Open-Source-Software als Katalysator im Technologietransfer am Beispiel des Monitoring-Frameworks Kieker

Wilhelm Hasselbring\textsuperscript{1} & André van Hoorn\textsuperscript{2}

\textsuperscript{1} Kiel University (CAU)
Software Engineering Group
&
\textsuperscript{2} University of Stuttgart
Reliable Software Systems Group

1. Kieler Open Source Business Konferenz
14. September 2015 @ Kiel
Monitoring on different system layers

Overview

Self-adaptive Software Performance Monitoring for Anomaly Localization

Application-level Monitoring
- Observations in field
  - Extensive infrastructure monitoring, application monitoring not widespread
  - Reactive monitoring probe injection only (after a critical performance drop has occurred)

Business Processes
- Key performance indicators, e.g. process throughput, ...
  - SLO appliance, workload, ...

Services
- Response times, operational profile, ...
  - Thread/connection pool sizes, ...

Application
- Heap size, ...
  - CPU/memory utilization, ...

Middleware Container
- Availability, reliability, ...

Virtual Machine

Operating System

Hardware

Monitoring practice in the “real world” (based on what we’ve seen)

- Focus on system level (network availability, resource utilization) or business level (key performance indicators)
- No systematic instrumentation on application level
- Monitoring as an “afterthought”: probes are only added when problems occurred.

W. Hasselbring, A. van Hoorn

Christian-Albrechts-Universität zu Kiel
“Java monitoring largely unknown.”

[codecentric GmbH 2009]
Scaling Facebook to 500 Million Users and Beyond

“Making lots of small changes and watching what happens only works if you’re actually able to watch what happens. At Facebook we **collect an enormous amount of data** — any particular server exports **tens or hundreds of metrics** that can be graphed. This isn’t just system level things like CPU and memory, it’s also **application level statistics** to understand why things are happening.

It’s important that the statistics are from the **real production machines** that are having the problems, when they’re having the problems – **the really interesting things only show up in production**. The stats also have to come from all machines, because a lot of important effects are hidden by averages and only show up in distributions, in particular 95th or 99th percentile.”

*Robert Johnson, Facebook Engineering Director*
Framework Features & Extension Points


Overview

- Modular, flexible, and extensible architecture (Probes, records, readers, writers, filters etc.)
- Pipes-and-filters framework for analysis configuration
- Distributed tracing (logging, reconstruction, visualization)
- Low overhead (designed for continuous operation)
- Evaluated in lab and industrial case studies

Kieker is open-source software (Apache License, V. 2.0)
http://kieker-monitoring.net

Recommended Tool of the SPEC Research Group

Kieker is distributed as part of SPEC RG’s repository of peer-reviewed tools for quantitative system evaluation and analysis,
http://research.spec.org/projects/tools.html
Dynamic Analysis with Kieker

Overview

- Software System with Monitoring Instrumentation
- Monitoring Probe
- Monitoring Log/Stream
- Measurement
- Monitoring Records
- Analysis
- Analysis Configuration (via API and WebGUI)
- Pipes and Filters
- Invocations/minute [x 1000]
- Calendar time (hh:mm)
- Workload Anomaly Detection
- Online and Offline Visualization

W. Hasselbring, A. van Hoorn
Kieker
Core Framework Components

Overview

Kieker.Monitoring

Monitoring

Probe

Monitoring Controller

Monitoring Writer

Monitoring Log/Stream

Kieker.Analysis

Pipe & Filter Configuration

Monitoring Reader

Analysis / Visualization Plugin

Analysis Controller

Java probes/samplers:

- Manual instrumentation
- Spring
- CXF/SOAP
- AspectJ
- Servlet
- <your interception technology>
- Sigar
- CPU utilization
- Memory usage
- <your technology>
- <your monitoring probe>

+ basic adapters for

- C#/.NET
- Visual Basic 6/COM
- COBOL

W. Hasselbring, A. van Hoorn

Kieker

14. September 2015 @ Kiel
Core Framework Components

Overview

Kieker.Monitoring

Monitoring Probe

Monitoring Controller

Monitoring Writer

JMX Interface

Periodic Sampling

Logging

Time Source

Monitoring Log/Stream

Monitoring Record

Kieker.Analysis

Monitoring Reader

Analysis / Visualization Plugin

Pipe & Filter Configuration

Analysis Controller

Monitoring Readers/Writers

File system

Java Messaging Service (JMS)

Java Management Ext. (JMX)

Database (SQL)

Named pipe

<your monitoring reader/writer>
Core Framework Components

Overview

Kieker.Monitoring

- Monitoring Probe
- Monitoring Controller
- Monitoring Writer
- JMX Interface
- Periodic Sampling
- Logging

Monitoring Log/Stream

- Monitoring Record

Kieker.Analysis

- Analysis Controller
- Analysis / Visualization Plugin
- Monitoring Reader
- Pipe & Filter Configuration

Visualization

- Architecture reconstruction
- Trace analysis
- Dependency graphs
- Sequence diagrams
- Call graphs
- <your visualization>
- <your trace analysis>
- <your reconstruction plugin>
- <your analysis plugin/tool>
Experiment similar to:

<table>
<thead>
<tr>
<th></th>
<th>A5</th>
<th>AsyncFS writer</th>
<th>16 cores</th>
<th>whole system is available</th>
</tr>
</thead>
</table>

Slope (m)

\[ m = 0.1 \]

\[ m = 0.9 \]
Regression Benchmarking
[Waller et al. 2015]

Overview

Mean Overhead of Kieker

https://build.se.informatik.uni-kiel.de/jenkins/job/kieker-nightly-release/plot/
Kieker Context: Application Examples

Overview

1. Architecture Discovery: Model Extraction and Visualization

2. Application Performance Management: Anomaly Detection + Diagnosis
Architecture Discovery: Model Extraction + Visualization (cont’d)

[van Hoorn et al. 2009]
APM: Anomaly Detection + Diagnosis (cont’d)

[Bielefeld 2012, Frotscher 2013]
What others are doing with Kieker

An example: Analysis of Calling Networks

Overview

In Figure 1: An excerpt from the "applicationContext.xml" file of JPetStore 6.

Software Method Call Process

Time

$t_1$ $t_2$ $t_3$ $t_n$

Software Method Call ... (b:B→d:D). The nodes c:C and c2:C are isolated because they are used by the staticmain() method rather than an object.

Scheme 1: Raw Calling Network

Scheme 2: Growing Calling Network

Scheme 3: Partitioned Calling Network

Xi’an Jiaotong University, Shaanxi

[Qu et al. 2015; 2014; 2013, Zheng et al. 2011]
Agenda

1. Overview

2. Review

3. Summary and Outlook
Looking back ... 2006–2009

Review
Looking back ... 2010–2013

Review

2010
- Menges
  - Kosse

2011
- DynaMod
- PubFlow

2012
- ESN SL
- EnSure
- Kieker Days '12

2013
- Kieker/Palladio Days '13

1.1 1.2
- JUnit

1.3 1.4
- FindBugs

1.5 1.6
- trac
- HSH Nordbank

1.7 1.8
- b+m
- dataport
- Xing
- NOVATEC

W. Hasselbring, A. van Hoorn
Kieker
14. September 2015 @ Kiel
References: Research and Industry

http://kieker-monitoring.net/research/references/

Review

Internal Researchers

Kiel University, Kiel, Germany – Researchers from the Kiel University’s Software Engineering Group investigate innovative techniques and methods for engineering, evolving, and operating continuously running software systems (research projects).

University of Stuttgart, Stuttgart, Germany – Researchers from the University of Stuttgart’s reliable Software Systems Group investigate innovative quantitative QoS analysis and forecasting methods for distributed software-intensive systems (research projects).

External Researchers

Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany – Researchers from the KIT’s Software Design and Quality Group are using Kieker for different purposes, e.g., for detecting and diagnosing performance problems in systematic experiments. We are also collaborating with KIT researchers in the context of the OBServe research project.

RWTH Aachen University, Aachen, Germany – Researchers from the RWTH Aachen University’s Software Construction Group are using Kieker for monitoring-based architecture reconstruction in their ARAMIS project on model-based software architecture evolution and analysis.

University of Novi Sad, Novi Sad, Serbia – Researchers from the University of Novi Sad were using Kieker for adaptive monitoring of software systems in the context of performance problem detection and diagnosis.

University of Würzburg, Würzburg, Germany – Researchers from the University of Würzburg’s Software Engineering Group are using Kieker for extracting performance models.

Warsaw University, Warsaw, Poland – Researchers from the Warsaw University employed Kieker for dynamic data acquisition of software architectures.

Xi’an Jiaotong University, Xi’an, Shaanxi, China – Researchers from the Xi’an Jiaotong University used Kieker for discovering architectural structures in software systems and to analyze software call graphs.

Industry

b+m Informatik AG, Metzdorf, Germany – With b+m, we collaborated in the context of the DynaMod and MENOCs research projects. Moreover, Kieker is being used by b+m, e.g., for architecture discovery of large-scale COBOL mainframe systems. Contributions by b+m are part of the Kieker release.

CEWE COLOR AG & Co. OHG, Oldenburg, Germany – With CEWE COLOR, we collaborated in the context of the TrustSoft research project. CEWE COLOR provided a JavaEE-based web portal as a case study system for application performance monitoring. Contributions by CEWE COLOR are part of the Kieker release.

Dataport AG, Altenholz, Germany – With Dataport, we collaborated in the context of the DynaMod research project. Dataport provided a SQL-based case study system for architecture discovery based on hybrid analysis with Kieker.

Eprints Services, Southampton, United Kingdom – With EPrints Services, we collaborated in the context of several thesis projects. We employ the EPrints system as a case study system for software performance analysis with Kieker for Perl-based systems. The EPrints team provides an integration of Kieker with EPrints as eplkacer.

EWE TEL GmbH, Oldenburg, Germany – With EWE TEL, we collaborated in the context of the TrustSoft research project. EWE TEL provided a Java-based web portal as a case study system for application performance monitoring. Contributions by EWE TEL are part of the Kieker release.

HSI Nordbank AG, Kiel, Germany – With HSI Nordbank, we collaborated in the context of the DynaMod research project. The HSI provided a C-based function library for architecture discovery based on hybrid analysis with Kieker.

Novatec GmbH, Leinfelden-Echterdingen, Germany – With Novatec, we currently collaborate in the context of different teaching projects on application performance management. Novatec published a nice blog article about their first experiences with Kieker.

SAP Research, Karlsruhe, Germany – Kieker is used as a tool to collect performance data for the Software Performance Cockpit. We are also collaborating with SAP Research in the context of the OBServe research project.

XING AG, Hamburg, Germany – With XING, we collaborated in the context of a Diploma thesis on online performance anomaly detection (OPAD). XING provided its core system xing.com for evaluating the Kieker-based OPAD implementation.
Kieker: A framework for application performance monitoring and dynamic software analysis
A Van Hoorn, J Walter, W Hasselbring - Proceedings of the 3rd ACM/SPEC International ..., 2012
Cited by 115 - Related articles - All 4 versions

Continuous monitoring of software services: Design and application of the Kieker framework
A van Hoorn, M Rohr, W Hasselbring, J Walter, J Ehlers... - 2009
Cited by 70 - Related articles - All 5 versions

Kieker: Continuous monitoring and on demand visualization of Java software behavior
M Rohr, A van Hoorn, J Malevska, N Sommer... - 2005
Cited by 42 - Related articles - All 8 versions
Evolution of Kieker’s Code Size (’master’)

LOC obtained via `wc -l <file>.java`

Review

Lines of Code (branch ’master’)

Number of Java files (branch ’master’)

KLOC (LOC in thousands)

LOC (sources + tests)

LOC (tests only)

Month of year (from Aug 2008 to Jun 2015)

05 09 01 05 09 01 05 09 01 05 09 01 05 09 01 05 09 01 05 09 01 05

2009 2010 2011 2012

Number of Java files

source + test files
test files only

W. Hasselbring, A. van Hoorn

Kieker

14. September 2015 @ Kiel
Development Phases

Phase 1: 2006 (Inception)
Phase 2: 2007–2009 (Production Systems)
Phase 3: 2009–2010 (Restructuring)
Phase 4: 2011–2012 (Quality Assurance, SPEC Review)
Phase 5: 2013–today (Distributed Development and Community Building)
1 Overview

2 Review

3 Summary and Outlook
Lessons Learned

Summary and Outlook

- Open source tool can increase visibility in academia and industry
- Funding for research projects is essential
  - Incubator for technology transfer.
- Besides projects, we also provide professional coaching and training for the software
- Kieker is also used as example and object for software engineering teaching
- Success factors
  - A crucial success factor for establishing Kieker was the early deployment in production systems (Phase 2)
  - Another boost came from the rigorous review process by the SPEC Research Group (Phase 4)
- Licensing is also relevant
  - Kieker is licensed under the Apache License, Version 2.0
  - Impact!
Current Activities (Selection)

- Kieker Trace Diagnosis
  - Trace diagnosis tool to identify typical performance problems
- High-throughput infrastructure for Kieker Analysis
  - Based on TeeTime [Wulf et al. 2014], http://teetime.sf.net
- Interoperability between APM tools (open exchange formats)
- Docker-based Kieker example (NetflixOSS RSS reader application)
- Analysis of Kieker Development process and infrastructure
- Split Kieker into multiple, independent components [Hasselbring 2002]
  - under discussion …

Ticket System: Current/Upcoming Issues

http://trac.kieker-monitoring.net
Various additional people contributed to Kieker in the past years:


—Alphabetic list of people who contributed in different form (source code, bug reports, promotion, etc) and intensity
Open-Source Software as Catalyzer for Technology Transfer: Kieker’s Development and Lessons Learned

Wilhelm Hasselbring and André van Hoorn

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 widespread 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. 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.

The Kieker framework’s web site—including downloads, documentation, publications, and references—is available at http://kieker-monitoring.net
Visit http://kieker-monitoring.net

Summary and Outlook

Kieker is distributed as part of SPEC® RG's repository of peer-reviewed tools for quantitative system evaluation and analysis.

http://research.spec.org/projects/tools.html

W. Hasselbring, A. van Hoorn
http://kieker-monitoring.net/blog/
kiekers-development-in-five-minutes-1-10/
For a comprehensive list of publications, talks, and theses about Kieker, visit:
http://kieker-monitoring.net/research/


Summary and Outlook


