G. Webster, Wiley Interscience, ISBN 0471139467, 1999, pp. 285-313.

[Arlat 2000] J. Arlat, J.-P. Blanquart, T. Boyer, Y. Crouzet, M.-H. Durand, J.-C. Fabre, M. Founau, M. Kaâniche, K. Kanoun, P. Le Meure, C. Mazet, D. Powell, F. Scheerens, P. Thévenod-Fosse, H. Waeselynck, *Software components and dependability - integration of COTS,* 159p., Hermès Science, Paris, 2000 (in French).

[Avizienis 2001] A. Avizienis, J.-C. Laprie, B. Randell. Fundamental Concepts of Dependability. Technical Report 739, pp. 1-21, Department of Computing Science, University of Newcastle upon Tyne, 2001.

[Baier 2002a] C. Baier, H. Hermanns, B. Haverkort, J.-P. Katoen. Automated Performance and Dependability Evaluation using Model Checking. In *Performance Evaluation of Complex Systems: Techniques and Tools,* (M. Calzarossa and S. Tucci, Eds.), (Rome, Italy), Lecture Notes in Computer Science, 2459, pp.261-89, Springer, 2002.

[Baier 2002b] C. Baier, J.-P. Katoen, H. Hermanns, B. Haverkort. Simulation for Continuous-time Markov Chains. In *Concurrency Theory (CONCUR),* (L. Brim, P. Jancar, M. Kretinzki, A. Kucera, Eds.), (Brno, Czech Republic), Lecture Notes in Computer Science, 2421, pp. 338-54, 2002.

[Baxter 2003] G. D. Baxter, K. Tan, S. Newell, P. R. F. Dear, A. Monk. Analysing Requirements for Decision Support in Neonatal Intensive Care. *Archives of Disease in Childhood*, 88 (1), p. .A46, 2003.

[Beder 2001] D.M. Beder, B. Randell, A. Romanovsky, C. M. F. Rubira-Calsavara. On Applying Coordinated Atomic Actions and Dependable Software Architectures for Developing Complex Systems. *Int. Symp. on Object-oriented Real-time Distributed Computing*, Margeburg, Germany, May 2001, IEEE, 4, pp. 103-112, 2001.

[Bell 2001] A. Bell, B. R. Haverkort. Serial and Parallel Out-of-Core Solution of Linear Systems Arising from Generalised Stochastic Petri Net Models. *High Performance Computing 2001*, Seattle, USA, April 22-26, 2001

[Besnard 2003] D. Besnard. Building Dependable Systems with Fallible Humans. In *5th CaberNet Open Workshop*, Porto Santo, Portugal, November 5-7, 2003.

[Besnard and Arief 2003] D. Besnard, B. Arief. Computer security impaired by legitimate users. To appear in *Journal of Computers & Security.*

[Besnard et al 2003] D. Besnard, D. Greathead, G Baxter. When mental models go wrong. Co-occurrences in dynamic, critical systems. To appear in International Journal of Human-Computer Interaction, 2003.

Chapter 15

[Alur et al 1993] R. Alur, C. Courcoubetis, T. Henzinger, P.-H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In: Hybrid systems, LNCS-736, pp. 209-229, Springer-Verlag, 1993.

[Alur and Dill 1995] R. Alur, D. Dill. A theory of timed automata. Theoretical Computer Science, 126, 2, pp. 183-235, 1995.

[Brinksma and Tretmans 2001] E. Brinksma, J. Tretmans. (Eds). Testing Transition Systems: An Annotated Bibliography. MOVEP 2000 - MOdelling and VErification of Parallel processes, Nantes, France, LNCS, Vol. 2067, Springer-Verlag, pp. 187 - 195, 2001

[Clarke et al 2002] E. Clarke, S. Jha, Y. Lu, H. Veith. Tree-Like Counterexamples in Model Checking. IEEE symposium on Logic in Computer Science (LICS), 2002

[Groce and Visser 2002] A. Groce, W. Visser. Model Checking Java Programs using Structural Heuristics. In: Proceedings of the ACM SIGSOFT 2002 Int. Symposium on Software Testing and Analysis, Software Engineering Notes, 27, 4, 2002.

[Henzinger 1996] T. Henzinger. The theory of hybrid automata. In: Proc. LICS'96, pp. 278-292, IEEE Computer Society Press, 1996.

[Henzinger 2000] T. Henzinge. The theory of hybrid automata. In: Proc. Verification of digital and hybrid systems, volume 170 of NATO ASI Series F: Computer and Systems Sciences, pp. 265-292, Springer-Verlag, 2000.

[Hermanns et al 2003] H. Hermanns, J. P. Katoen, J. Meyer-Kayser, M. Siegle, A tool for model checking Markov chains. Software Tools for Technology Transfer, 4, 2, pp. 153 – 172, 2003.

[Kwiatkowska et al 2002] M. Kwiatkowska, G. Norman, D. Parker, PRISM: Probabilistic Symbolic Model Checker. In: Proc. TOOLS 2002, volume 2324 of LNCS, pp. 200-204, Springer-Verlag 2002.

[Liggesmeyer et al 1998] P. Liggesmeyer et al. Qualitaetssicherung Software-basierter technischer Systeme - Problembereiche und loesungansaetze. Informatik Spektrum, 21. pp. 249-258, 1998.

[Rushby 1993] J. Rushby. Formal Methods and the Certification of Critical Systems. Technical Report CSL-93-7, 1993. Available on line: http://www.csl.sri.com/papers/csl-93-7/

Chapter 16

[Allen and Garlan 1997] R. Allen, D. Garlan. A Formal Basis for Architectural Connection. ACM Transactions on Software Engineering and Methodology (TOSEM), 6(3), pp. 213-249, 1997.

[Avizienis et al 2003] A. Avizienis, J.-C. Laprie, B. Randell. *Fundamental Concepts of Dependability*. Technical Report 739. Department of Computing Science. University of Newcastle upon Tyne. 2001.

[Bernado and Inverardi 2003] M. Bernado, P. Inverardi. *Formal Methods for Software Architectures*. Lecture Notes in Computer Science 2804. Springer. Berlin, Germany. 2003.

[Björkander and Kobryn 2003] M. Björkander, C. Kobryn. Architecting Systems with UML 2.0. *IEEE Software*. July/August 2003. pp. 57-61.

[Blair et al 2000] G.S. Blair, L. Blair, V. Issarny, P. Tuma, A. Zarras. The Role of Software Architecture in Constraining Adaptation in Component-Based Middleware Platforms. Proc. Middleware'2000: IFIP/ACM International Conference on Distributed Systems Platforms, LNCS 1795, pp. 164-184, 2000.

[Dabrowski and Mills 2002] C. Dabrowski, K. Mills. Understanding Self-Healing in Service-Discovery Systems. *Proc. of the 1st ACM SIGSOFT Workshop on Self-Healing Systems (WOSS'02)*. Charleston, SC, USA. November 2002. pp. 15-20.

[Dashofy et al 1999] E.M. Dashofy, N. Medvidovic and R.N. Taylor. Using Off-The-Shelf Middleware to Implement Connectors in Distributed Software Architectures. Proc. 21st International Conference on Software Engineering (ICSE), pp. 3-12, 1999.

[Dashofy et al 2002] E. Dashofy, A. van der Hoek, R. N. Taylor. Towards Architecture-Based Self-Healing Systems. *Proc. of the 1st ACM SIGSOFT Workshop on Self-Healing Systems (WOSS'02)*. Charleston, SC, USA. November 2002. pp. 21-26.