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

1 Staff

Dipl.-Ing. (FH) Helmut Allendorf (IT-support, until March 31, 2019), 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), Bastian Söllmann (IT-support, since April 01, 2019), Manfred Uebler (IT-support, until March 31, 2019), Margit Zenk (Secretary).

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

2 Overview

The programming systems group develops scientific solutions for software engineers in industry who work on **parallel software** for multicores and for distributed or embedded systems made thereof. The team takes a **code-centric approach**, constructs operational **prototypes**, and **evaluates** them both quantitatively and qualitatively.
Corner stones of our field of research:
(a) The chair works on **programming models** for **heterogeneous** parallelism, from which it then generates portable and efficient code for multicores, GPUs, accelerators, mobile devices, FPGAs, etc.
(b) The working group helps parallelize software for multicores. The tools developed therefor analyze code repositories and help developers in **migrating** and **refactoring** projects.
(c) The team analyzes code and implements **code analysis tools** that are fast, interactive, incremental and sometimes work in parallel themselves. They not only detect race conditions, conflicting accesses to resources, etc. The resulting suggestions on how to improve the code also show up in the IDE where they matter.
(d) The programming systems group **tests** parallel code and **diagnoses** the root causes of problems. The tools developed by the team for this purpose generate test data, track down erratic runtime behavior, and prevent **authenticity attacks**.

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 2019, we collected and published a data set of semantically similar methods from open source repositories.

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 actually shows at runtime 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.
Up to 2019, this project was a contribution of the Chair of Computer Science 2 to the IZ ESI (Embedded Systems Initiative, http://www.esi.fau.de/). In this context, several improvements for the quality of concurrent software were analyzed. The take-away result was that different approaches are applicable and required, but they also often suffer from long analysis times. Beyond the ESI project, we improved the usefulness of mutation testing by developing a tool for equivalence detection and test case generation. A submitted paper got accepted.

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 2019, we implemented additional features for the definition of language specifications and improved the efficiency of our program generator. These two contributions considerably increased the throughput of our tool. By developing additional language specifications, we were also able to uncover bugs in compilers for the programming languages Lua and SQL. The results of our work led to a publication that we submitted at the end of 2019 (and which has been accepted by now). Besides the work on our program generator, we also began working on a test case reduction technique. It reduces the size of a randomly generated test program that triggers a compiler bug since this eases the search for the bug's root cause.

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.
In 2019, we extended the prototype initiated during the previous year to display dynamic software behaviour and enabled cooperative (remote-)usability. Additionally, we implemented the interpretation of commit messages for anomaly detection and display system calls clustered according to use cases.
Our paper "Towards Collaborative and Dynamic Software Visualization in VR" has been accepted for publication at the International Conference on Computer Graphics Theory and Applications (VISIGRAPP) 2020. It presents the efficiency of our prototype at increasing the software understanding process.

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 2019, the following significant contributions were achieved: We improved the source-to-source compiler in order to properly support OpenMP-target-outlining for FPGA targets (incl. smoke tests). We completed the first working ORKA-HPC prototype supporting a complete OpenMP-to-FPGA flow. We formulated a genome for the pragma-based genetic optimization of the high-level synthesis step during

the ORKA-HPC flow. We extended the TaPaSCo composer to allow for hardware synchronization primitives inside of TaPaSCo systems.

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 2019, ParCAn was adjusted to the new execution model of NVIDIA's latest GPU architectures. With the introduction of the Volta architecture, threads can now achieve progress independently of the others. Since Volta every thread has its own program counter and call stack. Previously, a group of threads (called a warp) shared both a common program counter as well as a call stack. The threads either executed the same instruction or were idle (lock-step execution). Applications that are not adjusted to this execution model will compute wrong results. As threads now execute independently of each other, race conditions can occur within warps. Older lock-step fashioned execution models inserted synchronization points to prevent this implicitly. We inspected ParCAn's source code for code fragments susceptible to causing race conditions on new architectures. These fragments were adjusted to now execute properly on the latest NVIDIA architectures.

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 2019, we found that these models can also predict human movement (human movement model). We also determined that the LSTM models can either be fully self-sufficient at runtime or integrated as support points into localization estimates, e.g., into Pedestrian Dead Reckoning (PDR) methods. Additionally, we showed that models of the RNN family extrapolate movements into the future so that they compensate for the latency of the processing pipeline and beyond. Furthermore, we examined the explainability, interpretability, and robustness of the models examined here, and their reusability on the human movement. With the help of a simulator, we generated physically correct movements, e.g., positions of pedestrians, cyclists, cars, and planes. Based on this data, we showed that RNN models can interpolate between different types of movement and can compensate for missing data points, interpret white and random noise as such, and can extrapolate movements. The latter enables processing-specific latency to be compensated and enables human movement to be predicted from radio and inertial data in real time.

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 2019, we investigated the idea to use concolic execution in the context of the LLVM compiler infrastructure in order to hide a watermark in an unused register. Using a modified register allocation, one register can be reserved for storing the watermark.

GIFzuMINTS – *Computer Science Basics as an Essential Building Block of Modern STEM Field Curricula*: 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 studies. 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.
The GIFzuMINTS project ended in 2019 with a special highlight: On May 20, 2019, the Bavarian Minister of State for Science and Art, Bernd Sibler, and the deputy general manager of vbw bayme vbm, Dr. Christof Prechtl, visited us in a status meeting. Minister Bernd Sibler was impressed by the state and the results of the project.
By the end of the project, the measures developed and implemented were thoroughly evaluated and established as permanent offers. The revision course on computer science was transformed into a continuous virtual offer for self-study. The course for talented students that prepares them to participate in international programming competitions was expanded and set up as a formal module of the curriculum.

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 almost reached the high scores of the previous terms: 721 students attended AuD during the winter term 2018/19 and 320 students attended PFP during the summer term 2019 – the number of examinations hit 668 in AuD resp. 302 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 three master's thesis, a master project and seven bachelor's thesis in total during the period under report.

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 2019, 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. 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 represented the FAU at the NWERC 2019 in Eindhoven (NL). 27 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 16, 41, and 52 out of the 123 teams participating.

5 Awards and Prizes

The following publication received the *Best Paper Award*:
Thorsten Blaß, Michael Philippsen: GPU-Accelerated Fixpoint Algorithms for Faster Compiler Analyses In *Proceedings of the 28th International Conference on Compiler Construction* 2019. doi:10.1145/3302516.3307352

6 Publications 2019

[1] Feigl T., Kram S., Woller P., Siddiqui R.H., Philippsen M., Mutschler C.: A Bidirectional LSTM for Estimating Dynamic Human Velocities from a Single IMU. In *Proceedings of the 10th International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2019. doi:10.1109/IPIN.2019.8911814

[2] Blaß T., Philippsen M.: GPU-Accelerated Fixpoint Algorithms for Faster Compiler Analyses In *Proceedings of the 28th International Conference on Compiler Construction* 2019. doi:10.1145/3302516.3307352

[3] Mayer F., Knaust M., Philippsen M.: OpenMP on FPGAs - A Survey In *OpenMP: Conquering the Full Hardware Spectrum - Proceedings of the 15th International Workshop on OpenMP (IWOMP 2019)* 2019. doi:10.1007/978-3-030-28596-8_7

[4] Kamp M., Kreutzer P., Philippsen M.: SeSaMe: A Data Set of Semantically Similar Java Methods In *Proceedings of the 16th International Conference on Mining Software Repositories (MSR 2019)* 2019. doi:10.1109/MSR.2019.00079

[5] Feigl T., Roth D., Gradl S., Wirth M., Latoschik M.E., Eskofier B., Philippsen M., Mutschler C.: Sick Moves! Motion Parameters as Indicators of Simulator Sickness In *IEEE Transactions on Visualization and Computer Graphics* 2019. doi:10.1109/TVCG.2019.2932224

[6] Blaß T., Philippsen M.: Which Graph Representation to Select for Static Graph-Algorithms on a CUDA-capable GPU In *Proceedings of the 12th Workshop on General Purpose Processing Using GPUs* 2019. doi:10.1145/3300053.3319416