Kieker: A monitoring framework for software engineering research

Wilhelm Hasselbring a,*, André van Hoorn b

a Software Engineering Group, Kiel University, D-24098 Kiel, Germany
b Reliable Software Systems Group, University of Stuttgart, D-70569 Stuttgart, Germany

A R T I C L E I N F O

Keywords: Software engineering research Monitoring Dynamic analysis Software performance Reverse engineering

A B S T R A C T

Application-level monitoring and dynamic analysis of software systems are a basis for various tasks in software engineering research, such as performance evaluation and reverse engineering. The Kieker framework provides monitoring, analysis, and visualization support for these purposes. It commenced in 2006, and grew toward a high-quality open-source software that has been employed in a variety of software engineering research projects over the last decade. Several research groups constitute the open-source community to advance the Kieker framework. In this paper, we review Kieker’s history, development, and impact both in research and technology transfer with industry.

Code metadata

<table>
<thead>
<tr>
<th>Code metadata</th>
<th>1.14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current code version</td>
<td><a href="https://github.com/SoftwareImpacts/SIMPAC-2020-18">https://github.com/SoftwareImpacts/SIMPAC-2020-18</a></td>
</tr>
<tr>
<td>Permanent link to code/repository used for this code version</td>
<td>Apache License, Version 2.0</td>
</tr>
<tr>
<td>Legal Code License</td>
<td>Git</td>
</tr>
<tr>
<td>Code versioning system used</td>
<td>Java</td>
</tr>
<tr>
<td>Software code languages, tools, and services used</td>
<td>Java 7 or higher</td>
</tr>
<tr>
<td>Compilation requirements, operating environments &amp; dependencies</td>
<td><a href="http://kieker-monitoring.net/documentation/">http://kieker-monitoring.net/documentation/</a></td>
</tr>
<tr>
<td>If available Link to developer documentation/manual</td>
<td><a href="http://kieker-monitoring.net/support/">http://kieker-monitoring.net/support/</a></td>
</tr>
<tr>
<td>Support email for questions</td>
<td></td>
</tr>
</tbody>
</table>

Software metadata

<table>
<thead>
<tr>
<th>Software metadata</th>
<th>1.14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current software version</td>
<td><a href="http://kieker-monitoring.net/download/">http://kieker-monitoring.net/download/</a></td>
</tr>
<tr>
<td>Permanent link to executables of this version</td>
<td>Apache License, Version 2.0</td>
</tr>
<tr>
<td>Legal Software License</td>
<td></td>
</tr>
<tr>
<td>Computing platforms/Operating Systems</td>
<td>Various, including Linux, OS X, Microsoft Windows, Unix-like</td>
</tr>
<tr>
<td>Installation requirements &amp; dependencies</td>
<td>Java 7 or higher, and possibly other technologies depending on the use case</td>
</tr>
<tr>
<td>Link to user manual</td>
<td><a href="http://kieker-monitoring.net/documentation/">http://kieker-monitoring.net/documentation/</a></td>
</tr>
<tr>
<td>Support email for questions</td>
<td><a href="http://kieker-monitoring.net/support/">http://kieker-monitoring.net/support/</a></td>
</tr>
</tbody>
</table>

1. Application-level monitoring

The Kieker application-level monitoring framework is structured into a monitoring and an analysis part. On the monitoring side, monitoring probes collect measurements of instrumented software systems represented as monitoring records, which a monitoring writer passes to a configured monitoring log or stream. On the analysis side, monitoring readers import monitoring records of interest from the monitoring log/stream and pass them to a configurable pipe-and-filter architecture of analysis plugins. Fig. 1 illustrates a typical dynamic analysis workflow with Kieker. Focusing on application-level monitoring, Kieker includes monitoring probes for collecting timing and trace information from distributed executions of software systems. Additionally, probes for sampling hardware measures, e.g., CPU utilization and memory usage, are included.

* Corresponding author.
E-mail addresses: hasselbring@email.uni-kiel.de (W. Hasselbrig), van.hoorn@informatik.uni-stuttgart.de (A. van Hoorn).

https://doi.org/10.1016/j.simpa.2020.100019
Received 29 May 2020; Received in revised form 3 June 2020; Accepted 4 June 2020

2665-9638/© 2020 Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
The core research topics in Kieker are Application Performance Monitoring and Architecture Discovery, as well as the combination of both topics:

**Application Performance Monitoring** is concerned with continuously observing a software system's performance-specific runtime behavior, including analyses like assessing service-level compliance or detecting and diagnosing performance problems. Kieker provides Application Performance Monitoring support for both collecting and analyzing monitoring data.

**Architecture Discovery** is concerned with extracting architectural information from an existing software system, including both structural and behavioral aspects like identifying architectural entities (e.g., components and classes) and their interactions (e.g., local or remote procedure calls). Kieker provides Architecture Discovery support by monitoring and analyzing information about distributed execution traces and about consumed software and hardware resources.

### 2. Development history

Kieker development started in 2006 as a tool for monitoring response times of Java software operations [2]. In 2009, we considerably restructured Kieker toward a generic and extensible framework architecture [3]. In 2011, we significantly improved the quality management with continuous integration, issue tracking, and static code analysis. This process 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. 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 very useful as they triggered a lot of activities in the Kieker project and helped to further improve Kieker’s product quality (both code and documentation); Kieker’s visibility was increased considerably [1]. Recently, the Kieker analysis part was re-implemented with the high-throughput pipe-and-filter framework TeeTime [4–6].

#### 3. Research impact

Research areas where Kieker was successfully employed for software engineering research include analysis of software structure and metrics [7–14], performance analysis [15–19], timing behavior anomaly detection [20–23], online failure prediction [24], fault localization [25], online capacity management [26–28], self-adaptive monitoring [22,29,30], software architecture reconstruction [31,32], regression benchmarking [33,34], test case prioritization [35,36], and extraction of load profiles and usage models [37,38].

A number of Kieker-related research projects have been and are being conducted over the years. Examples are Declare (Declarative Performance Engineering) [39], diagnoseIT (Expert-Guided Automatic Diagnosis of Performance Problems) [40], ContiNuTy (Automated Load Testing in Continuous Software Engineering) [41], Orcas (Efficient Resilience Benchmarking of Microservice Architectures) [42], MooBench (Monitoring Overhead Benchmark) [33,43,44], iQAssistant (Integrated Observation and Modeling Techniques to Support Adaptation and Evolution of Software Systems) [45,46] and PubFlow (Workflows for Research Data Publication) [47].

On top of the Kieker framework, ExplorViz [48–50] and SynchroVis [51] emerged as tools for 3D visualization of Kieker monitoring logs. In these contexts, Kieker supports research on program comprehension [52,53], including 3D printing [54] and virtual reality [55]. A jQAssistant plugin provides graph-based visualizations of Kieker traces [56].

#### 4. Industry impact

During the past years, Kieker was employed in several industrial collaborations and technology transfer projects. This includes the funded technology-transfer projects DynaMod (Dynamic Analysis for Model-Driven Software Modernization) [57], MENGES (Model-Driven Engineering of Rail Control Centers) [58], and diagnoseIT (Expert-guided automatic diagnosis of performance problems in enterprise applications) [40]. Examples for such industrial collaborations with impact on Kieker’s development include the following:

- In collaboration with EWE TEL GmbH – one of the largest regional telecommunication providers in the north of Germany – their web-based Customer Self Service system was instrumented with Kieker for improved server capacity management.
- In a collaboration with CEWE COLOR AG &Co. OHG – Europe’s largest digital photo service provider – we instrumented their web portal providing services for ordering photo products.

---

1. 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 software tools are thoroughly evaluated by at least 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.
• We collaborated with XING AG, Hamburg—a social network for business contacts. XING’s core system served as a case study to evaluate Kieker’s approach for automatic performance anomaly detection.
• The Kieker monitoring adaptors for Visual Basic 6 and .NET have been developed in collaboration with DataPort A6R and HSH Nordbank AG as part of the DynaMod project to analyze the case study systems AIDA-SH and Nordic Analytics, respectively.
• Additional case studies were conducted for dynamic analysis of legacy COBOL [59] and Perl [16] systems.
• Since 2012, Kieker is integrated in b+m Informatik’s generative software development platform b+m gear [60].
• Since 2013, we are collaborating with Novatec Consulting GmbH on various topics around application performance management (APM) [61], including works on APM tool interoperability [62], as part of the mentioned diagnoseIT and ContinuITy projects.

These industrial collaborations and case studies also serve as evaluation of the Kieker approach, influence the development of Kieker, e.g., by feature requests, feedback, code contributions, and provide access to real-world data.

5. Community building

In November 2012, we welcomed 50 participants from academia and industry to our first so-called Kieker Days for developers and users of Kieker. As follow-up events, we organize the annual “Symposium on Software Performance” since 2013 as joint performance community meetings with the related research groups Descartes [63] and Palladio [64]. The 2019 edition of the symposium took place in Würzburg in November 2019 and the 2020 edition will be held in Leipzig.

6. Lessons learned and success factors

Looking back, a crucial success factor for establishing Kieker was the early deployment in industrial production systems. 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 reused by other researchers and employed successfully in technology transfer projects. Another boost came from the rigorous review process by the SPEC Research Group, which implied significant extensions to the quality assurance in our continuous integration setting.

Kieker’s architecture is designed as a component-based system for extensibility to allow for custom extensions. Such a modular architecture significantly improves the collaboration in open source research software projects [65]. 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 may 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 research and technology transfer, we provide professional coaching and training for Kieker. We also use Kieker as example software system for software engineering education.

Declarations of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Many colleagues contributed to Kieker in different ways and intensities. Note that we do not only acknowledge 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.

References


• The web site of the annual Symposium on Software Performance is available at http://www.performance-symposium.org/.

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


