Open-Source Software as Catalyzer for Technology Transfer: Kieker's Development and Lessons Learned

Wilhelm Hasselbring[1] and André van Hoorn[2]

[1] Kiel University, Department of Computer Science, 24118 Kiel, Germany
[2] University of Stuttgart, Institute of Software Technology, 70569 Stuttgart, Germany

Abstract: The monitoring framework Kieker commenced as a joint diploma thesis of the University of Oldenburg and a telecommunication provider in 2006, and grew toward a high-quality open-source project during the last years. Meanwhile, Kieker has been and is employed in various projects. Several research groups constitute the open-source community to advance the Kieker framework. In this paper, we review Kieker's history, development, and impact as catalyzer for technology transfer.

1 Introduction

The development of tools is common practice for researchers in order to demonstrate the practicality of developed research approaches and to qualitatively and quantitatively evaluate their research results. During the last years, there is an increasing trend that researchers make their tools publicly available under an open-source license, e.g., allowing a more thorough evaluation of work presented in research papers, as well as easing reproducibility of results and building on the work of others. The state of these tools ranges from proof-of-concept implementations to full-blown products. Popular examples of wide-spread and mature open-source tools originally developed and maintained by researchers include the probabilistic model checker PRISM [KNP11] and the R language and environment for statistical computing [R D08].

Since 2006, we have been developing the Kieker framework for dynamic analysis of software systems.[1] In this paper, we review Kieker's history, development, and impact as catalyzer for technology transfer. Parts of this paper have been published in a PhD dissertation [vH14, Chapter 15], which also includes a more detailed description of the framework (in addition to [RvHM+08, vHRH+09, vHWH12]) as well as its development process and infrastructure.

2 Kieker's Development and Impact

This section reviews the past years of Kieker development and gives some indication of the impact in terms of where and by whom Kieker has been developed and used.

[1] The Kieker framework's web site—including downloads, documentation, publications, and references—is available at `http://kieker-monitoring.net`

Accompanying paper for our talk at 1. Kieler Open Source Business Konferenz, Sept. 14, 2015, Kiel, Germany.

Appeared as: W. Hasselbring and A. van Hoorn. Open-Source Software as Catalyzer for Technology Transfer: Kieker's Development and Lessons Learned. Technical Report TR-1508, Department of Computer Science, Kiel University, Kiel, Germany. Aug. 2015.

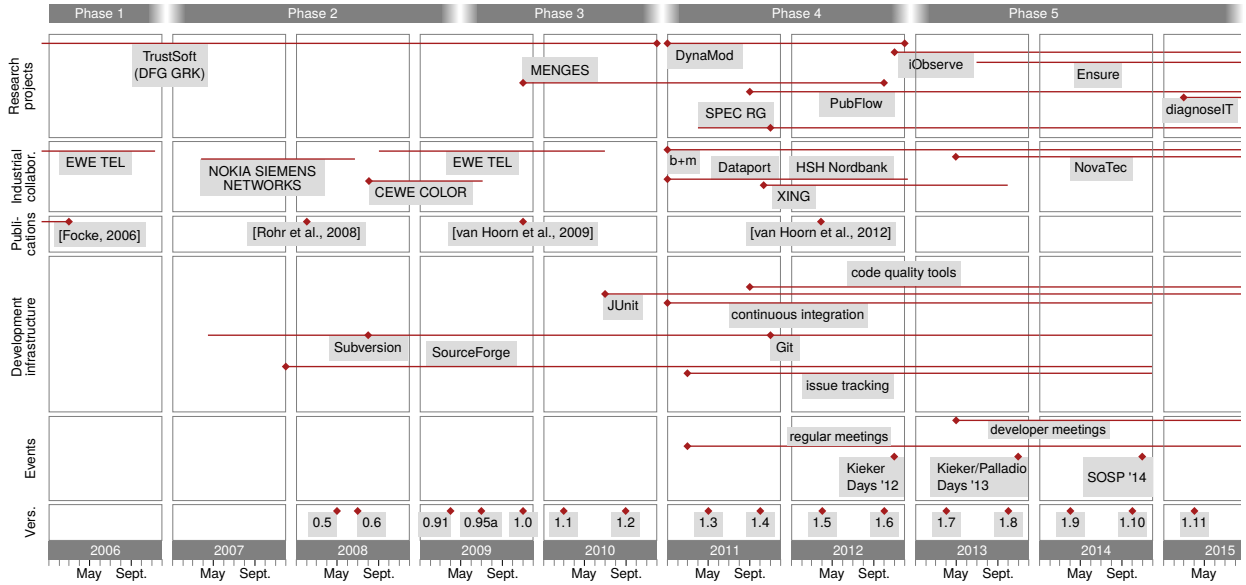

Figure 1: The timeline depicts durations of associated research projects, industrial collaborations, publications describing the framework, development tools, and released versions.

2.1 Evolution Phases

This section provides a review of Kieker's history starting from its origin in 2006 to the middle of 2015. We roughly divide the past years of evolution into five phases. The timeline in Figure 1 depicts the durations of each of these phases along with important events in the context of the Kieker project, which will be discussed in the remainder of this section.

Phase 1: 2006 (Inception)
Kieker originates from Focke's Diploma thesis [Foc06] on performance monitoring of middleware-based applications. The thesis was conducted at the University of Oldenburg (Software Engineering Group), in collaboration with the EWE TEL GmbH, Oldenburg. As part of his work, Focke developed a performance monitoring component for Java EE applications, called *Tpmon*, providing *aggregated* performance measures for Java methods: invocation counts as well as average, maximum, and minimum response times.

Phase 2: 2007–2009 (Production Systems)
Tpmon has been developed further by Matthias Rohr—who co-supervised Focke's thesis [Foc06]—for experimental evaluation as part of PhD research on timing behavior anomaly detection [RGH07, MRvHH09, EvHWH11, Roh14]. In the context of that research, van Hoorn got in touch with *Tpmon* in 2007 during the course of his Diploma thesis [vH07]. He used *Tpmon* for operation response time measurements in the experimental evaluation. At that time, *Tpmon* was tailored to measure information of operation executions and

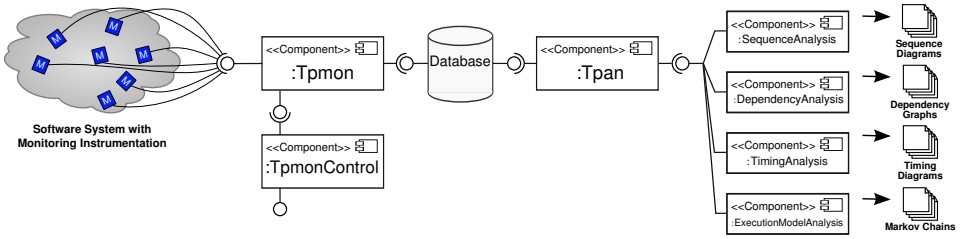

Figure 2: Overview of Kieker's architecture in 2007 [RvHM+08]

log these to either the file system or an SQL database (both supporting a synchronous and an asynchronous mode). In 2007, Kieker received its name when we prepared a first publication on the tool's architecture, and its trace extraction and visualization features [RvHM+08]. Figure 2 shows the visualization of Kieker's architecture from that publication. The analysis component including the trace reconstruction and visualization functionality was called *Tpan*.

We released first open-source versions of Kieker in 2008 (Version 0.5 in May, Version 0.6 in July). These versions included only the monitoring component *Tpmon* with the afore-mentioned variants of the file system and database writers. The total number of Java classes was 12; two AspectJ-based probes were included. As part of our collaborations with CEWE COLOR and EWE TEL (detailed below), Kieker was used for monitoring production systems.

In 2009, we included support for distributed tracing for Java systems that employ SOAP-based web service technology for remote communication (Version 0.91). This feature was a result of our collaboration with EWE TEL.

During this phase, only few documentation for Kieker existed. New users needed a lot of assistance to use the tool as a basis for their work. To our knowledge, Kieker was only used by ourselves as part of our research in the DFG Graduate School on Trustworthy Software Systems (TrustSoft) and the Software Engineering Group at the University of Oldenburg.

Phase 3: 2009–2010 (Restructuring)
In 2009, we considerably restructured Kieker towards the generalized and extensible framework architecture with records, writers, readers, and analysis plugins that it has today. The restructured architecture along with results on systematic overhead benchmarks were published in our 2009 technical report [vHRH+09]. Figure 3, showing the restructured architecture in terms of the core components and their interconnection, is taken from that report. Kieker's new architecture was released with Versions 0.95a (July 2009) and 1.0 (November 2009)—the first versions containing parts of the analysis component. That year, colleagues from Kiel University (Software Engineering Group) started to join the development.

In 2010, we added the system meta-model and the online trace reconstruction to the trace analysis tool. The documentation improved considerably by creating the user guide with examples. *Tpmon* and *Tpan* were renamed to *Kieker.Monitoring* and *Kieker.Analysis*. We

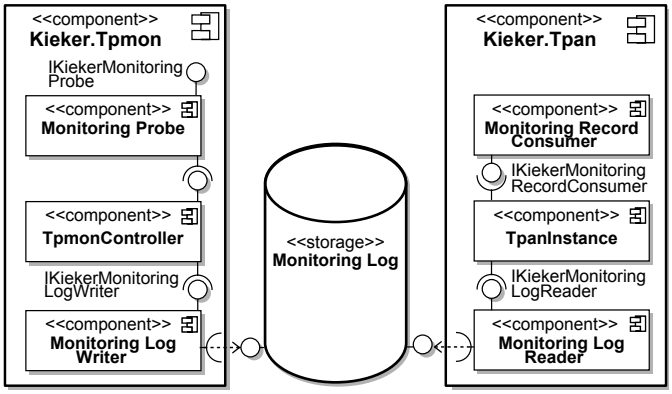

Figure 3: Overview of Kieker's restructured architecture [vHRH+09]

released the versions 1.1 (March) and 1.2 (September) that year.

Major results of this phase were the new framework architecture, improved documentation, and benchmarks. By the end of this phase, Kieker development moved completely to Kiel University.

Phase 4: 2011–2012 (Quality Assurance, SPEC Review)
In 2011, we started to use a number of additional development tools for continuous integration, issue tracking, and improving code quality (detailed in [vH14]). This was mainly driven by the successful application process for acceptance in the SPEC RG's repository of peer-reviewed tools for quantitative system evaluation and analysis—one of the core results of this phase.[2] The review process and the final acceptance for the tool repository have been a great success for Kieker for several reasons, e.g., the thorough reviews from an external perspective were extremely useful as they triggered a lot of activities in the Kieker project (including the infrastructure) and helped to further improve Kieker's product quality (including quality of code and documentation); Kieker's visibility was increased considerably. Version 1.3 (released in May 2011), the initial version submitted to the SPEC RG, included many new features. In version 1.4 (October 2011), which got accepted by the SPEC RG, the code quality was improved considerably based on the aforementioned development tool support. In 2012, we reworked Kieker's pipes-and-filters framework, introduced event-based tracing, and released a first version of the web-based UI for configuring and executing analysis configurations. Versions 1.5 (April) and 1.6 (October) were released in this year.

Major achievements during this phase were extensions to the feature set (including monitoring support for additional programming platforms), improvements to the code quality, and a number of additional case studies. The number of externals users grew. In Novem-

[2]The web site of the SPEC Research Group's repository of peer-reviewed tools for quantitative system evaluation and analysis is available at `http://research.spec.org/tools/`. Similar to the peer-reviewing process for scientific publications, submitted tools are thoroughly evaluated by a minimum number of three reviewers based on the following criteria: *i.)* relevance to the system evaluation community, *ii.)* overall utility, *iii.)* originality or novelty, *iv.)* tool maturity/user base, *v.)* ease-of-use and quality of documentation.

ber 2012, we welcomed 50 participants from academia and industry to our first Kieker Days (KoSSE Symposium on Application Performance Management).

Phase 5: 2013–today (Distributed Development and Community Building)
In this phase, the University of Stuttgart joined the Kieker development. Since ad-hoc meetings in person between developers in Kiel and Stuttgart became more difficult, we scheduled weekly developer meetings via a web conference system to discuss technical topics. This system has since then also been used for the monthly regular meetings.[3]

As follow-up events of our first Kieker Days in 2012, we organized the Symposium on Software Performance (SOSP) in Karlsruhe (2013) and in Stuttgart (2014) as joint performance community meetings with the related research groups Descartes and Palladio.[4] Both meetings attracted more than 50 and 60 participants respectively. The 2015 edition of the symposium will take place in Munich in November.

2.2 Research and Teaching Context

Kieker has been and is being developed in the context of different research and teaching activities. In most cases, Kieker is being employed for proof-of-concept implementations and quantitative evaluations of developed approaches. Kieker benefits from these activities in different forms of contributions and to different degrees of extent. Typical contributions include: *i.)* feedback with respect to documentation, framework usability and maturity, bug reports, etc., *ii.)* new application scenarios and case studies, *iii.)* refined or newly introduced features, as well as *iv.)* resources, e.g., in terms of technical infrastructure and funding for technical and academic staff and student assistants working for related research projects. The remainder of this section provides some insights into Kieker's research and teaching contexts. The research projects, including their start and end times (applying to completed projects), are also listed in the timeline in Figure 1. A list of references is available on the afore-mentioned Kieker web site.

Research Projects and Technology Transfer

Own Research Projects. Initially, Kieker has been developed at the University of Oldenburg in the Software Engineering Group as part of the DFG-funded Graduate School on Trustworthy Software Systems (TrustSoft). In 2008, Kiel University's Software Engineering Group joined development along with Prof. Hasselbring's move to Kiel. In 2011, Kieker development moved to Kiel completely. Since 2013, the University of Stuttgart's Reliable Software Systems Group joined Kieker development along with van Hoorn's move to Stuttgart. A number of Kieker-related third-party projects have been and are being conducted, e.g., DynaMod (2011–2012), PubFlow (since 2011), iObserve (since 2012), and diagnoseIT (since 2015). Several Kieker-related PhD theses, each of it being a research project for itself, have been and are being conducted both as part of the afore-mentioned third-party projects and the basic funding from the involved universities. A

[3] The agendas of all Kieker meetings are available at `http://trac.kieker-monitoring.net/wiki/Meetings/`

[4] The web site of the Symposium on Software Performance is available at `http://www.performance-symposium.org/`

complete list of research projects can be found on the afore-mentioned Kieker web site.

Use by Other Researchers. Kieker is not only used by us as the framework developers but also by others. Particularly in the research context, this is indicated by respective publications. Examples include the use of Kieker for research papers (e.g., [Dab12], [MDTS13], [OvHKV13], and [ZOLL11]), theses (e.g., [Ebe11], [Heg12], [Her12] [Wer12], and [Zob12]), and research tools (e.g., [Bar14], [BKR09]). External Kieker users come, e.g., from the Karlsruhe Institute of Technology, RWTH Aachen University, University of Würzburg, University of Novi Sad (Serbia), Warsaw University (Poland), and Xiâan Jiatong University (China). Some of these external works have also been conducted in collaboration with industrial partners such as SAP, Capgemini, and IBM.

Industrial Collaborations. During the past years, we had a number of industrial collaborations that involved the application of Kieker for dynamic analysis of production systems and, as part of this, influenced the development of Kieker, e.g., by feature requests, feedback, and code contributions. We will briefly discuss the industrial collaborations having most impact on the evolution and evaluation of Kieker. These collaborations and case studies also served as a qualitative evaluation of the Kieker approach, e.g., concerning fine-grained continuous monitoring on application level and requirements for production scenarios, e.g., w.r.t. logging.

Examples for industrial collaborations with impact on Kieker's development include the following. The afore-mentioned work by Focke [Foc06] initiated a collaboration with EWE TEL GmbH, Oldenburg—one of the largest regional telecommunication providers in the north of Germany. In 2008–2010, we continued this collaboration (see also [vHRH+09]). An EWE TEL developer contributed to Kieker's distributed tracing functionality, particularly via SOAP, and integrated Kieker in the production system, where it was in use for more than half a year [vH14]. In 2008, we started a collaboration with CEWE COLOR AG & Co. OHG, Oldenburg—Europe's largest digital photo service provider. In a case study, we instrumented a part one front-end server node of the Java EE-based load-balanced production system—a web portal providing services, such as ordering of photo prints and other photo products (see also [RvHH+10]). A CEWE COLOR developer contributed to Kieker's Servlet- and Spring-based probes for collecting trace information, and integrated Kieker in the production system, where it was in use for one week. Between 2011 and 2013, as part of two Diploma theses, we collaborated with XING AG, Hamburg—the wide-spread social network for business contacts with more than 12 million registered members as of September 2012. XING's core system, `http://xing.com`, served as a case study to evaluate Kieker's OPAD approach for automatic performance anomaly detection, developed in the theses. OPAD is implemented as a Kieker analysis plugin and has been integrated in XING's monitoring architecture. In the context of the DynaMod research project [vHFG+11] (01/2011–12/2012), we collaborated with the companies *i.)* b+m Informatik AG, Melsdorf, *ii.)* Dataport AöR, Altenholz, and *iii.)* HSH Nordbank AG, Kiel. The Kieker monitoring adapters for Visual Basic 6 and .NET, which have been developed as part of the DynaMod project, were employed to analyze the case study systems AIDA-SH (Dataport) and Nordic Analytics (HSH Nordbank). In the DynaMod context and beyond, b+m Informatik developers contributed to Kieker, e.g., in terms of functionality and bug fixes already included in recent Kieker re-

leases. Since 2012, Kieker is integrated in b+m Informatik's generative platform b+m gear [SV06]. Additional case studies were conducted for dynamic analysis of COBOL and Perl systems [KvHGH12, Ric12]. Since 2013, we are closely collaborating with NovaTec Consulting GmbH on various topics in the scope of application performance management (not limited to the diagnoseIT research project).

Teaching Kieker has been and is being used in different teaching courses conducted at the involved universities. Examples include student assignments and guest presentations as part of lectures on software engineering and parallel/distributed systems, development projects of groups of students, and theses (Bachelor's, Master's, and Diploma).

2.3 Contributors

Many colleagues contributed to Kieker in different ways and intensities.[5] Note that we do not only consider contributions to source code. The group of contributors can be divided into researchers and students affiliated with the involved universities, as well as externals, i.e., members from other academic or industrial institutions. The contributing researchers are usually involved because they are working on a Kieker-related research project (including PhD theses). Students usually contribute to Kieker as part of their work on Kieker-related study theses or their employment as student assistants. Externals usually contribute to Kieker during the course of collaborative projects (including papers).

3 Lessons Learned and Success Factors

Looking back, a crucial success factor for establishing Kieker was the early deployment in production systems (Phase 2). Such deployment environments put significant demands on the quality (in our case particularly for performance and reliability) on the software and its development process. Only with sufficient quality, software may be employed successfully in technology transfer projects. Another boost came from the rigorous review process by the SPEC Research Group (Phase 4), which implied significant extensions to the quality assurance in our continuous integration setting. Kieker is used in technology transfer such as the projects DynaMod [vHFG+11] and MENGES [GvHH+12], and diagnoseIT, but also in DFG-funded projects such as iObserve [HHJ+13] and PubFlow [BH13]. Besides research, we use Kieker as example software system for software engineering teaching.

Kieker's architecture is designed as a component-based system for extensibility to allow for custom extensions. The development of features is mainly driven by requests from research, technology transfer projects, and industry projects, rather than market research. This way, we keep the architecture lean and extensible. Kieker is licensed under the Apache License, Version 2.0, such that it my be exploited commercially without any restrictions. Such a license is a good legal framework for technology transfer. The "business model" of the contributing research groups is not based on envisioned revenue via licensing, instead we follow an open source business model based on *impact* of the software. More frequent use of the software means more impact, in this case. Such impact is a great foundation for follow-up projects. Besides projects, we also provide professional coaching and training for the software.

[5] A list of Kieker contributors is available at `http://kieker-monitoring.net/framework/`

As kind of spin-off projects, ExplorViz [FWWH13, FRH15] emerged as a tool for 3D visualization of monitoring logs (including Kieker logs), on screen [FFHW15], print [FKH15a] and in virtual reality [FKH15b], TeeTime [WEH14] as a possible basis for advanced Kieker analysis pipelines, and Hora [PvHG14] as a framework for hierarchical online failure prediction.

References

[Bar14] Cezary Bartoszuk. Callcount Project. `https://github.com/cbart/fly/tree/master/Callcount`, 2014.

[BH13] Peer C. Brauer and Wilhelm Hasselbring. PubFlow: A scientific data publication framework for marine science. In *Proceedings of the International Conference on Marine Data and Information Systems (IMDIS 2013)*, volume 54, pages 29–31, 2013.

[BKR09] Steffen Becker, Heiko Koziolek, and Ralf Reussner. The Palladio component model for Model-Driven Performance Prediction. *Elsevier Journal of Systems and Software (JSS)*, 82(1):3–22, 2009.

[Dab12] Robert Dabrowski. On Architecture Warehouses and Software Intelligence. In *Proceedings of the 4th International Mega-Conference on Future Generation Information Technology (FGIT 2012)*, volume 7709 of *LNCS*, pages 251–262. Springer, 2012.

[Ebe11] Stefan Eberlein. Erhebung und Analyse von Kennzahlen aus dem fachlichen Performance-Monitoring, 2011. Diploma Thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany.

[EvHWH11] Jens Ehlers, André van Hoorn, Jan Waller, and Wilhelm Hasselbring. Self-Adaptive Software System Monitoring for Performance Anomaly Localization. In *Proceedings of the 8th IEEE/ACM International Conference on Autonomic Computing (ICAC 2011)*, pages 197–200. ACM, 2011.

[FFHW15] Florian Fittkau, Santje Finke, Wilhelm Hasselbring, and Jan Waller. Comparing Trace Visualizations for Program Comprehension through Controlled Experiments. In *Proceedings of the IEEE International Conference on Program Comprehension (ICPC 2015)*, 2015.

[FKH15a] Florian Fittkau, Erik Koppenhagen, and Wilhelm Hasselbring. Research Perspective on Supporting Software Engineering via Physical 3D Models. In *Proceedings of the 3rd IEEE International Working Conference on Software Visualization (VISSOFT 2015)*, 2015.

[FKH15b] Florian Fittkau, Alexander Krause, and Wilhelm Hasselbring. Exploring Software Cities in Virtual Reality. In *Proceedings of the 3rd IEEE International Working Conference on Software Visualization (VISSOFT 2015)*, 2015.

[Foc06] Thilo Focke. Performance Monitoring von Middleware-basierten Applikationen. Diplomarbeit, University Oldenburg, 2006.

[FRH15] Florian Fittkau, Sascha Roth, and Wilhelm Hasselbring. ExplorViz: Visual Runtime Behavior Analysis of Enterprise Application Landscapes. In *Proceedings of the European Conference on Information Systems (ECIS 2015 Completed Research Papers)*. AIS Electronic Library, 2015.

[FWWH13] Florian Fittkau, Jan Waller, Christian Wulf, and Wilhelm Hasselbring. Live Trace Visualization for Comprehending Large Software Landscapes: The ExplorViz Approach. In *Proceedings of the IEEE International Working Conference on Software Visualization (VISSOFT 2013)*, 2013.

[GvHH+12] Wolfgang Goerigk, Reinhard von Hanxleden, Wilhelm Hasselbring, Gregor Hennings, Reiner Jung, Holger Neustock, Heiko Schaefer, Christian Schneider, Elferik Schultz,

Thomas Stahl, Steffen Weik, and Stefan Zeug. Entwurf einer domänenspezifischen Sprache für elektronische Stellwerke. In *Software Engineering 2012*, volume P-198 of *LNI*, pages 119–130. GI, 2012.

[Heg12] Christoph Heger. Automatische Problemdiagnose in Performance-Unit-Tests, 2012. Master's Thesis, Karlsruhe Institute of Technology.

[Her12] Nikolas Roman Herbst. Workload Classification and Forecasting, 2012. Diploma Thesis, Karlsruhe Institute of Technology.

[HHJ+13] Wilhelm Hasselbring, Robert Heinrich, Reiner Jung, Andreas Metzger, Klaus Pohl, Ralf Reussner, and Eric Schmieders. iObserve: Integrated Observation and Modeling Techniques to Support Adaptation and Evolution of Software Systems. Forschungsbericht, Kiel University, Kiel, Germany, 2013.

[KNP11] Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of Probabilistic Real-time Systems. In *Proceedings of the 23rd International Conference on Computer Aided Verification (CAV '11)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[KvHGH12] Holger Knoche, André van Hoorn, Wolfgang Goerigk, and Wilhelm Hasselbring. Automated Source-Level Instrumentation for Dynamic Dependency Analysis of COBOL systems. In *Proceedings of the 14th Workshop Software-Reengineering (WSR '12)*, pages 33–34, 2012.

[MDTS13] Vladimir Markovets, Robert Dabrowski, Grzegorz Timoszuk, and Krzysztof Stencel. Know Thy Source Code. In *Proceedings of the 6th Balkan Conference in Informatics (BCI '13)*, volume 1036, pages 128–131. CEUR-WS.org, 2013.

[MRvHH09] Nina S. Marwede, Matthias Rohr, André van Hoorn, and Wilhelm Hasselbring. Automatic Failure Diagnosis in Distributed Large-Scale Software Systems based on Timing Behavior Anomaly Correlation. In *Proceedings of the 13th European Conference on Software Maintenance and Reengineering (CSMR'09)*, pages 47–57. IEEE, 2009.

[OvHKV13] Dušan Okanović, André van Hoorn, Zora Konjović, and Milan Vidaković. SLA-Driven Adaptive Monitoring of Distributed Applications for Performance Problem Localization. *Computer Science and Information Systems (ComSIS)*, 10(10):26–51, 2013.

[PvHG14] Teerat Pitakrat, André van Hoorn, and Lars Grunske. Increasing Dependability of Component-based Software Systems by Online Failure Prediction. In *Proceedings of the 10th European Dependable Computing Conference (EDCC '14)*, pages 78–81. IEEE, 2014.

[R D08] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008.

[RGH07] Matthias Rohr, Simon Giesecke, and Wilhelm Hasselbring. Timing Behavior Anomaly Detection in Enterprise Information Systems. In *Proceedings of the 9th International Conference on Enterprise Information Systems (ICEIS'07)*, pages 494–497. INSTICC Press, 2007.

[Ric12] Bettual Richter. Dynamische Analyse von COBOL-Systemarchitekturen zum modellbasierten Testen ("Dynamic analysis of COBOL system architectures for model-based testing", in German), 2012. Diploma Thesis, Kiel University.

[Roh14] Matthias Rohr. *Workload-sensitive Timing Behavior Analysis for Fault Localization in Software Systems*. Kiel, Germany, 2014. Dissertation, Faculty of Engineering, Kiel University.

[RvHH+10] Matthias Rohr, André van Hoorn, Wilhelm Hasselbring, Marco Lübcke, and Sergej Alekseev. Workload-Intensity-Sensitive Timing Behavior Analysis for Distributed Multi-User Software Systems. In *Proceedings of the 1st Joint WOSP/SIPEW International Conference on Performance Engineering (WOSP/SIPEW '10)*, pages 87–92. ACM, 2010.

[RvHM+08] Matthias Rohr, André van Hoorn, Jasminka Matevska, Nils Sommer, Lena Stöver, Simon Giesecke, and Wilhelm Hasselbring. Kieker: Continuous Monitoring and on Demand Visualization of Java Software Behavior. In *Proceedings of the IASTED International Conference on Software Engineering 2008 (SE '08)*, pages 80–85. ACTA Press, 2008.

[SV06] Thomas Stahl and Markus Völter. *Model-Driven Software Development – Technology, Engineering, Management*. Wiley & Sons, 2006.

[vH07] André van Hoorn. Workload-sensitive Timing Behavior Anomaly Detection in Large Software Systems, 2007. Master's thesis (Diplomarbeit), Department of Computer Science, University of Oldenburg, Germany. 125 pages.

[vH14] André van Hoorn. *Model-Driven Online Capacity Management for Component-Based Software Systems*. Number 2014/6 in Kiel Computer Science Series. Department of Computer Science, Kiel University, Kiel, Germany, 2014. Dissertation, Faculty of Engineering, Kiel University.

[vHFG+11] André van Hoorn, Sören Frey, Wolfgang Goerigk, Wilhelm Hasselbring, Holger Knoche, Sönke Köster, Harald Krause, Marcus Porembski, Thomas Stahl, Marcus Steinkamp, and Norman Wittmüss. DynaMod Project: Dynamic Analysis for Model-Driven Software Modernization. In *Proceedings of the International Workshop on Model-Driven Software Migration (MDSM) 2011*, volume 708, pages 12–13. CEUR, 2011.

[vHRH+09] André van Hoorn, Matthias Rohr, Wilhelm Hasselbring, Jan Waller, Jens Ehlers, Sören Frey, and Dennis Kieselhorst. Continuous Monitoring of Software Services: Design and Application of the Kieker Framework. Technical Report TR-0921, Department of Computer Science, Kiel University, Germany, 2009.

[vHWH12] André van Hoorn, Jan Waller, and Wilhelm Hasselbring. Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis. In *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering (ICPE '12)*, pages 247–248. ACM, 2012.

[WEH14] Christian Wulf, Nils Christian Ehmke, and Wilhelm Hasselbring. Toward a Generic and Concurrency-Aware Pipes & Filters Framework. In *Proceedings of the on Software Performance 2014: Joint Descartes/Kieker/Palladio Days*, pages 70–82, 2014.

[Wer12] Alexander Wert. Uncovering Performance Antipatterns by Systematic Experiments, 2012. Master's Thesis, Karlsruhe Institute of Technology.

[Zob12] Christian Zobel. Monitoring komplexer verteilter Softwaresysteme, 2012. Master's Thesis, Hochschule Mannheim, University of Applied Sciences.

[ZOLL11] Qinghua Zheng, Zhijiang Ou, Linfeng Liu, and Ting Liu. A Novel Method on Software Structure Evaluation. In *Proceedings of the 2nd IEEE International Conference on Software Engineering and Service (ICSESS '11)*, pages 251–254. IEEE, 2011.