# Annual Report of the Chair of Computer Science 2 (Programming Systems)

## 1 Staff

Dipl.-Ing. (FH) Helmut Allendorf (IT-support), Michael Baer, M. Sc., Dipl.-Inf. Thorsten Blaß, Tobias Feigl, M. Sc., Hon.-Prof. Dr.-Ing. Bernd Hindel, Florian Jung, M. Sc., Marius Kamp, M. Sc., Hon.-Prof. Dr.-Ing. Detlef Kips, Patrick Kreutzer, M. Sc., Florian Mayer, M. Sc., Dipl.-Inf. Daniela Novac, Dr.-Ing. Norbert Oster, Akad. ORat, Prof. Dr. Michael Philippsen (Ordinarius), Prof. em. Dr. Hans Jürgen Schneider (Emeritus), Manfred Uebler (IT-support), Margit Zenk (Secretary)

Guests and external teaching staff: Dr.-Ing. Josef Adersberger, Veronika Dashuber, M. Sc., Dr.-Ing. Martin Jung (external), Dipl.-Math. Jakob Krainz, Florian Lautenschlager, M. Sc., Dr.-Ing. Christopher Mutschler, Dr.-Ing. Klaudia Dussa-Zieger (external)

## 2 Overview

Since 2002 Prof. Michael Philippsen is heading the Chair of Computer Science 2 which was founded in 1972. The group's main focus is on **Programming Systems for heterogeneous Multicores**.

Up to 2014/2015, we mainly took a system-level perspective and worked on the interface to both the parallel hardware and the OS, on runtime system issues, and on how to best express parallelism in program code. We strived to best harvest the potential of parallelism that is slumbering within applications and to find the most efficient ways to map it to the parallelism provided by the hardware. As such systems-oriented topics continue to be relevant, some of our projects continue to address them.

Nowadays the Programming Systems Group mainly finds answers for professional software engineers who develop industry-sized parallel programs for multicores, for distributed networks of multicores, for parallel cloud computing, and for networks of embedded systems.

## 3 Research projects

**AnaCoRe** – *Analysis of Code Repositories*: Software developers often modify their projects in a similar or repetitive way. The reasons for these changes include the adoption of a changed interface to a library, the correction of mistakes in functionally similar components, or the parallelization of sequential parts of a program. If developers have to perform the necessary changes on their own, the modifications can easily introduce errors, for example due to a missed change location. Therefore, an automatic technique is desireable that identifies similar changes and uses this knowledge to support developers with further modifications.
SYFEX is a new approach to symbolic code execution that computes the semantic similarity of two code fragments. In 2017 and 2018 we optimized the implementation of SYFEX. We also began collecting a data set of semantically similar methods from open source repositories. In order to make it feasible to search for semantically similar code in bigger code projects, we developed a new technique based in genetic programming, to hierarchically combine base comparators. We further improved the implementation of this approach in 2017 and 2018. Additionally, we focused on evaluating the approach with pairs

of methods from software repositories and from programming exercises.

**AuDeRace** – *Automatic Detection of Race-Conditions*: Recent software contains more and more parallelism. This introduces several new bug patterns, like deadlocks and concurrent memory accesses, that are harder or even impossible to be detected reliably using conventional test methods. Whether the faulty behavior arises depends on the concrete scheduling of the threads which is indeterministic and varies between individual executions depending on the underlaying system. Due to this unpredictable behavior such bugs do not necessarily manifest in an arbitrary test run or may never arise in the testing environment at all.
In 2018 the focus moved to a deterministic execution of test cases. A concept to reproduce results during the execution was developed: In addition to the test case, a schedule specifies the dynamic behaviour of the threads. Instrumenting the code at previously marked positions and other relevant byte code instructions allows a separate control thread to enforce the schedule. When modifying the source code, the marked positions in the code need to be updated as well to keep them consistent with the test cases. A merging technique similar to the ones used in version control systems shall be used to automatically update the positions.
This project is a contribution of the Chair of Computer Science 2 (Programming Systems) to the IZ ESI (Embedded Systems Initiative, http://www.esi.fau.de/ ) dar.

**AutoCompTest** – *Automatic Testing of Compilers*: Compilers for programming languages are very complex applications and their correctness is crucial: If a compiler is erroneous (i.e., if its behavior deviates from that defined by the language specification), it may generate wrong code or crash with an error message. Often, such errors are hard to detect or circumvent. The lack of an appropriate test program generator and the high costs associated with the development of such a tool often prevent the automatic testing of compilers in practice.
In 2018, we started the development of such a tool. As input, it requires a specification of a programming language's syntactic and semantic rules by means of an abstract attribute grammar. Such a grammar allows for a short notation of the rules on a high level of abstraction. Our newly devised algorithm then generates test programs that conform to all of the specified rules. It uses several novel technical ideas to reduce its expected runtime. This way, it can generate large sets of test programs in acceptable time, even when executed on a standard desktop computer. A first evaluation of our approach did not only show that it is efficient and effective, but also that it is versatile. Our approach detected several bugs in the C compilers gcc and clang (and achieved a bug detection rate which is comparable to that of a state-of-the-art C program generator from the literature) as well as multiple bugs in different SMT solvers. Some of the bugs that we detected were previously unknown to the respective developers.

**DfD** – *Design for Diagnosability*: Many software systems behave obtrusively during the test phase or even in normal operation. The diagnosis and the therapy of such runtime anomalies is often time consuming and complex, up to being impossible. „Design for Diagnosability“ is a tool chain that consists of modeling languages, components, and tools targeted towards increasing the diagnosability of software systems.
We continued to make further contributions to the research project in 2018: We have published a paper at PROFES 2018 that describes techniques and insights on how runtime data in a large software project can be offered to all project participants at the development stage to improve their collaboration. We have maintained the Chronix Open Source project and stabilized it further (updating versions, fixing bugs, etc.).

**GIFzuMINTS** – *Computer Science Basics as a basis for future-oriented MINT studies*: Basic courses of computer science taught during the first terms of the university studies are crucial for a successful degree. Unfortunately they have also proven as a problematic obstacle that often lead to a premature termination of the trial. In order to counteract the situation, we further extended the support of prospective students

during the transition phase from school to university and during the first terms of their studies.
In 2018 we analyzed the impact of all the different measures we developed throughout the project. We quantified the outcome of increasing the number of tutorial courses and deepening the support of students by applying qualified tutors. Additionally, we also assessed the impact of attending the revision course on the results of the homework submissions and the final examination.

**Holoware** – *Cooperative Exploration and Analysis of Software in a Virtual/Augmented Reality Appliance*: Understanding software has a large share in the programming efforts of a software systems, up to 30% in development projects and up to 80% in maintenance projects. Therefore, an efficient and effective way for comprehending software is neccessary in a modern software engineering workplace. Three-dimensional software visualization already boosts comprehension and efficiency, so utilization of latest virtual reality techniques seems natural.
Since the project started in September 2018, the following significant contributions have already been made. First we developed of a functional VR visualization prototype for demonstration and research purposes. We defined a mapping between dynamic run time data and static structure as a base for later analysis and visualization tasks. Finally we developed a first draft and implementation of the trace anomaly detection by an unsupervised learning procedure.

**ICPC** – *International Collegiate Programming Contest an der FAU*: Since 1977 the International Collegiate Programming Contest (ICPC) takes place every year. Teams of three students try to solve about 13 programming problems within five hours. What makes this task even harder, is that there is only one computer available per team.
In 2018 we conducted two local contests in Erlangen. In the winter semester there was a team contest with teams consisting of at most three students. The main goal of this contest was to interest new students in the contests. We had 30 FAU teams plus 57 more teams from universities all over Germany. Before the second contest, as in the previous years, in the summer term the seminar *Hello World - Programming for the Advanced* served to prepare students from different disciplines in algorithms and contest problems. In the German wide contest of the summer term we selected the students that would represent the FAU at the NWERC 2018 in Eindhoven (NL). 30 teams with students of computer science, computational engineering, mathematics as well as informations and communication technology took the challenge. We formed the NWERC teams out of the best participants in the qualifications, given the age restrictions. At the NWERC in Eindhoven, our teams reached places 53, 57 and 63 out of the 119 teams participating.

**ORKA-HPC** – *OpenMP for reconfigurable heterogenous architectures*: High-Performance Computing (HPC) is an important component of Europe's capacity for innovation and it is also seen as a building block of the digitization of the European industry. Reconfigurable technologies such as Field Programmable Gate Array (FPGA) modules are gaining in importance due to their energy efficiency, performance, and flexibility.
In 2018 we developed of a source-to-source compiler prototype for the rewriting of OpenMP C source code. Additionally an HLS compiler prototype capable of translating C code into hardware has been developed as well. Finally we designed and implemented several experimental FPGA infrastructures for the execution of accelerator cores.

**ParCAn** – *Parallel code analysis on a GPU*: In compiler construction there are analyses that propagate information along the edges of a graph and modify it, until a fix point is reached and the information no longer changes. In this project we build the ParCAn framework to accelerate such analyses by exploiting the massive parallelism of graphic cards.
In 2018 we completed our comparative study on the efficiency of graph data structures on GPUs. To show the effectiveness of our framework we integrated it into the LLVM compiler framework. We picked four LLVM analyses and parallelized them with ParCAn. Ample measurements show that our framework can

accelerate LLVM's compilation process by up to 40%. A publication was accepted at the *28th International Conference on Compiler Construction* and will receive the Best-Paper-Award.

**RuNN** – *Recurrent Neuronal Networks (RNNs) for Real-Time Estimation of Nonlinear Motion Models*: With the growing availability of information about an environment (e.g., the geometry of a gymnasium) and about the objects therein (e.g., athletes in the gymnasium), there is an increasing interest in bringing that information together profitably (so-called information fusion) and in processing that information. For example, one would like to reconstruct physically correct animations (e.g., in virtual reality, VR) of complex and highly dynamic movements (e.g., in sports situations) in real-time. The core objective of the project is therefore to evaluate how machine learning methods can be used to describe complex and nonlinear movements. The aim is to investigate whether RNNs physically describe the movements of an object correctly and how existing methods can be supported or replaced.
In 2018, a deeper understanding of the initial situation and problem definition was first established. Methods of machine and deep learning for motion detection and motion reconstruction based on inertial, camera, and radio sensors were studied as well as various methods for feature extraction. The findings were used to stabilize so-called relative Pedestrian Dead Reckoning (PDR) methods using motion classifiers. The deeper radio signal understanding allowed the mapping of long-term errors in RNN-based motion models to improve position accuracy, stability, and to predict near real-time. Additionally, a large-scale social science study opened the world's largest virtual dinosaur museum and showed that a pre-selected (application-optimized) model of human movement robustly and accurately (meaning no significant impact on simulator sickness) maps human motion, resp. predicts it.

**SoftWater** – *Software Watermarking*: Software watermarking means hiding selected features in code, in order to identify it or prove its authenticity. This is useful for fighting software piracy, but also for checking the correct distribution of open-source software. The goal of our research is the development of a watermarking framework that automates this process by introducing the watermark while compiling the code.
In 2018 two methods on symbolic execution and function synthesis were analysed within the bounds of two bachelor theses in order to determine the most appropriate one for our approach.

## 4 Teaching

The Chair for Programming Systems teaches the compulsory modules *Algorithms and Data Structures (AuD)* during the winter term and *Parallel and Functional Programming (PFP)* during the summer term. Since those modules are offered to many other degree programs from different faculties (especially Business & Information Systems resp. International Information Systems, Information and Communication Technology, Mathematics and many more), the number of attending students and examinations reached a new high score since the start of the modules: 773 students attended AuD during the winter term 2017/18 and 382 students attended PFP during the summer term 2018 – the number of examinations hit 712 in AuD resp. 329 in PFP. The Chair offers different modules on *Compiler Construction*, *Clustercomputing* and *Testing of Software Systems* to students specializing on programming systems. The tutorials *Hallo Welt! für Fortgeschrittene* and *Machine Learning* were also fully booked within short time.
The Chair for Programming Systems supervised five master's thesis, three master projects and three bachelor's thesis in total during the period under report.

## 5 Publications 2018

[1] Lugrin JL., Kern F., Schmidt R., Kleinbeck C., Roth D., Daxer C., Feigl T., Mutschler C., Latoschik ME.: A Location-Based VR Museum. In *10th International Conference on Virtual Worlds and Games for Serious Applications (VS Games 2018)*. 2018. doi:10.1109/VS-Games.2018.8493404

[2] Roth D., Kleinbeck C., Feigl T., Mutschler C., Latoschik ME.: Beyond Replication: Augmenting Social Behaviors in Multi-User Social Virtual Realities In *25th IEEE Conference on Virtual Reality and 3D User Interfaces (IEEE VR 2018)* 2018. doi:10.1109/VR.2018.8447550

[3] Feigl T., Mutschler C., Philippsen M.: Head-to-Body-Pose Classification in No-Pose VR Tracking Systems. In *25th IEEE Conference on Virtual Reality and 3D User Interfaces (IEEE VR 2018)*. 2018. doi:10.1109/VR.2018.8446495

[4] Feigl T., Mutschler C., Philippsen M.: Human Compensation Strategies for Orientation Drifts. In *25th IEEE Conference on Virtual Reality and 3D User Interfaces*. 2018. doi:10.1109/VR.2018.8446300

[5] Lautenschlager F., Ciolkowski M.: Making Runtime Data Useful for Incident Diagnosis: An Experience Report. In *International Conference on Product-Focused Software Process Improvement (PROFES 2018)*. 2018. doi:10.1007/978-3-030-03673-7_33

[6] Gorse L., Löffler C., Mutschler C., Philippsen M.: Optical Camera Communication for Active Marker Identification in Camera-based Positioning Systems. In *15th Workshop on Positioning, Navigation and Communications (WPNC'18)*. 2018. doi:10.1109/WPNC.2018.8555846

[7] Feigl T., Nowak T., Philippsen M., Edelhäußer T., Mutschler C.: Recurrent Neural Networks on Drifting Time-of-Flight Measurements. In *9th International Conference on Indoor Positioning and Indoor Navigation (IPIN 2018)*. 2018. doi:10.1109/IPIN.2018.8533813

[8] Feigl T., Mutschler C., Philippsen M.: Supervised Learning for Yaw Orientation Estimation. In *Proceedings of the 9th International Conference on Indoor Positioning and Indoor Navigation (IPIN 2018)*. 2018. doi:10.1109/IPIN.2018.8533811