RWTH Aachen

Department of Computer Science
Technical Report

Achieving Highly Reliable Embedded Software: An empirical evaluation of different approaches

Falk Salewski and Stefan Kowalewski

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2007-8

RWTH Aachen · Department of Computer Science · June 2007

Achieving Highly Reliable Embedded Software: An empirical evaluation of different approaches

Falk Salewski and Stefan Kowalewski

Lehrstuhl für Informatik 11
RWTH Aachen, Germany
Email: {salewski, kowalewski}@informatik.rwth-aachen.de

Abstract. Designing highly reliable embedded software is a challenge and several approaches are known to improve the reliability of this software. However, all approaches have their advantages and disadvantages which makes empirical evaluations investigating their potentials necessary. In this paper, different approaches of software reliability improvement for embedded systems were compared on basis of experiments conducted at our institute. The first approach is an instance of N-version programming based on forced diversity. Two fundamentally diverse hardware platforms (microcontroller and CPLD/FPGA) were used to force diversity. Another experiment was conducted in which participants designed their software on one hardware platform only. The second half of this experiment was used for review and testing. Based on our experiments, the potentials of our application of N-version programming, review and testing are compared with respect to different fault categories (specification, implementation, application) identified during evaluation.

1 Introduction

Designing highly reliable embedded systems is a challenge. The greatest challenge is seen by many developers in the software part of embedded systems [Sto96]. In the domain of embedded systems, more and more systems require certain levels of reliability as for example future drive-by-wire systems in the automotive industry. These embedded systems are subject to strong development constraints. Aspects as low cost, variant management, and (hard) real-time requirements have to be considered. In this context, applications of specific software reliability improvement[1] measures are needed. In this report[2], we will have a closer look at the application of three important approaches, namely N-version programming (NVP)[3], testing and review.

The approach of NVP, firstly introduced by [CA77] seems very promising, but in the well known experiment of Knight and Leveson [KL86] it has been shown that developers tend to make the same faults. Different approaches modeling this dependency structure (e.g. [LM89]) and corresponding empirical studies [BBvdM04,CL04] are known which allow certain (model-based) predictions of failure probabilities in NVP systems. Other approaches try to decrease the dependencies between the different software versions. One of these approaches is "forced diversity" introduced by [LM89] with an empirical evaluation in [LH93]. The basic assumption is that different development methodologies lead to diversity in decision and thus diversity in the behavior of the resulting product.

[1] in this context reliability is usually achieved by functional correctness
[2] A short version of this report has been published in [SK07a]
[3] As a matter of course, testing and review are needed in NVP also, but we will analyze the potentials of these approaches separately and discuss benefits of combinations afterwards.

However, recent publications [LPS99,SWK06] show, that even these improved approaches can lead to undesired dependencies between the diverse software versions. The approach of diverse NVP used for the evaluation in this report is based on different hardware platforms used in today's embedded systems. Beside classical platforms as microcontrollers (MCU), nowadays programmable logic devices (PLD) as CPLDs and FPGAs offer interesting alternatives for several applications which were not available before at reasonable conditions. Designing systems on basis of PLDs differs a lot from designing the same task on MCUs which offers possibilities to force a certain amount of diversity. Therefore, the effect of this additional diversity on NVP has been analyzed with the help of experiments conducted at our institute which are described in Section 2.

Testing, as the second approach investigated, can be applied at different stages in the design cycle and is currently one of the most important means of verification in embedded systems [PvSK90]. The disadvantage of all testing approaches is that in typical applications not every possible input combination can be checked. Embedded systems are often real-time systems which complicates the testing process additionally. Not only the values and the timing of the actual inputs but also previous inputs (values and timing) can affect the outputs increasing the amount of test cases needed. Additionally, at least the system test has to be done at hardware interfaces which makes particular test interfaces necessary. Special approaches as for example *design for testability* [VWVW96] try to overcome these problems, however, real-time properties are still a challenge for many testing approaches. The evaluation in this report applies black box testing with an automatic test environment designed for this purpose and focuses on the general potentials of testing (see Section 2).

Different approaches of reviews are well known [Fag76,PvSK90] and applied in industry. The idea is to inspect code, written by another person, to reveal problems in the code (insufficient documentation, bad code structure, potential risks as division by zero, undesired overflows, etc.) and to identify inconsistencies with the specification. Reviews offer chances to identify problems with respect to functional requirements, but also with respect to non-functional requirements as maintainability and reliability. However, the result of each review process depends a lot on the persons used as reviewers and their review performance is hard to quantify. If real-time properties have to be considered, as often necessary in embedded systems, additional challenges have to be met in the review process. Especially the time, which is needed for the execution of sequential code, is hard to determine not to speak of program parts which can be disrupted by interrupts at any time. In this report we investigate the potentials of code inspection. Details of the review-technique used can be found in Section 2.

The option of using as many of these approaches as possible seems promising but is resulting in high costs (development time and human resources). Since in many embedded applications reasonable costs of the resulting products have to be achieved (as in the mentioned automotive domain), evaluations which investigate the potentials of different approaches and their combinations to prevent failures are needed. This investigation is done for example in [LPS00,PSL00] in which an NVP approach is compared with one "good version" based on a reliability growth model. According to these results, both approaches could lead to better results. To achieve further results with respect to these problems of software reliability, we conducted controlled experiments based on an automotive real-time application investigating applications of diverse hardware NVP, review and testing and evaluated the potentials for reliability improvement of these different approaches.

The remaining report is structured as follows: In Section 2, the designs of the experiments used for the comparing analyses are presented while the following Section 3 presents the results of these experiments. Based on these results, we conducted a categorization of the faults handled with the different approaches in Section 4 and discussed potentials of the different approaches. Finally, we discussed the validity of the results in Section 5 and conclude with Section 6.

2 Design of Experiments

In this section, the experiments conducted at our institute are presented briefly. They were used to obtain the empirical data needed for the comparing evaluation of the different approaches of reliability improvement accomplished in later sections of this report.

2.1 NVP-Experiments

A first experiment with respect to diverse hardware NVP has been conducted at our institute and has been described in [SWK06]. In order to validate the results and to investigate additional aspects, the experiment was replicated in a modified form. This second experiment took place in winter term 2005/2006 with 24 computer science students (5th semester or higher) forming 12 groups. In this experiment, MCUs and FPGAs were used as diverse hardware platforms and each group had to program both hardware platforms starting with a platform picked randomly (6 groups started with MCUs, 6 groups started with FPGAs). While the task of the first experiment was mainly the frequency measurement of four independent speed signals and a communication via CAN bus (see [SWK05,SWK06] for details), the task of the second experiment had been extended with 6 additional tasks in order to increase the complexity of the application. Those tasks contained the generation and integration of additional information in the CAN message (mark old values, identify certain scenarios) and interaction with the user via three buttons (BTNA, BTNB, BTNC) resulting in an output of certain process values on LEDs in real-time (BTNA) or additional CAN messages (BTNB: a test message including a test counter which had to be incremented with each test message sent, BTNC: a message containing peak values of the input signals). For more information, see also *device under test* in Fig. 1.

2.2 Test&Review Experiment

The aim of this experiment was to identify which types of faults could be identified by review and by testing respectively. The experiment took place in winter term 2006/2007 during a lab course with 19 computer science students (5th semester or higher) forming 12 groups. Each group had to program the same task as used in the second NVP-experiment described above. However, only the microcontroller hardware was used for implementation which took part in the first half of the lab course. The second half of the lab course was used for review and testing which was organized as follows: Randomly, each group was assigned a version for review and another version for testing. The versions were anonymised to avoid interaction between the implementation and the verification group and it was assured that no group checked their own version. Half of the groups started with review while the other half started with testing in order to mask out effects of execution order and to reduce the number of test equipment needed for the experiment. After three weeks (three appointments with 3h each) the groups changed from testing to review and vice versa. In the last two weeks, all groups had the chance to improve their own version on basis of the review and test reports. In the following, the test and review process used in the experiment will be presented briefly.

For testing, all six test groups were equipped with an own automated test environment similar to the one used for evaluation (see Fig. 1 and Section 2.3). In this case, the functions of the microcontroller (MCU) and the FPGA were combined on a single FPGA and the test signal generation excluded the signals for the three buttons (had to be activated manually) to allow a simpler hardware platform for the test environment. Each group received a documentation for the test environment and an empty test report form. Additionally, all test groups received the machine code for testing and an empty test report form. The following aspects had to be filled out in every test report:

- requirements tested + results
- requirements not tested + reasons
- scenarios identified which could lead to problems + results
- quality of version tested (range 1..5, subjective opinion of the reviewers)
- final remarks concerning implementation and specification

This given form of the report helped the students during test case creation and eased the later analysis of all test reports for the evaluation.

As in the case of testing, every review group received an empty review report form, a short review instruction and the source code to review. Additionally they received the corresponding documentation which should help them to understand the code if necessary. The review report form covered the same aspects as in case of the test report and an additional grading of the reviewability.

2.3 Experiment Evaluation

Two different kinds of tests were applied during all experiments: acceptance tests and evaluation tests.

In order to receive versions with a certain minimum level of quality, each version had to pass an automated acceptance test. While this test consisted of 20 frequencies generated randomly in the first experiment (equal values for all inputs), a semi automated acceptance test was used for the second and third experiment (8 different representative input combinations generated by an FPGA programmed for this purpose). If a group failed this test, the group members had the chance to improve their implementation and try another acceptance test. However, an overall number of 5 groups did not pass the acceptance tests. For this reason only 55 of the 60 overall versions were considered for the evaluations presented in this report (see Fig. 2 for details).

The following comprehensive evaluation test was used to determine the failures in all accepted versions. The failures identified during this evaluation test were used for the later reliability[4] evaluation. The versions were tested by an automatic real-time test environment designed for this purpose (see Fig. 1). The basic function of the test environment was to generate the physical signals (reset, 4 individual frequencies and 3 button signals) needed as an input for the device under test (DUT) and to record the CAN output of the DUT as shown in Fig. 1. Different test cases based on the black-box approach (random values, extreme values, values representing certain scenarios and environment conditions) have been used during evaluation with an overall number of more than 54000 lines of test data[5]. The black box approach was used, since all versions had to be treated in the same way to avoid external influences with respect to the experiment results. Further details of the evaluation and challenges resulting from real-time requirements and specific properties of embedded systems can be found in [SK07c].

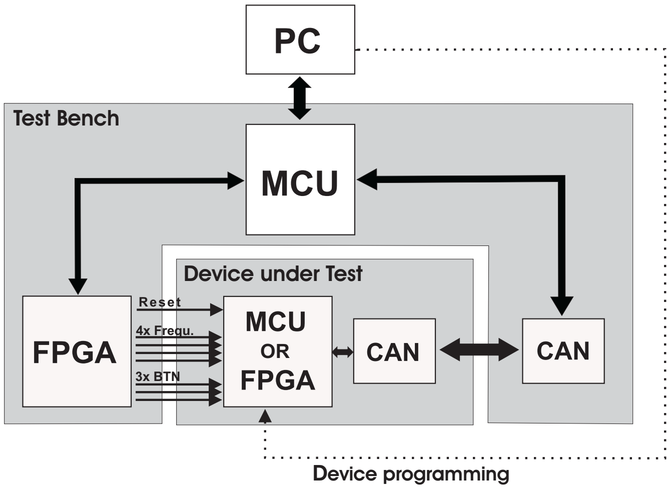

Fig. 1. Test environment used for evaluation

3 Experiment Results

As already mentioned before, all versions which passed the acceptance test were evaluated by using the automatic test environment described in section 2.3. The failures found during this evaluation are presented in the following.

3.1 First experiment (NVP)

The results of our first NVP experiment showed high numbers of dependent failures [SWK06]. For the analysis in this report we took a closer look at the

[4] in this application high reliability is resulting from functional correctness

[5] one line of test data = one set of input values + time span these input values are present at the inputs

most recent common mode failures[6] and listed them in Fig. 2 (Exp.1). The first failure mentioned in this table was present in the first messages after reset in all MCU and one CPLD version. The behavior after reset was not explicitly specified (resulting in the same requirements as during runtime) which could have led to this failure. Another reason could be seen in improper initialization methods, especially in case of the MCUs. According to the parallel structure of the CPLD and the fact that less initializations are needed in case of this type of hardware (no interrupt handlers, timers, etc have to be initialized) this specific fault occurred only in one CPLD version. Therefore, forced diversity by different hardware platforms was successful in this case.

The following failures (No. 2-4) result from not considering certain input situations like very low, very high or changing inputs. These failures occurred in many MCU and CPLD versions, however, differences can be seen between both hardware platforms. In CPLD versions the use of different measurement intervals was usually avoided (most probably because it was complicated to implement on this hardware platform). For this reason, changes of the input frequencies were handled faster with CPLDs (No. 4) coming at the cost that the accuracy in case of low input frequencies is insufficient (No. 2).

Although these failures occurred in MCUs and CPLDs with different intensity, it cannot be concluded that they guarantee diversity. Depending on the combination of versions, any failure mentioned could occur on both hardware platforms.

3.2 Second experiment (NVP, extended task)

A more complex task was used for the second NVP experiment as presented in Section 2. To avoid the comparatively simple errors (overflows, etc.) found in the first experiment, we conducted a stronger acceptance test in this second experiment. The results can be seen clearly in Fig. 2 (Exp.2, No. 2 and 3): Overflows were detected by the acceptance test as well as most of the problems with low frequency input. As before, a high number of dependent failures occurred as presented in Fig. 2 (Exp.2, No. 4, 5, 6). It seems that the additional functionality added increased the problems of response times and accuracy, especially in FPGA versions (compared to CPLD, No. 4). Further on, a common mode failure could be identified in the additional tasks (No. 10). Several versions did not implement the test message counter as specified starting with an incorrect value. The second failure within the test message (No. 11) was hardware dependent and occurred in FPGA versions only.

Obviously, some fault sources identified are *implementation depended* (No. 1, 6-12) while others are *implementation independent* (No. 2, 3-5). According to our results, the NVP approach applied might be more suitable to mitigate implementation specific faults. These issues will be looked at closer in Section 4.

3.3 Third experiment (Test & Review, extended task)

The failures, present in the majority of the versions created in the NVP experiments, seem to be found easily as soon as they have been identified once. Furthermore, we had added ambiguous statements to the specification which had

[6] failure as a result of similar faults in different redundant modules [Sto96]

No.	Failure describtion	Type of problem	# versions with this failure				
			Exp. 1*		Exp. 2		Exp.3
			MCU	CPLD	MCU	FPGA	MCU
1	Wrong or delayed CAN messages after reset, especially if the input signal frequency is high.	Impl./ Spec.	100%	10%	36%	20%	25%
2	Wrong values in the CAN message as soon as input signals are of very low frequency (<5Hz). → measurement interval probably too short	Appl.	33%	80%	18%	40%	25%
3	Wrong values in the CAN message as soon as input signals are close above the maximum frequency explicilty specified. → Overflows or wrong determination of maximum output value	Impl./ Appl.	42%	20%	**0%**	**0%**	**0%**
4	Missing or delayed CAN messages or messages with wrong values if subsequent input signal values change quickly.	Appl.	83%	40%	91%	100%	75%
5	Wrong or delayed results as soon as different values are fed into the four measurement channels while changes of subsequent input signal values are limited to ~3.5% (200Hz) of the max. frequency → et al.: faulty determination of the measurement interval	Appl.	n.a.	n.a.	82%	90%	75%
6	Test cases with only equal test values at all inputs do not always lead to 4 identical results (not necessarily a failure).	Impl.	33%	30%	64%	20%	42%
7	Device stops sending of CAN messages as soon as the input frequency has been above a certain threshold**	Impl.	**0%**	**0%**	9%	**0%**	**0%**
8	No messages after reset if the input signal frequency is low	Impl.	**0%**	**0%**	**0%**	**0%**	8%
9	Device stops sending of CAN messages as soon as two buttons are pressed sequentially with high frequency → probably leading to an undefined state in state machine	Impl.	n.a.	n.a.	**0%**	10%	**0%**
10	Testmessage counter does not always start with 0 as specified → at least in some cases: faulty interaction between testmessage counter and sending method	Impl.	n.a.	n.a.	27%	60%	17%
11	Testmessage counter is not incremented correctly (is incremented by more than 1) → faulty interaction between testmessage counter and sending method	Impl.	n.a.	n.a.	**0%**	60%	**0%**
12	User input via buttons changes the number of wrong values for the worse → real-time properties affected by user input	Impl.	n.a.	n.a.	27%	**0%**	8%
		number of versions used for analysis:	12	10	11	10	12

* only elementary acceptance test
** close above the maximum input signal frequency explicitly specified if no button in pressed
Impl. = Implementation, Appl. = Application, Spec. = Specification
Exp.3: versions have been debugged according to individual test and review reports

Fig. 2. Failures found during Evaluation

No.	Problem	Type of problem	identified directly		identified indirectly	
			by review	by test	by review	by test
1	Spec.: Identify if one wheel is > 50% faster/slower than the other wheel on the same axis: 50% faster ≠ 50% slower	ambigious specification	1	0	1	1
2	Spec.: *Send a CAN message containing the peak speed values since the last request (…)*: Not defined if values occuring during test messages should be considered for peak value determination	ambigious specification	1	1	0	0
3	*As above, but*: Not defined if a second peak message must be sent if the peak message process is interrupted	ambigious specification	1	0	0	0
4	Spec: *Displaying of values on LEDs*: Not defined if LEDs must be switched ON or OFF in case of a logic 1	ambigious specification	2	0	0	0
5	Numbering of testmessage starts with 0x01 instead ot 0x00 (present in 5 versions, identified in 4 versions)	implementation problem	3	1	0	0
6	Fast changes in the input frequencies lead to response times longer than specified (present in all undebugged versions)	application problem	8	9	0	0
7	CAN transmit LED is activated in cases in which no CAN message is sent	implementation problem	1	0	0	0
8	No CAN messages sent if input frequency is zero or very low (present in at least two, identified in one version)	implementation problem	1	1	0	0

Fig. 3. Problems identified by review and test

not been identified by any experiment participant. For this reason we conducted a third experiment including a verification part consisting of review and testing as described above. The most common failures/problems identified during this verification part are listed in Fig. 3. A high number of the failures result from ambiguous statements in the specification (No. 1-4, Fig. 3). These problems occurred in every review and testing process and one might wonder why they were not identified by all of the groups. However, ambiguities were usually identified only if the verification group understood the specification differently than the implementation group which occurred only in the minority of the cases.

The problem according to fast changes of frequency values at the inputs (No. 6) has been identified by 5 of the 6 initial review groups as well as by 5 of the 6 initial test groups. This results might have influenced the second verification part, but surprisingly the problem was identified in less versions in the second part of the verification (only 3/6 of review groups and 4/6 of test groups identified the problem). However, it has to be mentioned that the resulting failures according to this problem differ in intensity and might be easier to find in some versions. Additionally, few review groups claimed that statements about the timing behavior of the code were not possible by review.

Further implementation faults in one or more versions have been identified (No. 5, 7, 8). Of special interest is problem No. 5, as it occurred in several versions: A test counter should be realized starting with 0x00 and incrementing by 1 with every CAN message sent. In 5 of the 12 undebugged versions the test counter did not start with 0x00 but with 0x01. One reason for this failure was that the line of code for the incrementation was placed incorrectly (incrementation took place before the counter value was read the first time). This problem was identified in 4/5 of the versions, mostly by review.

In a last step, all experiment participants had the opportunity to improve their versions. The results are listed in the last column of Fig. 2 (Exp.3). While many failures were identified and removed, the mitigation of some failures (as No. 4 and 5 of Fig. 2) would have needed major changes in the software architecture. According to limited time for these changes, many final versions of experiment 3 still contain faults (mostly: implementation cannot deal with fast changes of the input values within the specified time).

4 Fault Classification & Potentials of the Approaches

During evaluation of the NVP experiment results, it became obvious that different sources exist for the failures found. Those failure sources (faults) have been identified as follows:

- **Specification specific faults:** the specification was misleading or ambiguously.
- **Application specific faults:** application specific problems and challenges have not been understood and thus have not been handled sufficiently (e.g. forget to handle a certain scenario/input constellation).
- **Implementation specific faults:** specification and application specific problems have been identified correctly, but faults have been made during implementation (e.g. incomplete case structure).

The failures found in the NVP experiment have been analyzed with respect to these fault categories and many implementation specific faults identified in Fig. 2 could be mitigated by diverse hardware NVP. However, even implementation specific faults as No. 10 (test message counter, see Section 3.3) occurred in several versions developed independently on diverse hardware platforms.

While it is stated in [BBvdM04] that the specification must be correct to allow successful NVP, potentials of NVP to deal with specification problems are seen in other publications [GR01]. Some specification specific faults could be tolerated by diverse hardware NVP in our experiments, if one hardware platform was guiding to the correct implementation, as it was the case for example in failure No.1 (Fig. 2) in the first experiment. Other specification specific faults can only be found if different teams will interpret the specification differently. According to our results, different developers interpret the specification differently, but we see no hint that the **majority** of the resulting versions is correct. For this reason, NVP might uncover specification problems, but redundancy concepts based on majority voting, as for example a two out of three (2oo3, TMR) system, would be probably no solution to mask the specification specific faults present in our experiment data.

As a third fault category, we introduced application specific faults which are, beside specification specific faults, a challenging problem in NVP. Despite the immense effort put into different development processes, languages and programming styles by using completely different hardware platforms in our NVP experiments, several problems remained the same in all implementations leading to identical wrong results in several cases. These problems (No. 2-5, Fig. 2) result from the application itself, are implementation independent and thus cannot be avoided by this approach of NVP. A solution might be *functional diversity* as described in [LPS99] which offers higher independence of failure behavior according to different functionalities implemented in the diverse software version (although this approach still comprises certain risks of common mode failures [LPS99]).

In the following, the results of our application of NVP are compared with those of the third experiment in which review and testing were applied for reliability improvement. They were compared with the help of the failures presented in Fig. 2 and Fig. 3 and the results, presented in Fig.4 qualitatively, are discussed in the following.

As expected, diverse hardware NVP allowed to mitigate most of the implementation specific faults (e.g. No.7-9, 11, 12 in Fig. 2). One exception was the implementation of the test counter (No. 10). This easy task was faulty in several versions for the reasons already described. In case of review and testing, many, but not all implementation specific problems were identified. In case of non real-time tasks reviews uncovered more faults while testing was more successful in case of real-time functionalities. For this reason, high potentials of implementation specific fault mitigation are assigned for NVP (not the best value according to common mode failure described), closely followed by test and review, since many implementation specific problems had not been identified by all reviewers/testers.

Application specific faults were present in the majority of all versions, even in those created on different hardware platforms. Some of these application specific faults occurred less often on the first hardware while others were present less often

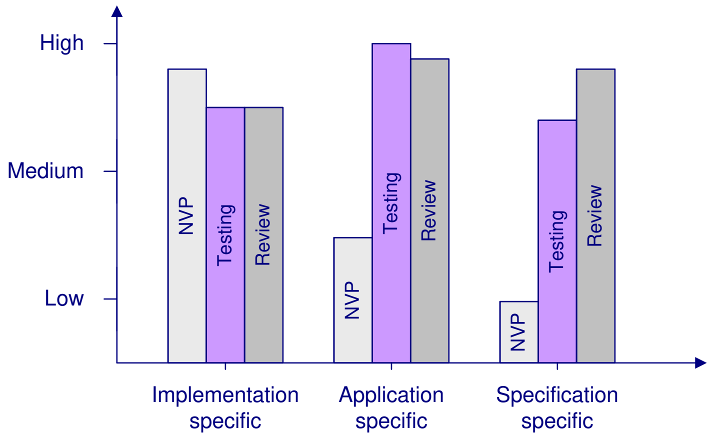

Fig. 4. Failure mitigation potentials of different approaches with respect to fault categories

on the other hardware. However, these differences were usually small (exceptions are No.2, Exp.1 and No.6, Exp2. in Fig. 2). For this reason, only low to medium potentials are seen for diverse hardware NVP to mitigate application specific faults. On the other hand, testing and review discovered most application specific problem as No.6 in Fig. 3. With respect to application specific faults, testing showed a little more advantages in comparison to review since also unexpected faults were identified during testing while reviewers typically concentrated on finding known problems in the code.

Finally, specification specific faults were a problem for all three approaches. In case of diverse hardware NVP, only few specification specific faults could be avoided as described above. In case of testing and review, all known specification problems had been identified, but in several cases only by a minority of the groups (No.1-4 in Fig. 3). Review seemed to have the highest potentials to reveal specification specific problems, since the specification had been analyzed closely for review, while it was used only for test case generation in the test process.

Summarizing, diverse hardware NVP, review and testing showed different potentials of fault mitigation with respect to the three categories of failure sources as depicted in Fig. 4. While further empirical results are needed to help designers of safety critical embedded software to apply the optimal combination of reliability improvement approaches, the results of this report dealing with three important approaches are a first step in this direction. Further on, the categorization into specification specific, application specific and implementation specific faults allows a useful and systematic comparison of the different approaches.

5 Threats on Validity

In the following two subsections, the results presented in this report were analyzed with respect to internal and external validity. In this context, *internal validity* represents the correctness of the experiment itself while *external validity* represents portability of the results to other applications [WRH+00].

5.1 Threats on Internal Validity

With respect to the first two experiments (NVP) two threats on internal validity have to be considered. According to questionnaires, handed out at the beginning and the end of all lab courses, students were more experienced in C/microcontrollers than in VHDL/FPGAs. This different previous knowledge was tried to adjust by a two day introductory course (described in [SWK05]) prior to the experiment. The different skills in the programming languages might have influenced the structure and complexity of the resulting codes and thus might have influenced the results. Moreover, one compilation of VHDL code took up to 2 min while C code was compiled in a few seconds. This aspect might have influenced the development. Nevertheless, we do not see any threat on internal validity since this is a given difference between FPGA and MCU programming.

Although the students were asked not to exchange information of their way of review and test, this aspect could not be avoided completely and has to be considered as threat on internal validity in the third experiment. Accordingly, the number of groups which found a specific fault might have been smaller if information exchange between the groups could have been avoided completely.

5.2 Threats on External Validity

As in any other experiment, the results could depend on the type and the complexity of the task used. The task used in our experiments was an automotive real-time application which, in our opinion had typical properties of small to medium sized embedded applications. However, additional experiments are desirable to minimize potential dependencies between the task and the results.

The faults found by review and testing might have been influenced by the review technique used. In contrast to formalized reviews with several reviewers as for example *code inspection* described in [Fag76] we used only two reviewers and the guidelines for the reviewers were limited to the review report form. For testing, black box testing was applied which is the most popular testing technique for the verification of real-time systems, but other test techniques might be possible. In both cases, review and testing, it has to be considered that the students had implemented this application by themselves before and thus had a good understanding of the problems which might have eased the review and testing process.

Finally, an important point concerning external validity is the quality of participants regarding experience and development knowledge. Although this problem certainly is applicable on this experiment, the objective here was to show a general difference of an effect. Using students for empirical evaluations is a viable approach referring to [SAA+03]. In addition, in [BDW02] it is stated that no difference of programming expertise between professional and non-professional developers could be found, while [HRW00] states that at least last-year software engineering students and professionals have a comparable assessment ability.

6 Conclusion and Future Work

6.1 Conclusion

Experiments have been conducted at our institute investigating the potentials of applications of diverse hardware NVP, testing and reviews for embedded systems.

During the evaluation of our empirical results, three major fault categories were identified, namely implementation specific, application specific and specification specific faults. The three different approaches analyzed in this report showed different potentials with respect to these three fault categories.

During the analysis of the two NVP experiments, high numbers of dependent faults had been found which resulted from application specific faults. In other words, the specification was correct and has been understood by the developer, but during implementation similar or identical faults have been introduced into many of the versions developed independently on different hardware platforms. The reason for this common mode failures are application specific difficulties, which exist independently from the implementation. A similar problem exists for specification specific problems. Although often stated that a correct specification is mandatory for the NVP approach, several ambiguities could be uncovered by NVP, or even mitigated by an implementation on one hardware platform (No.1, Exp1, Fig. 2). However, our results show no hint that the majority of implementations delivers correct results with respect to the intended behavior (leading to the mentioned problem for approaches based on majority voting). For the third fault category, namely implementation specific faults, it had been expected that diverse hardware NVP would allow maximum diversity between the faults in the different versions. However, even in this case in which the specification was correct and easy to understand and no application specific difficulties were present, at least one common mode failure has been introduced in several versions (No.10 in Fig. 2). Review and testing showed medium to high potentials with respect to all three fault categories. With respect to specification specific faults, reviews showed the highest potentials. It seems that during review the specification is read more intensely than during testing in which the specification is used only for test case generation. In case of application specific faults, testing showed the highest potentials. A reason might be that, while review focuses on finding known problems in the code, testing could reveal unexpected faults. In the case of implementation specific faults, testing and review showed lower potentials for reliability improvement than our NVP approach. These implementation specific faults were often related to real-time requirements which are comparatively hard to detect by testing and review.

Finally, the results have been discussed with respect to internal and external validity. According to this discussion, it has to be mentioned that the aim of this report is to emphasize the special advantages and disadvantages of the different approaches presented in this report with respect to different sources of faults. However, additional experiments are needed to support the results presented in this report and investigations of other *constructive* (design processes, functional diversity, specific architectures, etc.) and *analytical* (specific testing approaches, model checking, etc.) reliability measures are desirable.

Summarizing, diverse hardware NVP was able to mitigate many (but not all) implementation specific faults present in our experiments while potentials to mitigate application and specification specific faults were comparatively low. Reviews were especially useful to uncover problems in the specification while testing was especially useful to detect application specific faults. To apply the results, each safety critical application has to be analyzed carefully to determine the application specific needs and potentials of fault mitigation. Based on this analysis, in combination with empirical results as those presented in this report, a combination of suitable approaches can be applied to achieve highly reliable embedded software.

6.2 Future Work

The potentials of review and testing might be different for FPGAs. Approaches of testing and of formal verification could benefit from the program structure of these devices (e.g. no problems according to interrupts). Reviews might benefit from possible separation of concerns in FPGAs (parallel structure) on the one hand, but interaction between several parallel units in real-time might be harder to understand on the other hand. These potentials are currently analyzed at our institute [SK07b].

With the application of automatic code generation in the automotive industry, the idea of using this code generated automatically as one version in an NVP approach is arising. The second version would be implemented manually allowing certain diversity (approach with two fail silent units programmed with diverse software to avoid shut down of both units according to a software fault). This approach might have advantages and should be further evaluated.

References

[BBvdM04] J.G.W. Bentley, P.G. Bishop, and M.J.P. van der Meulen. An empirical exploration of the difficulty function. In *Computer Safety, Reliability and Security (Safecomp)*, 2004.

[BDW02] Jean-Marie Burkhardt, Françoise Deétienne, and Susan Wiedenbeck. Object-oriented program comprehension: Effect of expertise, task and phase. *Empirical Software Engineering*, 7:115–156, 2002.

[CA77] L. Chen and A. Avizienis. On the implementation of n-version programming for software fault tolerance during program execution. In *International Computer Software and Applications Conference (COMPSAC)*, 1977.

[CL04] X. Cai and M. R. Lyu. An empirical study on reliability modeling for diverse software systems. In *15th International Symposium on Software Reliability Engineering (ISSRE)*, 2004.

[Fag76] Michael Fagan. Design and code inspections to reduce errors in program development. Technical report, IBM, 1976.

[GR01] K. E. Grosspietsch and A. Romanovsky. An evolutionary and adaptive approach for n-version programming. In *IEEE Euromicro Conference*, 2001.

[HRW00] Martin Höst, Björn Regnell, and Claes Wohlin. Using students as subjects - a comparative study of students and professionals in lead-time impact assessment. *Empirical Software Engineering*, 5:201–214, 2000.

[KL86] J. C. Knight and N. G. Leveson. An experimental evaluation of the assumption of independence in multiversion programming. *IEEE Trans. Softw. Eng.*, 12, 1986.

[LH93] M. R. Lyu and Yu-Tao He. Improving the n-version programming process through the evolution of a design paradigm. *IEEE Transactions on Reliability*, 42, 1993.

[LM89] B. Littlewood and D. R. Miller. Conceptual modeling of coincident failures in multiversion software. In *IEEE Trans. Softw. Eng.*, 1989.

[LPS99] B. Littlewood, P. Popov, and L. Strigini. A note on modelling functional diversity. In *Reliability Engineering an System Safety*, 1999.

[LPS00] Bev Littlewood, Peter Popov, and Lorenzo Strigini. N-version design versus on good version. In *International Conference on Dependable Systems & Networks (DSN)*, 2000.

[PSL00] Peter Popov, Lorenzo Strigini, and Bev Littlewood. Choosing between fault-tolerance and increased v&v for improving reliability. In *International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, 2000.

[PvSK90] David L. Parnas, John van Schouwen, and Shu Po Kwan. Evaluation of safety-critical software. *Communications of the ACM*, 33:636–648, 1990.

[SAA+03] Dag I. K. Sjøberg, Bente Anda, Erik Arisholm, Tore Dybå, Magne Jørgensen, Amela Karahasanović, and Marek Vokáč. Challenges and recommendations when increasing the realism of controlled software engineering experiments. *ESERNET 2001-2003, LNCS 2765*, pages 24–38, 2003.

[SK07a] Falk Salewski and Stefan Kowalewski. Achieving highly reliable embedded software: an empirical evaluation of different approaches. In *Proceedings of SafeComp 2007, Lecture Notes in Computer Science*, 2007.

[SK07b] Falk Salewski and Stefan Kowalewski. The effect of hardware platform selection on safety-critical software in embedded systems: Empirical evaluations. In *IEEE Second International Symposium on Industrial Embedded Systems (SIES)*, 2007.

[SK07c] Falk Salewski and Stefan Kowalewski. Testing issues in empirical reliability evaluation of embedded real-time systems. In *IEEE Real-Time and Embedded Technology and Applications Symposium, WiP session (RTAS)*, 2007.

[Sto96] Neil Storey. *Safety-Critical Computer Systems*. Prentice Hall, 1996.

[SWK05] Falk Salewski, Dirk Wilking, and Stefan Kowalewski. Diverse hardware platforms in embedded systems lab courses: A way to teach the differences. In *First Workshop on Embedded System Education (WESE)*, volume 2. SIGBED Review, 2005.

[SWK06] Falk Salewski, Dirk Wilking, and Stefan Kowalewski. The effect of diverse hardware platforms on n-version programming in embedded systems - an empirical evaluation. In *3rd International Workshop on Dependable Embedded Sytems (WDES)*, 2006.

[VWVW96] H.P.E. Vranken, M.F. Witteman, and R.C. Van Wuijtswinkel. Design for testability in hardware software systems. *IEEE Design & Test of Computers*, 13:79–86, 1996.

[WRH+00] Claes Wohlin, Per Runeson, Martin Höst, Magnus Ohlson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering*. Kluwer Academic Publishers, 2000.

Aachener Informatik-Berichte

This is the list of all technical reports since 1987. To obtain copies of reports please consult http://aib.informatik.rwth-aachen.de/ or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: `biblio@informatik.rwth-aachen.de`

1987-01 *	Fachgruppe Informatik: Jahresbericht 1986
1987-02 *	David de Frutos Escrig, Klaus Indermark: Equivalence Relations of Non-Deterministic Ianov-Schemes
1987-03 *	Manfred Nagl: A Software Development Environment based on Graph Technology
1987-04 *	Claus Lewerentz, Manfred Nagl, Bernhard Westfechtel: On Integration Mechanisms within a Graph-Based Software Development Environment
1987-05 *	Reinhard Rinn: Über Eingabeanomalien bei verschiedenen Inferenzmodellen
1987-06 *	Werner Damm, Gert Döhmen: Specifying Distributed Computer Architectures in AADL*
1987-07 *	Gregor Engels, Claus Lewerentz, Wilhelm Schäfer: Graph Grammar Engineering: A Software Specification Method
1987-08 *	Manfred Nagl: Set Theoretic Approaches to Graph Grammars
1987-09 *	Claus Lewerentz, Andreas Schürr: Experiences with a Database System for Software Documents
1987-10 *	Herbert Klaeren, Klaus Indermark: A New Implementation Technique for Recursive Function Definitions
1987-11 *	Rita Loogen: Design of a Parallel Programmable Graph Reduction Machine with Distributed Memory
1987-12	J. Börstler, U. Möncke, R. Wilhelm: Table compression for tree automata
1988-01 *	Gabriele Esser, Johannes Rückert, Frank Wagner Gesellschaftliche Aspekte der Informatik
1988-02 *	Peter Martini, Otto Spaniol: Token-Passing in High-Speed Backbone Networks for Campus-Wide Environments
1988-03 *	Thomas Welzel: Simulation of a Multiple Token Ring Backbone
1988-04 *	Peter Martini: Performance Comparison for HSLAN Media Access Protocols
1988-05 *	Peter Martini: Performance Analysis of Multiple Token Rings
1988-06 *	Andreas Mann, Johannes Rückert, Otto Spaniol: Datenfunknetze
1988-07 *	Andreas Mann, Johannes Rückert: Packet Radio Networks for Data Exchange
1988-08 *	Andreas Mann, Johannes Rückert: Concurrent Slot Assignment Protocol for Packet Radio Networks
1988-09 *	W. Kremer, F. Reichert, J. Rückert, A. Mann: Entwurf einer Netzwerktopologie für ein Mobilfunknetz zur Unterstützung des öffentlichen Straßenverkehrs
1988-10 *	Kai Jakobs: Towards User-Friendly Networking
1988-11 *	Kai Jakobs: The Directory - Evolution of a Standard
1988-12 *	Kai Jakobs: Directory Services in Distributed Systems - A Survey

1988-13 * Martine Schümmer: RS-511, a Protocol for the Plant Floor
1988-14 * U. Quernheim: Satellite Communication Protocols - A Performance Comparison Considering On-Board Processing
1988-15 * Peter Martini, Otto Spaniol, Thomas Welzel: File Transfer in High Speed Token Ring Networks: Performance Evaluation by Approximate Analysis and Simulation
1988-16 * Fachgruppe Informatik: Jahresbericht 1987
1988-17 * Wolfgang Thomas: Automata on Infinite Objects
1988-18 * Michael Sonnenschein: On Petri Nets and Data Flow Graphs
1988-19 * Heiko Vogler: Functional Distribution of the Contextual Analysis in Block-Structured Programming Languages: A Case Study of Tree Transducers
1988-20 * Thomas Welzel: Einsatz des Simulationswerkzeuges QNAP2 zur Leistungsbewertung von Kommunikationsprotokollen
1988-21 * Th. Janning, C. Lewerentz: Integrated Project Team Management in a Software Development Environment
1988-22 * Joost Engelfriet, Heiko Vogler: Modular Tree Transducers
1988-23 * Wolfgang Thomas: Automata and Quantifier Hierarchies
1988-24 * Uschi Heuter: Generalized Definite Tree Languages
1989-01 * Fachgruppe Informatik: Jahresbericht 1988
1989-02 * G. Esser, J. Rückert, F. Wagner (Hrsg.): Gesellschaftliche Aspekte der Informatik
1989-03 * Heiko Vogler: Bottom-Up Computation of Primitive Recursive Tree Functions
1989-04 * Andy Schürr: Introduction to PROGRESS, an Attribute Graph Grammar Based Specification Language
1989-05 J. Börstler: Reuse and Software Development - Problems, Solutions, and Bibliography (in German)
1989-06 * Kai Jakobs: OSI - An Appropriate Basis for Group Communication?
1989-07 * Kai Jakobs: ISO's Directory Proposal - Evolution, Current Status and Future Problems
1989-08 * Bernhard Westfechtel: Extension of a Graph Storage for Software Documents with Primitives for Undo/Redo and Revision Control
1989-09 * Peter Martini: High Speed Local Area Networks - A Tutorial
1989-10 * P. Davids, Th. Welzel: Performance Analysis of DQDB Based on Simulation
1989-11 * Manfred Nagl (Ed.): Abstracts of Talks presented at the WG '89 15th International Workshop on Graphtheoretic Concepts in Computer Science
1989-12 * Peter Martini: The DQDB Protocol - Is it Playing the Game?
1989-13 * Martine Schümmer: CNC/DNC Communication with MAP
1989-14 * Martine Schümmer: Local Area Networks for Manufactoring Environments with hard Real-Time Requirements
1989-15 * M. Schümmer, Th. Welzel, P. Martini: Integration of Field Bus and MAP Networks - Hierarchical Communication Systems in Production Environments
1989-16 * G. Vossen, K.-U. Witt: SUXESS: Towards a Sound Unification of Extensions of the Relational Data Model

1989-17 * J. Derissen, P. Hruschka, M.v.d. Beeck, Th. Janning, M. Nagl: Integrating Structured Analysis and Information Modelling
1989-18 A. Maassen: Programming with Higher Order Functions
1989-19 * Mario Rodriguez-Artalejo, Heiko Vogler: A Narrowing Machine for Syntax Directed BABEL
1989-20 H. Kuchen, R. Loogen, J.J. Moreno Navarro, M. Rodriguez Artalejo: Graph-based Implementation of a Functional Logic Language
1990-01 * Fachgruppe Informatik: Jahresbericht 1989
1990-02 * Vera Jansen, Andreas Potthoff, Wolfgang Thomas, Udo Wermuth: A Short Guide to the AMORE System (Computing Automata, MOnoids and Regular Expressions)
1990-03 * Jerzy Skurczynski: On Three Hierarchies of Weak SkS Formulas
1990-04 R. Loogen: Stack-based Implementation of Narrowing
1990-05 H. Kuchen, A. Wagener: Comparison of Dynamic Load Balancing Strategies
1990-06 * Kai Jakobs, Frank Reichert: Directory Services for Mobile Communication
1990-07 * Kai Jakobs: What's Beyond the Interface - OSI Networks to Support Cooperative Work
1990-08 * Kai Jakobs: Directory Names and Schema - An Evaluation
1990-09 * Ulrich Quernheim, Dieter Kreuer: Das CCITT - Signalisierungssystem Nr. 7 auf Satellitenstrecken; Simulation der Zeichengabestrecke
1990-11 H. Kuchen, R. Loogen, J.J. Moreno Navarro, M. Rodriguez Artalejo: Lazy Narrowing in a Graph Machine
1990-12 * Kai Jakobs, Josef Kaltwasser, Frank Reichert, Otto Spaniol: Der Computer fährt mit
1990-13 * Rudolf Mathar, Andreas Mann: Analyzing a Distributed Slot Assignment Protocol by Markov Chains
1990-14 A. Maassen: Compilerentwicklung in Miranda - ein Praktikum in funktionaler Programmierung (written in german)
1990-15 * Manfred Nagl, Andreas Schürr: A Specification Environment for Graph Grammars
1990-16 A. Schürr: PROGRESS: A VHL-Language Based on Graph Grammars
1990-17 * Marita Möller: Ein Ebenenmodell wissensbasierter Konsultationen - Unterstützung für Wissensakquisition und Erklärungsfähigkeit
1990-18 * Eric Kowalewski: Entwurf und Interpretation einer Sprache zur Beschreibung von Konsultationsphasen in Expertensystemen
1990-20 Y. Ortega Mallen, D. de Frutos Escrig: A Complete Proof System for Timed Observations
1990-21 * Manfred Nagl: Modelling of Software Architectures: Importance, Notions, Experiences
1990-22 H. Fassbender, H. Vogler: A Call-by-need Implementation of Syntax Directed Functional Programming
1991-01 Guenther Geiler (ed.), Fachgruppe Informatik: Jahresbericht 1990
1991-03 B. Steffen, A. Ingolfsdottir: Characteristic Formulae for Processes with Divergence
1991-04 M. Portz: A new class of cryptosystems based on interconnection networks

1991-05 H. Kuchen, G. Geiler: Distributed Applicative Arrays
1991-06 * Ludwig Staiger: Kolmogorov Complexity and Hausdorff Dimension
1991-07 * Ludwig Staiger: Syntactic Congruences for w-languages
1991-09 * Eila Kuikka: A Proposal for a Syntax-Directed Text Processing System
1991-10 K. Gladitz, H. Fassbender, H. Vogler: Compiler-based Implementation of Syntax-Directed Functional Programming
1991-11 R. Loogen, St. Winkler: Dynamic Detection of Determinism in Functional Logic Languages
1991-12 * K. Indermark, M. Rodriguez Artalejo (Eds.): Granada Workshop on the Integration of Functional and Logic Programming
1991-13 * Rolf Hager, Wolfgang Kremer: The Adaptive Priority Scheduler: A More Fair Priority Service Discipline
1991-14 * Andreas Fasbender, Wolfgang Kremer: A New Approximation Algorithm for Tandem Networks with Priority Nodes
1991-15 J. Börstler, A. Zündorf: Revisiting extensions to Modula-2 to support reusability
1991-16 J. Börstler, Th. Janning: Bridging the gap between Requirements Analysis and Design
1991-17 A. Zündorf, A. Schürr: Nondeterministic Control Structures for Graph Rewriting Systems
1991-18 * Matthias Jarke, John Mylopoulos, Joachim W. Schmidt, Yannis Vassiliou: DAIDA: An Environment for Evolving Information Systems
1991-19 M. Jeusfeld, M. Jarke: From Relational to Object-Oriented Integrity Simplification
1991-20 G. Hogen, A. Kindler, R. Loogen: Automatic Parallelization of Lazy Functional Programs
1991-21 * Prof. Dr. rer. nat. Otto Spaniol: ODP (Open Distributed Processing): Yet another Viewpoint
1991-22 H. Kuchen, F. Lücking, H. Stoltze: The Topology Description Language TDL
1991-23 S. Graf, B. Steffen: Compositional Minimization of Finite State Systems
1991-24 R. Cleaveland, J. Parrow, B. Steffen: The Concurrency Workbench: A Semantics Based Tool for the Verification of Concurrent Systems
1991-25 * Rudolf Mathar, Jürgen Mattfeldt: Optimal Transmission Ranges for Mobile Communication in Linear Multihop Packet Radio Networks
1991-26 M. Jeusfeld, M. Staudt: Query Optimization in Deductive Object Bases
1991-27 J. Knoop, B. Steffen: The Interprocedural Coincidence Theorem
1991-28 J. Knoop, B. Steffen: Unifying Strength Reduction and Semantic Code Motion
1991-30 T. Margaria: First-Order theories for the verification of complex FSMs
1991-31 B. Steffen: Generating Data Flow Analysis Algorithms from Modal Specifications
1992-01 Stefan Eherer (ed.), Fachgruppe Informatik: Jahresbericht 1991
1992-02 * Bernhard Westfechtel: Basismechanismen zur Datenverwaltung in strukturbezogenen Hypertextsystemen
1992-04 S. A. Smolka, B. Steffen: Priority as Extremal Probability
1992-05 * Matthias Jarke, Carlos Maltzahn, Thomas Rose: Sharing Processes: Team Coordination in Design Repositories

1992-06 O. Burkart, B. Steffen: Model Checking for Context-Free Processes

1992-07 * Matthias Jarke, Klaus Pohl: Information Systems Quality and Quality Information Systems

1992-08 * Rudolf Mathar, Jürgen Mattfeldt: Analyzing Routing Strategy NFP in Multihop Packet Radio Networks on a Line

1992-09 * Alfons Kemper, Guido Moerkotte: Grundlagen objektorientierter Datenbanksysteme

1992-10 Matthias Jarke, Manfred Jeusfeld, Andreas Miethsam, Michael Gocek: Towards a logic-based reconstruction of software configuration management

1992-11 Werner Hans: A Complete Indexing Scheme for WAM-based Abstract Machines

1992-12 W. Hans, R. Loogen, St. Winkler: On the Interaction of Lazy Evaluation and Backtracking

1992-13 * Matthias Jarke, Thomas Rose: Specification Management with CAD

1992-14 Th. Noll, H. Vogler: Top-down Parsing with Simultaneous Evaluation on Noncircular Attribute Grammars

1992-15 A. Schuerr, B. Westfechtel: Graphgrammatiken und Graphersetzungssysteme(written in german)

1992-16 * Graduiertenkolleg Informatik und Technik (Hrsg.): Forschungsprojekte des Graduiertenkollegs Informatik und Technik

1992-17 M. Jarke (ed.): ConceptBase V3.1 User Manual

1992-18 * Clarence A. Ellis, Matthias Jarke (Eds.): Distributed Cooperation in Integrated Information Systems - Proceedings of the Third International Workshop on Intelligent and Cooperative Information Systems

1992-19-00 H. Kuchen, R. Loogen (eds.): Proceedings of the 4th Int. Workshop on the Parallel Implementation of Functional Languages

1992-19-01 G. Hogen, R. Loogen: PASTEL - A Parallel Stack-Based Implementation of Eager Functional Programs with Lazy Data Structures (Extended Abstract)

1992-19-02 H. Kuchen, K. Gladitz: Implementing Bags on a Shared Memory MIMD-Machine

1992-19-03 C. Rathsack, S.B. Scholz: LISA - A Lazy Interpreter for a Full-Fledged Lambda-Calculus

1992-19-04 T.A. Bratvold: Determining Useful Parallelism in Higher Order Functions

1992-19-05 S. Kahrs: Polymorphic Type Checking by Interpretation of Code

1992-19-06 M. Chakravarty, M. Köhler: Equational Constraints, Residuation, and the Parallel JUMP-Machine

1992-19-07 J. Seward: Polymorphic Strictness Analysis using Frontiers (Draft Version)

1992-19-08 D. Gärtner, A. Kimms, W. Kluge: pi-Red^+ - A Compiling Graph-Reduction System for a Full Fledged Lambda-Calculus

1992-19-09 D. Howe, G. Burn: Experiments with strict STG code

1992-19-10 J. Glauert: Parallel Implementation of Functional Languages Using Small Processes

1992-19-11 M. Joy, T. Axford: A Parallel Graph Reduction Machine

1992-19-12 A. Bennett, P. Kelly: Simulation of Multicache Parallel Reduction

1992-19-13 K. Langendoen, D.J. Agterkamp: Cache Behaviour of Lazy Functional Programs (Working Paper)

1992-19-14 K. Hammond, S. Peyton Jones: Profiling scheduling strategies on the GRIP parallel reducer

1992-19-15 S. Mintchev: Using Strictness Information in the STG-machine

1992-19-16 D. Rushall: An Attribute Grammar Evaluator in Haskell

1992-19-17 J. Wild, H. Glaser, P. Hartel: Statistics on storage management in a lazy functional language implementation

1992-19-18 W.S. Martins: Parallel Implementations of Functional Languages

1992-19-19 D. Lester: Distributed Garbage Collection of Cyclic Structures (Draft version)

1992-19-20 J.C. Glas, R.F.H. Hofman, W.G. Vree: Parallelization of Branch-and-Bound Algorithms in a Functional Programming Environment

1992-19-21 S. Hwang, D. Rushall: The nu-STG machine: a parallelized Spineless Tagless Graph Reduction Machine in a distributed memory architecture (Draft version)

1992-19-22 G. Burn, D. Le Metayer: Cps-Translation and the Correctness of Optimising Compilers

1992-19-23 S.L. Peyton Jones, P. Wadler: Imperative functional programming (Brief summary)

1992-19-24 W. Damm, F. Liu, Th. Peikenkamp: Evaluation and Parallelization of Functions in Functional + Logic Languages (abstract)

1992-19-25 M. Kesseler: Communication Issues Regarding Parallel Functional Graph Rewriting

1992-19-26 Th. Peikenkamp: Charakterizing and representing neededness in functional loginc languages (abstract)

1992-19-27 H. Doerr: Monitoring with Graph-Grammars as formal operational Models

1992-19-28 J. van Groningen: Some implementation aspects of Concurrent Clean on distributed memory architectures

1992-19-29 G. Ostheimer: Load Bounding for Implicit Parallelism (abstract)

1992-20 H. Kuchen, F.J. Lopez Fraguas, J.J. Moreno Navarro, M. Rodriguez Artalejo: Implementing Disequality in a Lazy Functional Logic Language

1992-21 H. Kuchen, F.J. Lopez Fraguas: Result Directed Computing in a Functional Logic Language

1992-22 H. Kuchen, J.J. Moreno Navarro, M.V. Hermenegildo: Independent AND-Parallel Narrowing

1992-23 T. Margaria, B. Steffen: Distinguishing Formulas for Free

1992-24 K. Pohl: The Three Dimensions of Requirements Engineering

1992-25 * R. Stainov: A Dynamic Configuration Facility for Multimedia Communications

1992-26 * Michael von der Beeck: Integration of Structured Analysis and Timed Statecharts for Real-Time and Concurrency Specification

1992-27 W. Hans, St. Winkler: Aliasing and Groundness Analysis of Logic Programs through Abstract Interpretation and its Safety

1992-28 * Gerhard Steinke, Matthias Jarke: Support for Security Modeling in Information Systems Design

1992-29 B. Schinzel: Warum Frauenforschung in Naturwissenschaft und Technik

1992-30 A. Kemper, G. Moerkotte, K. Peithner: Object-Orientation Axiomatised by Dynamic Logic

1992-32 * Bernd Heinrichs, Kai Jakobs: Timer Handling in High-Performance Transport Systems

1992-33 * B. Heinrichs, K. Jakobs, K. Lenßen, W. Reinhardt, A. Spinner: Euro-Bridge: Communication Services for Multimedia Applications

1992-34 C. Gerlhof, A. Kemper, Ch. Kilger, G. Moerkotte: Partition-Based Clustering in Object Bases: From Theory to Practice

1992-35 J. Börstler: Feature-Oriented Classification and Reuse in IPSEN

1992-36 M. Jarke, J. Bubenko, C. Rolland, A. Sutcliffe, Y. Vassiliou: Theories Underlying Requirements Engineering: An Overview of NATURE at Genesis

1992-37 * K. Pohl, M. Jarke: Quality Information Systems: Repository Support for Evolving Process Models

1992-38 A. Zuendorf: Implementation of the imperative / rule based language PROGRES

1992-39 P. Koch: Intelligentes Backtracking bei der Auswertung funktional-logischer Programme

1992-40 * Rudolf Mathar, Jürgen Mattfeldt: Channel Assignment in Cellular Radio Networks

1992-41 * Gerhard Friedrich, Wolfgang Neidl: Constructive Utility in Model-Based Diagnosis Repair Systems

1992-42 * P. S. Chen, R. Hennicker, M. Jarke: On the Retrieval of Reusable Software Components

1992-43 W. Hans, St.Winkler: Abstract Interpretation of Functional Logic Languages

1992-44 N. Kiesel, A. Schuerr, B. Westfechtel: Design and Evaluation of GRAS, a Graph-Oriented Database System for Engineering Applications

1993-01 * Fachgruppe Informatik: Jahresbericht 1992

1993-02 * Patrick Shicheng Chen: On Inference Rules of Logic-Based Information Retrieval Systems

1993-03 G. Hogen, R. Loogen: A New Stack Technique for the Management of Runtime Structures in Distributed Environments

1993-05 A. Zündorf: A Heuristic for the Subgraph Isomorphism Problem in Executing PROGRES

1993-06 A. Kemper, D. Kossmann: Adaptable Pointer Swizzling Strategies in Object Bases: Design, Realization, and Quantitative Analysis

1993-07 * Graduiertenkolleg Informatik und Technik (Hrsg.): Graduiertenkolleg Informatik und Technik

1993-08 * Matthias Berger: k-Coloring Vertices using a Neural Network with Convergence to Valid Solutions

1993-09 M. Buchheit, M. Jeusfeld, W. Nutt, M. Staudt: Subsumption between Queries to Object-Oriented Databases

1993-10 O. Burkart, B. Steffen: Pushdown Processes: Parallel Composition and Model Checking

1993-11 * R. Große-Wienker, O. Hermanns, D. Menzenbach, A. Pollacks, S. Repetzki, J. Schwartz, K. Sonnenschein, B. Westfechtel: Das SUKITS-Projekt: A-posteriori-Integration heterogener CIM-Anwendungssysteme

1993-12 * Rudolf Mathar, Jürgen Mattfeldt: On the Distribution of Cumulated Interference Power in Rayleigh Fading Channels

1993-13 O. Maler, L. Staiger: On Syntactic Congruences for omega-languages

1993-14 M. Jarke, St. Eherer, R. Gallersdoerfer, M. Jeusfeld, M. Staudt: ConceptBase - A Deductive Object Base Manager

1993-15 M. Staudt, H.W. Nissen, M.A. Jeusfeld: Query by Class, Rule and Concept

1993-16 * M. Jarke, K. Pohl, St. Jacobs et al.: Requirements Engineering: An Integrated View of Representation Process and Domain

1993-17 * M. Jarke, K. Pohl: Establishing Vision in Context: Towards a Model of Requirements Processes

1993-18 W. Hans, H. Kuchen, St. Winkler: Full Indexing for Lazy Narrowing

1993-19 W. Hans, J.J. Ruz, F. Saenz, St. Winkler: A VHDL Specification of a Shared Memory Parallel Machine for Babel

1993-20 * K. Finke, M. Jarke, P. Szczurko, R. Soltysiak: Quality Management for Expert Systems in Process Control

1993-21 M. Jarke, M.A. Jeusfeld, P. Szczurko: Three Aspects of Intelligent Cooperation in the Quality Cycle

1994-01 Margit Generet, Sven Martin (eds.), Fachgruppe Informatik: Jahresbericht 1993

1994-02 M. Lefering: Development of Incremental Integration Tools Using Formal Specifications

1994-03 * P. Constantopoulos, M. Jarke, J. Mylopoulos, Y. Vassiliou: The Software Information Base: A Server for Reuse

1994-04 * Rolf Hager, Rudolf Mathar, Jürgen Mattfeldt: Intelligent Cruise Control and Reliable Communication of Mobile Stations

1994-05 * Rolf Hager, Peter Hermesmann, Michael Portz: Feasibility of Authentication Procedures within Advanced Transport Telematics

1994-06 * Claudia Popien, Bernd Meyer, Axel Kuepper: A Formal Approach to Service Import in ODP Trader Federations

1994-07 P. Peters, P. Szczurko: Integrating Models of Quality Management Methods by an Object-Oriented Repository

1994-08 * Manfred Nagl, Bernhard Westfechtel: A Universal Component for the Administration in Distributed and Integrated Development Environments

1994-09 * Patrick Horster, Holger Petersen: Signatur- und Authentifikationsverfahren auf der Basis des diskreten Logarithmusproblems

1994-11 A. Schürr: PROGRES, A Visual Language and Environment for PROgramming with Graph REwrite Systems

1994-12 A. Schürr: Specification of Graph Translators with Triple Graph Grammars

1994-13 A. Schürr: Logic Based Programmed Structure Rewriting Systems

1994-14 L. Staiger: Codes, Simplifying Words, and Open Set Condition

1994-15 * Bernhard Westfechtel: A Graph-Based System for Managing Configurations of Engineering Design Documents

1994-16 P. Klein: Designing Software with Modula-3

1994-17 I. Litovsky, L. Staiger: Finite acceptance of infinite words

1994-18 G. Hogen, R. Loogen: Parallel Functional Implementations: Graphbased vs. Stackbased Reduction

1994-19 M. Jeusfeld, U. Johnen: An Executable Meta Model for Re-Engineering of Database Schemas

1994-20 * R. Gallersdörfer, M. Jarke, K. Klabunde: Intelligent Networks as a Data Intensive Application (INDIA)

1994-21 M. Mohnen: Proving the Correctness of the Static Link Technique Using Evolving Algebras

1994-22 H. Fernau, L. Staiger: Valuations and Unambiguity of Languages, with Applications to Fractal Geometry

1994-24 * M. Jarke, K. Pohl, R. Dömges, St. Jacobs, H. W. Nissen: Requirements Information Management: The NATURE Approach

1994-25 * M. Jarke, K. Pohl, C. Rolland, J.-R. Schmitt: Experience-Based Method Evaluation and Improvement: A Process Modeling Approach

1994-26 * St. Jacobs, St. Kethers: Improving Communication and Decision Making within Quality Function Deployment

1994-27 * M. Jarke, H. W. Nissen, K. Pohl: Tool Integration in Evolving Information Systems Environments

1994-28 O. Burkart, D. Caucal, B. Steffen: An Elementary Bisimulation Decision Procedure for Arbitrary Context-Free Processes

1995-01 * Fachgruppe Informatik: Jahresbericht 1994

1995-02 Andy Schürr, Andreas J. Winter, Albert Zündorf: Graph Grammar Engineering with PROGRES

1995-03 Ludwig Staiger: A Tight Upper Bound on Kolmogorov Complexity by Hausdorff Dimension and Uniformly Optimal Prediction

1995-04 Birgitta König-Ries, Sven Helmer, Guido Moerkotte: An experimental study on the complexity of left-deep join ordering problems for cyclic queries

1995-05 Sophie Cluet, Guido Moerkotte: Efficient Evaluation of Aggregates on Bulk Types

1995-06 Sophie Cluet, Guido Moerkotte: Nested Queries in Object Bases

1995-07 Sophie Cluet, Guido Moerkotte: Query Optimization Techniques Exploiting Class Hierarchies

1995-08 Markus Mohnen: Efficient Compile-Time Garbage Collection for Arbitrary Data Structures

1995-09 Markus Mohnen: Functional Specification of Imperative Programs: An Alternative Point of View of Functional Languages

1995-10 Rainer Gallersdörfer, Matthias Nicola: Improving Performance in Replicated Databases through Relaxed Coherency

1995-11 * M.Staudt, K.von Thadden: Subsumption Checking in Knowledge Bases

1995-12 * G.V.Zemanek, H.W.Nissen, H.Hubert, M.Jarke: Requirements Analysis from Multiple Perspectives: Experiences with Conceptual Modeling Technology

1995-13 * M.Staudt, M.Jarke: Incremental Maintenance of Externally Materialized Views

1995-14 * P.Peters, P.Szczurko, M.Jeusfeld: Oriented Information Management: Conceptual Models at Work

1995-15 * Matthias Jarke, Sudha Ram (Hrsg.): WITS 95 Proceedings of the 5th Annual Workshop on Information Technologies and Systems
1995-16 * W.Hans, St.Winkler, F.Saenz: Distributed Execution in Functional Logic Programming
1996-01 * Jahresbericht 1995
1996-02 Michael Hanus, Christian Prehofer: Higher-Order Narrowing with Definitional Trees
1996-03 * W.Scheufele, G.Moerkotte: Optimal Ordering of Selections and Joins in Acyclic Queries with Expensive Predicates
1996-04 Klaus Pohl: PRO-ART: Enabling Requirements Pre-Traceability
1996-05 Klaus Pohl: Requirements Engineering: An Overview
1996-06 * M.Jarke, W.Marquardt: Design and Evaluation of Computer–Aided Process Modelling Tools
1996-07 Olaf Chitil: The Sigma-Semantics: A Comprehensive Semantics for Functional Programs
1996-08 * S.Sripada: On Entropy and the Limitations of the Second Law of Thermodynamics
1996-09 Michael Hanus (Ed.): Proceedings of the Poster Session of ALP96 - Fifth International Conference on Algebraic and Logic Programming
1996-09-0 Michael Hanus (Ed.): Proceedings of the Poster Session of ALP 96 - Fifth International Conference on Algebraic and Logic Programming: Introduction and table of contents
1996-09-1 Ilies Alouini: An Implementation of Conditional Concurrent Rewriting on Distributed Memory Machines
1996-09-2 Olivier Danvy, Karoline Malmkjær: On the Idempotence of the CPS Transformation
1996-09-3 Víctor M. Gulías, José L. Freire: Concurrent Programming in Haskell
1996-09-4 Sébastien Limet, Pierre Réty: On Decidability of Unifiability Modulo Rewrite Systems
1996-09-5 Alexandre Tessier: Declarative Debugging in Constraint Logic Programming
1996-10 Reidar Conradi, Bernhard Westfechtel: Version Models for Software Configuration Management
1996-11 * C.Weise, D.Lenzkes: A Fast Decision Algorithm for Timed Refinement
1996-12 * R.Dömges, K.Pohl, M.Jarke, B.Lohmann, W.Marquardt: PRO-ART/CE* — An Environment for Managing the Evolution of Chemical Process Simulation Models
1996-13 * K.Pohl, R.Klamma, K.Weidenhaupt, R.Dömges, P.Haumer, M.Jarke: A Framework for Process-Integrated Tools
1996-14 * R.Gallersdörfer, K.Klabunde, A.Stolz, M.Eßmajor: INDIA — Intelligent Networks as a Data Intensive Application, Final Project Report, June 1996
1996-15 * H.Schimpe, M.Staudt: VAREX: An Environment for Validating and Refining Rule Bases
1996-16 * M.Jarke, M.Gebhardt, S.Jacobs, H.Nissen: Conflict Analysis Across Heterogeneous Viewpoints: Formalization and Visualization
1996-17 Manfred A. Jeusfeld, Tung X. Bui: Decision Support Components on the Internet

1996-18 Manfred A. Jeusfeld, Mike Papazoglou: Information Brokering: Design, Search and Transformation

1996-19 * P.Peters, M.Jarke: Simulating the impact of information flows in networked organizations

1996-20 Matthias Jarke, Peter Peters, Manfred A. Jeusfeld: Model-driven planning and design of cooperative information systems

1996-21 * G.de Michelis, E.Dubois, M.Jarke, F.Matthes, J.Mylopoulos, K.Pohl, J.Schmidt, C.Woo, E.Yu: Cooperative information systems: a manifesto

1996-22 * S.Jacobs, M.Gebhardt, S.Kethers, W.Rzasa: Filling HTML forms simultaneously: CoWeb architecture and functionality

1996-23 * M.Gebhardt, S.Jacobs: Conflict Management in Design

1997-01 Michael Hanus, Frank Zartmann (eds.): Jahresbericht 1996

1997-02 Johannes Faassen: Using full parallel Boltzmann Machines for Optimization

1997-03 Andreas Winter, Andy Schürr: Modules and Updatable Graph Views for PROgrammed Graph REwriting Systems

1997-04 Markus Mohnen, Stefan Tobies: Implementing Context Patterns in the Glasgow Haskell Compiler

1997-05 * S.Gruner: Schemakorrespondenzaxiome unterstützen die paargrammatische Spezifikation inkrementeller Integrationswerkzeuge

1997-06 Matthias Nicola, Matthias Jarke: Design and Evaluation of Wireless Health Care Information Systems in Developing Countries

1997-07 Petra Hofstedt: Taskparallele Skelette für irregulär strukturierte Probleme in deklarativen Sprachen

1997-08 Dorothea Blostein, Andy Schürr: Computing with Graphs and Graph Rewriting

1997-09 Carl-Arndt Krapp, Bernhard Westfechtel: Feedback Handling in Dynamic Task Nets

1997-10 Matthias Nicola, Matthias Jarke: Integrating Replication and Communication in Performance Models of Distributed Databases

1997-11 * R. Klamma, P. Peters, M. Jarke: Workflow Support for Failure Management in Federated Organizations

1997-13 Markus Mohnen: Optimising the Memory Management of Higher-Order Functional Programs

1997-14 Roland Baumann: Client/Server Distribution in a Structure-Oriented Database Management System

1997-15 George Botorog: High-Level Parallel Programming and the Efficient Implementation of Numerical Algorithms

1998-01 * Fachgruppe Informatik: Jahresbericht 1997

1998-02 Stefan Gruner, Manfred Nagel, Andy Schürr: Fine-grained and Structure-Oriented Document Integration Tools are Needed for Development Processes

1998-03 Stefan Gruner: Einige Anmerkungen zur graphgrammatischen Spezifikation von Integrationswerkzeugen nach Westfechtel, Janning, Lefering und Schürr

1998-04 * O. Kubitz: Mobile Robots in Dynamic Environments

1998-05 Martin Leucker, Stephan Tobies: Truth - A Verification Platform for Distributed Systems

1998-06 * Matthias Oliver Berger: DECT in the Factory of the Future

1998-07 M. Arnold, M. Erdmann, M. Glinz, P. Haumer, R. Knoll, B. Paech, K. Pohl, J. Ryser, R. Studer, K. Weidenhaupt: Survey on the Scenario Use in Twelve Selected Industrial Projects

1998-09 * Th. Lehmann: Geometrische Ausrichtung medizinischer Bilder am Beispiel intraoraler Radiographien

1998-10 * M. Nicola, M. Jarke: Performance Modeling of Distributed and Replicated Databases

1998-11 * Ansgar Schleicher, Bernhard Westfechtel, Dirk Jäger: Modeling Dynamic Software Processes in UML

1998-12 * W. Appelt, M. Jarke: Interoperable Tools for Cooperation Support using the World Wide Web

1998-13 Klaus Indermark: Semantik rekursiver Funktionsdefinitionen mit Striktheitsinformation

1999-01 * Jahresbericht 1998

1999-02 * F. Huch: Verifcation of Erlang Programs using Abstract Interpretation and Model Checking — Extended Version

1999-03 * R. Gallersdörfer, M. Jarke, M. Nicola: The ADR Replication Manager

1999-04 María Alpuente, Michael Hanus, Salvador Lucas, Germán Vidal: Specialization of Functional Logic Programs Based on Needed Narrowing

1999-05 * W. Thomas (Ed.): DLT 99 - Developments in Language Theory Fourth International Conference

1999-06 * Kai Jakobs, Klaus-Dieter Kleefeld: Informationssysteme für die angewandte historische Geographie

1999-07 Thomas Wilke: CTL+ is exponentially more succinct than CTL

1999-08 Oliver Matz: Dot-Depth and Monadic Quantifier Alternation over Pictures

2000-01 * Jahresbericht 1999

2000-02 Jens Vöge, Marcin Jurdzinski A Discrete Strategy Improvement Algorithm for Solving Parity Games

2000-03 D. Jäger, A. Schleicher, B. Westfechtel: UPGRADE: A Framework for Building Graph-Based Software Engineering Tools

2000-04 Andreas Becks, Stefan Sklorz, Matthias Jarke: Exploring the Semantic Structure of Technical Document Collections: A Cooperative Systems Approach

2000-05 Mareike Schoop: Cooperative Document Management

2000-06 Mareike Schoop, Christoph Quix (eds.): Proceedings of the Fifth International Workshop on the Language-Action Perspective on Communication Modelling

2000-07 * Markus Mohnen, Pieter Koopman (Eds.): Proceedings of the 12th International Workshop of Functional Languages

2000-08 Thomas Arts, Thomas Noll: Verifying Generic Erlang Client-Server Implementations

2001-01 * Jahresbericht 2000

2001-02 Benedikt Bollig, Martin Leucker: Deciding LTL over Mazurkiewicz Traces

2001-03 Thierry Cachat: The power of one-letter rational languages

2001-04	Benedikt Bollig, Martin Leucker, Michael Weber: Local Parallel Model Checking for the Alternation Free mu-Calculus
2001-05	Benedikt Bollig, Martin Leucker, Thomas Noll: Regular MSC Languages
2001-06	Achim Blumensath: Prefix-Recognisable Graphs and Monadic Second-Order Logic
2001-07	Martin Grohe, Stefan Wöhrle: An Existential Locality Theorem
2001-08	Mareike Schoop, James Taylor (eds.): Proceedings of the Sixth International Workshop on the Language-Action Perspective on Communication Modelling
2001-09	Thomas Arts, Jürgen Giesl: A collection of examples for termination of term rewriting using dependency pairs
2001-10	Achim Blumensath: Axiomatising Tree-interpretable Structures
2001-11	Klaus Indermark, Thomas Noll (eds.): Kolloquium Programmiersprachen und Grundlagen der Programmierung
2002-01 *	Jahresbericht 2001
2002-02	Jürgen Giesl, Aart Middeldorp: Transformation Techniques for Context-Sensitive Rewrite Systems
2002-03	Benedikt Bollig, Martin Leucker, Thomas Noll: Generalised Regular MSC Languages
2002-04	Jürgen Giesl, Aart Middeldorp: Innermost Termination of Context-Sensitive Rewriting
2002-05	Horst Lichter, Thomas von der Maßen, Thomas Weiler: Modelling Requirements and Architectures for Software Product Lines
2002-06	Henry N. Adorna: 3-Party Message Complexity is Better than 2-Party Ones for Proving Lower Bounds on the Size of Minimal Nondeterministic Finite Automata
2002-07	Jörg Dahmen: Invariant Image Object Recognition using Gaussian Mixture Densities
2002-08	Markus Mohnen: An Open Framework for Data-Flow Analysis in Java
2002-09	Markus Mohnen: Interfaces with Default Implementations in Java
2002-10	Martin Leucker: Logics for Mazurkiewicz traces
2002-11	Jürgen Giesl, Hans Zantema: Liveness in Rewriting
2003-01 *	Jahresbericht 2002
2003-02	Jürgen Giesl, René Thiemann: Size-Change Termination for Term Rewriting
2003-03	Jürgen Giesl, Deepak Kapur: Deciding Inductive Validity of Equations
2003-04	Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Improving Dependency Pairs
2003-05	Christof Löding, Philipp Rohde: Solving the Sabotage Game is PSPACE-hard
2003-06	Franz Josef Och: Statistical Machine Translation: From Single-Word Models to Alignment Templates
2003-07	Horst Lichter, Thomas von der Maßen, Alexander Nyßen, Thomas Weiler: Vergleich von Ansätzen zur Feature Modellierung bei der Softwareproduktlinienentwicklung
2003-08	Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Mechanizing Dependency Pairs
2004-01 *	Fachgruppe Informatik: Jahresbericht 2003

2004-02 Benedikt Bollig, Martin Leucker: Message-Passing Automata are expressively equivalent to EMSO logic
2004-03 Delia Kesner, Femke van Raamsdonk, Joe Wells (eds.): HOR 2004 – 2nd International Workshop on Higher-Order Rewriting
2004-04 Slim Abdennadher, Christophe Ringeissen (eds.): RULE 04 – Fifth International Workshop on Rule-Based Programming
2004-05 Herbert Kuchen (ed.): WFLP 04 – 13th International Workshop on Functional and (Constraint) Logic Programming
2004-06 Sergio Antoy, Yoshihito Toyama (eds.): WRS 04 – 4th International Workshop on Reduction Strategies in Rewriting and Programming
2004-07 Michael Codish, Aart Middeldorp (eds.): WST 04 – 7th International Workshop on Termination
2004-08 Klaus Indermark, Thomas Noll: Algebraic Correctness Proofs for Compiling Recursive Function Definitions with Strictness Information
2004-09 Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Parameterized Power Domination Complexity
2004-10 Zinaida Benenson, Felix C. Gärtner, Dogan Kesdogan: Secure Multi-Party Computation with Security Modules
2005-01 * Fachgruppe Informatik: Jahresbericht 2004
2005-02 Maximillian Dornseif, Felix C. Gärtner, Thorsten Holz, Martin Mink: An Offensive Approach to Teaching Information Security: "Aachen Summer School Applied IT Security"
2005-03 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp: Proving and Disproving Termination of Higher-Order Functions
2005-04 Daniel Mölle, Stefan Richter, Peter Rossmanith: A Faster Algorithm for the Steiner Tree Problem
2005-05 Fabien Pouget, Thorsten Holz: A Pointillist Approach for Comparing Honeypots
2005-06 Simon Fischer, Berthold Vöcking: Adaptive Routing with Stale Information
2005-07 Felix C. Freiling, Thorsten Holz, Georg Wicherski: Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks
2005-08 Joachim Kneis, Peter Rossmanith: A New Satisfiability Algorithm With Applications To Max-Cut
2005-09 Klaus Kursawe, Felix C. Freiling: Byzantine Fault Tolerance on General Hybrid Adversary Structures
2005-10 Benedikt Bollig: Automata and Logics for Message Sequence Charts
2005-11 Simon Fischer, Berthold Vöcking: A Counterexample to the Fully Mixed Nash Equilibrium Conjecture
2005-12 Neeraj Mittal, Felix Freiling, S. Venkatesan, Lucia Draque Penso: Efficient Reductions for Wait-Free Termination Detection in Faulty Distributed Systems
2005-13 Carole Delporte-Gallet, Hugues Fauconnier, Felix C. Freiling: Revisiting Failure Detection and Consensus in Omission Failure Environments
2005-14 Felix C. Freiling, Sukumar Ghosh: Code Stabilization
2005-15 Uwe Naumann: The Complexity of Derivative Computation

2005-16 Uwe Naumann: Syntax-Directed Derivative Code (Part I: Tangent-Linear Code)
2005-17 Uwe Naumann: Syntax-directed Derivative Code (Part II: Intraprocedural Adjoint Code)
2005-18 Thomas von der Maßen, Klaus Müller, John MacGregor, Eva Geisberger, Jörg Dörr, Frank Houdek, Harbhajan Singh, Holger Wußmann, Hans-Veit Bacher, Barbara Paech: Einsatz von Features im Software-Entwicklungsprozess - Abschlußbericht des GI-Arbeitskreises “Features”
2005-19 Uwe Naumann, Andre Vehreschild: Tangent-Linear Code by Augmented LL-Parsers
2005-20 Felix C. Freiling, Martin Mink: Bericht über den Workshop zur Ausbildung im Bereich IT-Sicherheit Hochschulausbildung, berufliche Weiterbildung, Zertifizierung von Ausbildungsangeboten am 11. und 12. August 2005 in Köln organisiert von RWTH Aachen in Kooperation mit BITKOM, BSI, DLR und Gesellschaft fuer Informatik (GI) e.V.
2005-21 Thomas Noll, Stefan Rieger: Optimization of Straight-Line Code Revisited
2005-22 Felix Freiling, Maurice Herlihy, Lucia Draque Penso: Optimal Randomized Fair Exchange with Secret Shared Coins
2005-23 Heiner Ackermann, Alantha Newman, Heiko Röglin, Berthold Vöcking: Decision Making Based on Approximate and Smoothed Pareto Curves
2005-24 Alexander Becher, Zinaida Benenson, Maximillian Dornseif: Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks
2006-01 * Fachgruppe Informatik: Jahresbericht 2005
2006-02 Michael Weber: Parallel Algorithms for Verification of Large Systems
2006-03 Michael Maier, Uwe Naumann: Intraprocedural Adjoint Code Generated by the Differentiation-Enabled NAGWare Fortran Compiler
2006-04 Ebadollah Varnik, Uwe Naumann, Andrew Lyons: Toward Low Static Memory Jacobian Accumulation
2006-05 Uwe Naumann, Jean Utke, Patrick Heimbach, Chris Hill, Derya Ozyurt, Carl Wunsch, Mike Fagan, Nathan Tallent, Michelle Strout: Adjoint Code by Source Transformation with OpenAD/F
2006-06 Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Divide-and-Color
2006-07 Thomas Colcombet, Christof Löding: Transforming structures by set interpretations
2006-08 Uwe Naumann, Yuxiao Hu: Optimal Vertex Elimination in Single-Expression-Use Graphs
2006-09 Tingting Han, Joost-Pieter Katoen: Counterexamples in Probabilistic Model Checking
2006-10 Mesut Günes, Alexander Zimmermann, Martin Wenig, Jan Ritzerfeld, Ulrich Meis: From Simulations to Testbeds - Architecture of the Hybrid MCG-Mesh Testbed
2006-11 Bastian Schlich, Michael Rohrbach, Michael Weber, Stefan Kowalewski: Model Checking Software for Microcontrollers
2006-12 Benedikt Bollig, Joost-Pieter Katoen, Carsten Kern, Martin Leucker: Replaying Play in and Play out: Synthesis of Design Models from Scenarios by Learning

2006-13 Wong Karianto, Christof Löding: Unranked Tree Automata with Sibling Equalities and Disequalities

2006-14 Danilo Beuche, Andreas Birk, Heinrich Dreier, Andreas Fleischmann, Heidi Galle, Gerald Heller, Dirk Janzen, Isabel John, Ramin Tavakoli Kolagari, Thomas von der Maßen, Andreas Wolfram: Report of the GI Work Group "Requirements Management Tools for Product Line Engineering"

2006-15 Sebastian Ullrich, Jakob T. Valvoda, Torsten Kuhlen: Utilizing optical sensors from mice for new input devices

2006-16 Rafael Ballagas, Jan Borchers: Selexels: a Conceptual Framework for Pointing Devices with Low Expressiveness

2006-17 Eric Lee, Henning Kiel, Jan Borchers: Scrolling Through Time: Improving Interfaces for Searching and Navigating Continuous Audio Timelines

2007-01 * Fachgruppe Informatik: Jahresbericht 2006

2007-02 Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, and Harald Zankl: SAT Solving for Termination Analysis with Polynomial Interpretations

2007-03 Jürgen Giesl, René Thiemann, Stephan Swiderski, and Peter Schneider-Kamp: Proving Termination by Bounded Increase

2007-04 Jan Buchholz, Eric Lee, Jonathan Klein, Jan Borchers: coJIVE: A System to Support Collaborative Jazz Improvisation

2007-05 Uwe Naumann: On Optimal DAG Reversal

2007-06 Joost-Pieter Katoen, Thomas Noll, and Stefan Rieger: Verifying Concurrent List-Manipulating Programs by LTL Model Checking

2007-07 Alexander Nyßen, Horst Lichter: MeDUSA - MethoD for UML2-based Design of Embedded Software Applications

* These reports are only available as a printed version.
Please contact biblio@informatik.rwth-aachen.de to obtain copies.